



ITSO: a novel inverse transform sampling-based optimization algorithm for stochastic search

Nikolaos P. Bakas¹ · Vagelis Plevris² · Andreas Langousis³ · Savvas A. Chatzichristofis⁴

Accepted: 13 April 2021

© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract

Optimization algorithms appear in the core calculations of numerous Artificial Intelligence (AI) and Machine Learning methods and Engineering and Business applications. Following recent works on AI's theoretical deficiencies, a rigorous context for the optimization problem of a *black-box* objective function is developed. The algorithm stems directly from the theory of probability, instead of presumed inspiration. Thus the convergence properties of the proposed methodology are inherently stable. In particular, the proposed optimizer utilizes an algorithmic implementation of the n -dimensional inverse transform sampling as a search strategy. No control parameters are required to be tuned, and the trade-off among exploration and exploitation is, by definition, satisfied. A theoretical proof is provided, concluding that when falling into the proposed framework, either directly or incidentally, any optimization algorithm converges. The numerical experiments verify the theoretical results on the efficacy of the algorithm apropos reaching the sought optimum.

Keywords Stochastic optimization · Inverse transform sampling · Black-box function · Global convergence

List of symbols

$\mathbf{x}^*$	The argument of f corresponding to the current optimum during the optimization process	h	Convergence history of optimization algorithms
$\mathbf{x}_m$	The sought $\arg \min f$	$i = \{1, 2, \dots, f_e\}$	Iterator for the f_e
A	Search space $\in \mathbb{R}^n$	$j = \{1, 2, \dots, n\}$	Iterator for the dimensions of A
		n	Number of dimensions of the set A
		f_e	The maximum function evaluations

✉ Nikolaos P. Bakas
n.bakas@cyi.ac.cy

Vagelis Plevris
vplevris@qu.edu.qa

Andreas Langousis
andlag@alum.mit.edu

Savvas A. Chatzichristofis
s.chatzichristofis@nup.ac.cy

¹ Computation-Based Science and Technology Research Center, The Cyprus Institute, 20 Konstantinou Kavafi Street, Aglantzia, 2121 Nicosia, Cyprus

² Department of Civil and Architectural Engineering, Qatar University, P.O. Box: 2713, Doha, Qatar

³ Department of Civil Engineering, University of Patras, 265 04 Patras, Greece

⁴ Intelligent Systems Lab & Department of Computer Science, Neapolis University Pafos, 2 Danais Avenue, 8042 Pafos, Cyprus

1 Introduction

Despite the numerous research works and industrial applications of Artificial Intelligence algorithms, they have been criticized about lacking a solid theoretical background (see e.g. Hutson 2018). The empirical results demonstrate impressive performance, however, their theoretical foundation and analysis are often vague (see e.g. Sculley et al. 2018). Machine Learning (ML) models, frequently aim at identifying an optimal solution (see e.g. Sra et al. 2012, and Bottou et al. 2018), which is computationally hard and attempts to explain the procedure often based on the evaluation of the function's gradients (see e.g. Rudin 2019, and Wu et al. 2017). Accordingly, the identification of whether a stochastic algorithm will work or not remains an open question for many research and real-world applications (see e.g. Doerr 2020a, b, Papadrakakis et al. 2005,

and Papadrakakis et al. 2001). A vast number of optimization algorithms have been developed and applied for the solution of engineering problems, such as Moayyeri et al. 2019, Plevris and Papadrakakis 2011, Lagaros et al. 2009, and Lagaros et al. 2006 in Structural Engineering, as well as Wang et al. 2019a, Cui et al. 2019, and Wang et al. 2019b in Environmental Engineering. Furthermore, optimization algorithms have been utilized in Financial modeling and portfolio optimization (see e.g. Lin et al. 2004, Gilli and Schumann 2012, Ponomareva et al. 2015), in Chemistry (e.g. Siddique and Adeli, 2017, Hibbert, 1993, Clayton et al., 2019), in Biology (e.g. Degasperis et al., 2017, Holden and Freitas, 2005, Chang et al. 2004), in Robotics (e.g. Kapoutsis et al. 2016, and Kapoutsis et al. 2019), etc. Accordingly, mathematical proofs regarding their algorithmic convergence have been developed (for instance see Bull 2011, Rudolph 1994, Clerc and Kennedy 2002), however their basic formulation often stems from nature-inspired procedures (see e.g. Opara and Arabas 2019, De et al. 2019) and not solid mathematical frameworks. Thus, many research works have been published to investigate the performance of black-box algorithms (see e.g. Muñoz and Smith-Miles 2017, Audet and Hare 2017, and Audet and Kokkolaras 2016).

In its elementary form, the purpose of efficient optimization algorithms is to find the argument yielding the minimum value of a *black-box* function $f(x)$, defined on a set A , $f: A \rightarrow \mathbb{R}^n$. Accordingly, the inverse problem of maximization is the minimization of the negated function $-f(x)$, while problems with multiple objective functions often utilize single function optimization algorithms to attain the best possible solution. Here A is assumed to be a compact subset of the Euclidean space $\mathbb{R}^n$, where n is the number of dimensions, however, the proposed method applies similarly to discrete and continuous topological spaces. The unknown, black-box function f , returns values for any given input $x_{ij} = \mathbf{X}_i = (x_{i1}, x_{i2}, \dots, x_{in})$ at each computational discrete time step $i = \{1, 2, \dots, f_e\}$, where f_e is the number of maximum function evaluations. The sought solution is a vector $\mathbf{x}_{min} \in A$, such that $f(\mathbf{x}_{min}) \leq f(\mathbf{x}), \forall \mathbf{x} \in A$, which may be written as:

$$\mathbf{x}_{min} = \arg \min f(\mathbf{x}) := \{\mathbf{x} \in A \subseteq \mathbb{R}^n \mid \forall \mathbf{y} \in A : f(\mathbf{y}) \geq f(\mathbf{x})\} \quad (1)$$

The purpose of this work is to provide a rigorous context for the optimization problem of a *black-box* function, by adhering to Probability Theory, aiming at identifying the best possible solution $\mathbf{x}_{min}$, within the given iterations f_e , during the execution of the algorithm.

The rest of the paper is organized as follows: In Sect. 2, the proposed Inverse Transform Sampling Optimizer (ITSO) is presented. Additionally, the same

Section provides details on the theoretical formulation of the algorithm and some programming and implementation techniques. Illustrative examples of the optimization history are also comprised. In Sect. 3 the theoretical proof of convergence is provided. Section 4 briefly presents the programming techniques that were investigated, to implement the proposed method into a computer code. Section 5 presents three groups of experiments to illustrate the efficiency of the algorithm. The comparison regards 13 non-linear loss functions, 17 other optimization methods, for 10 and 20 dimensions of search space, and 5000 and 10000 iterations per dimension. Finally, conclusions are drawn in Sect. 6.

2 Optimization by inverse transform sampling

Let the probability distribution of the optimal values of f , be considered as the product of some monotonically decreasing kernel k over f at some time-step i and hence given input $\mathbf{x}_i$. This assumption stands in the foundation of the method and can be considered as rational, instead of an arbitrary selection strategy, inspired by natural or other phenomena. It is a straightforward application of the Probability Theory to the problem. The kernel may be considered as parametric, concerning time i . The selection of the kernel k should satisfy the condition that its limit is the Dirac delta function centered at $\arg \min f$,

$$\lim_{i \rightarrow \infty} k_i(f) = \delta_m(x), \quad (2)$$

where δ_m denotes the Dirac function centered at $\arg \min f$. Although the selection of the kernel k is ambiguous, it can be chosen among a variety of functions satisfying Eq. 2, such as the Gaussian:

$$k_i(f) = \exp(-f(\mathbf{x}_i)^2 g(i)), \quad (3)$$

where $g(i)$ is a time increasing pattern. The function $g(i)$ controls the shape of the kernel k , approximating numerically Eq. 2. Additionally, we may use:

$$k_i(f) = \max f - f(\mathbf{x}_i), \quad (4)$$

$$k_i(f) = 1 - \frac{f(\mathbf{x}_i) - \min f + e_i}{\max f - \min f + e_i} \wedge e_i \rightarrow 0, \quad (5)$$

or any other non-negative Lebesgue-integrable function, which reverses the order of the given set of all $f_{\hat{i}}$, where $\hat{i}$ is the permutation of the indices $1, 2, \dots, i$, such that the sequence of $f_{\hat{i}}$ being monotonically strictly decreasing. The duplicate values of $f_{\hat{i}}$ should be extracted to avoid numerical instabilities. These duplicates often appeared in the empirical calculations, especially when the algorithm

was close to a local or global stationary point. A variety of kernel functions k were investigated, and the results weren't affected significantly, even when distorting $k(f)$ with some random noise $X' \sim \mathcal{U}(a, b) \forall k(f_i) \in (a, b)$.

Accordingly, for each dimension j of the vector space A , the marginal probability density function P_{X_j} is obtained numerically from the kernel function:

$$P_{X_j}(x_{ij}) = k(f(x_{ij})) \forall j \in \{1, 2, \dots, n\}. \quad (6)$$

$P_{X_j}(x_{ij})dx$ is the probability that $\arg \min f$ falls within the infinitesimal interval $[x_{ij} - dx/2, x_{ij} + dx/2]$. The corresponding cumulative distribution function F_{X_j} , can be calculated by:

$$F_{X_j}(x_{ij}) = \int_{lb_j}^{x_{ij}} P_{X_j}(\xi) d\xi, \quad (7)$$

where lb_j is the lower bound of the j th dimension of the vector space A and can be numerically evaluated by some numerical integration rule, such as the Riemann sum $S_j = \sum_{i=1}^n P_{X_j}(x_{ij}^*) \Delta x_{ij}$, with $P_{X_j}(x_{ij}^*)$ computed by the application of the kernel k on some x_{ij} , such that $f(x_{ij}^*) = (f(x_{ij}) + f(x_{i-1,j}))/2$, or another approximation scheme.

In the following Algorithm 1, the Inverse Transform Sampling method's algorithmic implementation is demonstrated. The symbols are also noted in the Nomenclature section.

Algorithm 1: ITSO-Mathematical Framework

```

Data:  $A, f_e, n$ 
Result:  $\mathbf{x}^* = \arg \min f, f^* = f(\mathbf{x}^*)$ 
1 while  $i \leq f_e$  do
2    $j \leftarrow \mathcal{U}\{1, n\}$ ;
3    $r_i \leftarrow \mathcal{U}(0, 1)$ ;
4   Sort  $x_{ij} \forall i$ ;
5    $x_{ij} \leftarrow F_j^{-1}(r_i)$ ;
6    $f_i = f(\mathbf{x}_i)$ ;
7   if  $f_i \leq f^*$  then
8      $f^* \leftarrow f_i$ ;
9      $x_j^* \leftarrow x_{ij}$ ;
10  else
11     $x_{ij} \leftarrow x_j^*$ ;
12  end
13 end
14 return  $\mathbf{x}^* = \arg \min f$ 
    
```

A graphical demonstration of the evolution of the Probability Density Functions (PDF), as well as the corresponding Cumulative Distribution Functions (CDF), is presented in Fig. 1, for $f(x) = (x - 5)^2$, which is a function with one extreme value and in Fig. 2, for $f(x) = \sin(x + 0.7) + 0.01 \times (x + 0.7)^2$, which has many extrema. Interestingly, as the function evaluations increase, the CDF is characterised by sharp slopes, which are positioned in regions where the PDF exhibits high values, and

hence function $f(x)$ returns low values. Figures 1 and 2 offer an intuitive representation of the procedure for finding the minimum of a function, within the suggested framework.

3 Convergence properties

During the optimization process, the algorithm generates some input variables x_{ij} as arguments for the *black-box* function f . By utilizing the values of f , we may compute the values of the distribution $P_{X_j}(x_{ij})$, by Eq. 6, and $F_{X_j}(x_{ij})$ by Eq. 7, for all x_{ij} .

Definition 1 We define the argument of f within f_e iterations, corresponding to the minimum of f as $x_m = \arg \min f$.

In iteration i , the algorithm will have searched the space $\hat{A} \subset A$, within the limits lb_j, ub_j . By definition, P is the probability density (likelihood) that the optimum occurs in a region $\mathbf{x} \pm \mathbf{dx}$. Hence,

$$E[\mathbf{X}] = \begin{Bmatrix} E[X_1] \\ E[X_2] \\ \vdots \\ E[X_n] \end{Bmatrix} = \begin{Bmatrix} \int_{lb_1}^{ub_1} \xi P_{X_1}(\xi) d\xi \\ \int_{lb_2}^{ub_2} \xi P_{X_2}(\xi) d\xi \\ \vdots \\ \int_{lb_n}^{ub_n} \xi P_{X_n}(\xi) d\xi \end{Bmatrix}, \quad (8)$$

where $E[\cdot]$, denotes the expectation of a random variable or vector.

Theorem 1 If P_{X_j} tends to Dirac δ , and a Global Minimum of the objective function f exists, then ITSO will converge to $\mathbf{x}_m = \arg \min f$.

Proof For each dimension j , we may write

$$E[X_j] = \int_{lb_j}^{ub_j} \xi P_{X_j}(\xi) d\xi = \int_{lb_j}^{ub_j} \xi F'_{X_j}(\xi) d\xi, \quad (9)$$

and integrating by parts, we obtain

$$\begin{aligned} E[X_j] &= [\xi F_{X_j}(\xi)]_{lb_j}^{ub_j} \\ &\quad - \int_{lb_j}^{ub_j} 1 F_{X_j}(\xi) d\xi = ub_j \times 1 - lb_j \times 0 - \int_{lb_j}^{ub_j} 1 F_{X_j}(\xi) d\xi. \end{aligned} \quad (10)$$

By applying the theorem of the antiderivative of inverse functions (Parker 1955), we obtain

$$\int_0^1 F_{X_j}^{-1}(y) dy + \int_{lb_j}^{ub_j} F_{X_j}(x) dx = ub_j \times 1 - lb_j \times 0. \quad (11)$$

Hence, for Eqs. 10 and 11 we deduce that

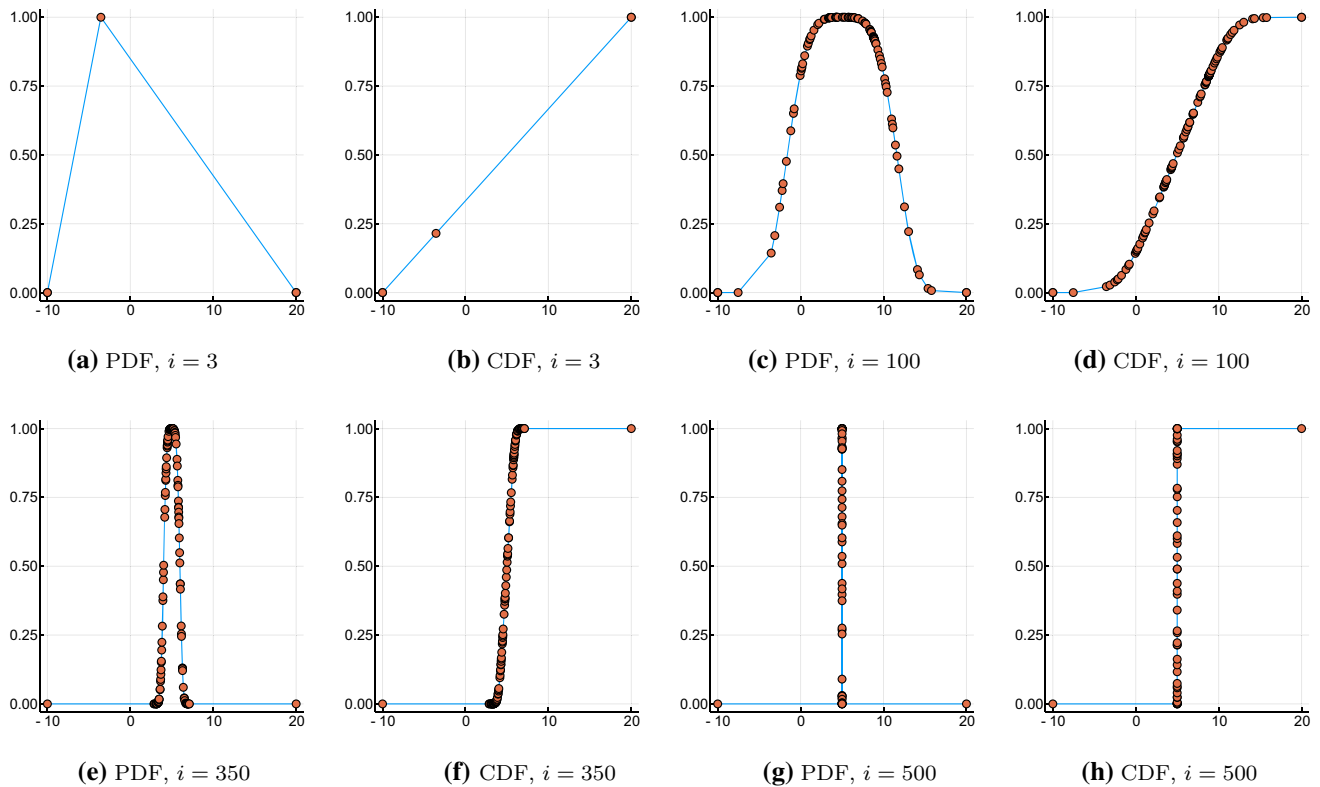


Fig. 1 Probability density and cumulative distribution functions, utilizing ITSO search strategy, for function $f(x) = (x - 5)^2$. The shape sequentially tends to the Heaviside function, centered at $x_m = 5$

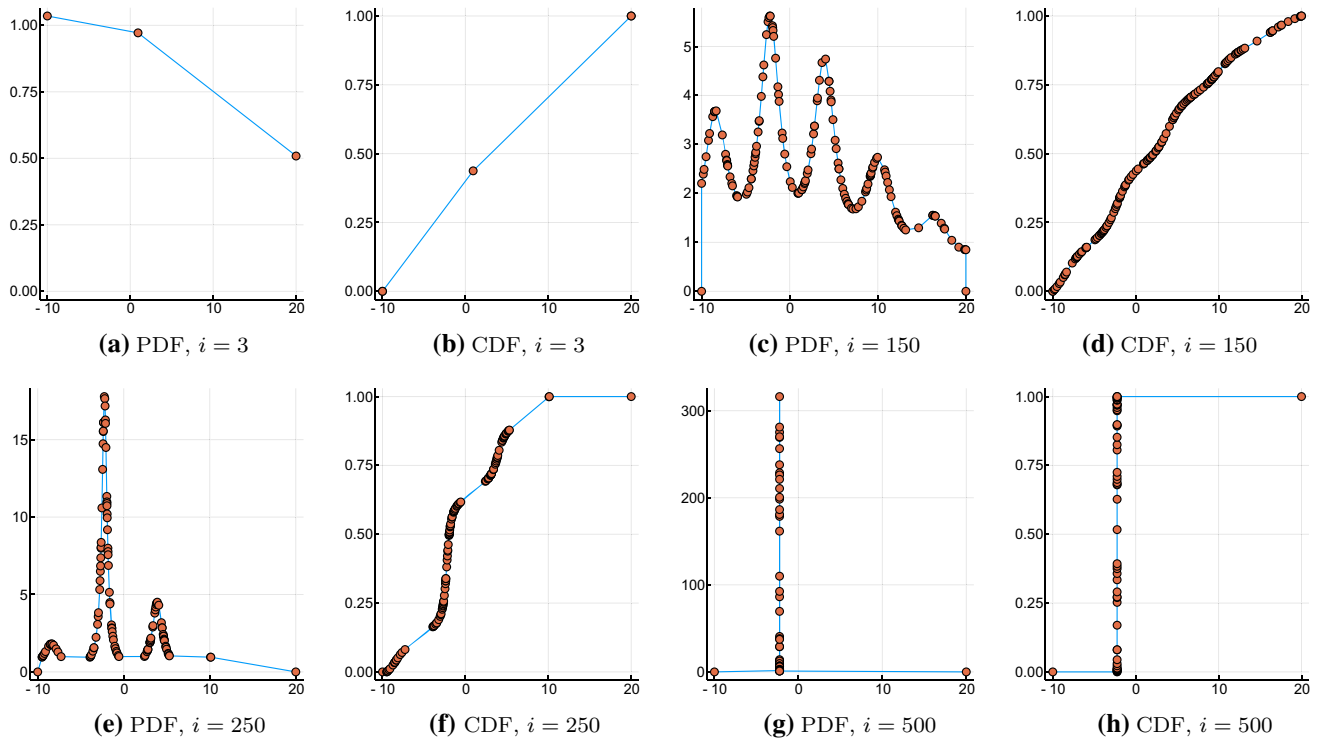


Fig. 2 Probability density and cumulative distribution functions, utilizing ITSO search strategy, for function $f(x) = \sin(x + 0.7) + 0.01 \times (x + 0.7)^2$. The shape sequentially tends to the Heaviside function, centered at $x_m = -2.24$

$$E[X_j] = \int_0^1 F_{X_j}^{-1}(y) dy. \quad (12)$$

If a Global Minimum of f exists, then any uncertainty in estimating the actual location of $\mathbf{x}_m = \arg \min f$ should originate from the initialization of the convergence algorithm. In this case, the PDF of independent realizations of the location of the minimum should narrow down with the progression of the algorithm, tending to the Dirac function centered at the argument that minimizes f , i.e.

$$\delta_{m,j}(x) = \begin{cases} +\infty, & x = \{x_j : \mathbf{x} = \mathbf{x}_m = \arg \min f\} \\ 0, & \text{otherwise} \end{cases}, \quad (13)$$

and

$$H_{m,j}(x) := \int_{-\infty}^x \delta_m(s) ds. \quad (14)$$

As P_{X_j} tends to the Dirac function, F_{X_j} tends to the Heaviside step function centered at x_m , hence by Eq. 12 we deduce that

$$E[X_j] \xrightarrow{1} i \rightarrow f_e \int_0^1 H_{m,j}^{-1}(y) dy, \quad (15)$$

and by Eq. 11

$$E[X_j] \xrightarrow{u} b_j - \int_0^1 H_{m,j}(y) dy = ub_j - (ub_j - x_m) \times 1 = x_{m,j} \quad (16)$$

□

4 Programming techniques

A variety of programming techniques were investigated to implement the proposed method into a computer source code. To keep the algorithm lightweight, simple and reduce computational complexity and time, we applied inverse transform sampling by keeping each iteration i the best function evaluations and randomly sampling them. This is equivalent to a kernel function that vanishes over the worst function evaluations and distributing the probability mass to the best-performing ones. Accordingly, similar programming techniques may be investigated in future works within the suggested framework.

Algorithm 2: ITSO-Short Pseudocode

```

1 Initialize:  $\mathbf{x} = \text{rand}(n)$ ,  $\text{opti\_x} = \mathbf{x}$ ,  $\text{opti\_f} = f(\text{opti\_x})$ ;
2 for  $i = 1 : f_e$  do
3     for  $j = 1 : n$  do
4          $\text{indsBEST} = \text{sortperm}(\text{opti\_evals})[1 : \alpha]$ ;
5          $\text{inp\_x} = \mathbf{x\_evals}[\text{indsBEST}, j]$ ;
6          $\text{inp\_y} = \text{opti\_evals}[\text{indsBEST}]$ ;
7          $rr = \min\{\text{inp\_x}\}$ 
8              $+\text{rand}(0, 1)(\max\{\text{inp\_x}\} - \min\{\text{inp\_x}\})$ ;
9          $\mathbf{x}[j] = rr$ ;
10         $f_i = f(\mathbf{x})$ ;
11        if  $f_i \leq \text{opti\_f}$  then
12             $\text{opti\_f} = f_i$ ;
13             $\text{opti\_x} = \mathbf{x}$ ;
14        else
15             $\mathbf{x} = \text{opti\_x}$ ;
16        end
17    end
18 end
19 return  $\text{opti\_f}$ ,  $\text{opti\_x}$ 
    
```

Algorithm 2 represents a simplified version of the supplementary code in Appendix 1, which may be easily programmed. The variable **opti_evals** is a dynamic vector, containing all values of the objective function returned during the optimization history, until step i , and **x_evals** is an $i \times n$ matrix, containing all the design vectors $\mathbf{x}_{1:i}$, corresponding to **opti_evals**. The integer α is a parameter regarding how many instances of the optimization history are kept to sample among them randomly; for example, if $f_e = 10^4$, we may select $\alpha = 10^2$. The choice of α may affect the results, as it regards the selection of the number of best solutions kept, till the current execution step of the algorithm. For the Numerical Experiments we use $\alpha = i$ for the iterations less or equal to the $\frac{1}{3}$ of the total function evaluations f_e . After this point (i.e. $i > \frac{f_e}{3}$), we use $\alpha = \lfloor \frac{f_e}{3} \times (1 - \frac{i}{f_e}) \rfloor + 2$, in order to shrink the search hypercube, as i increases, and more information is available for the optimal values of f . In this context, α is progressively reduced with increasing i , and also rounded to the smallest integer($\lfloor \cdot \rfloor$). For the case $i = f_e$, α takes the value of 2, corresponding the dichotomy of the last two points. This is implemented in line 32 of the file *ITSO-short.jl* in the supplementary computer code as per Appendix 1.

In this variation of the code, the Inverse Transform Sampling is approximately implemented in line 7, by calculating the random variable rr , among the extrema of the vector **inp_x**, corresponding to the range where: for the j^{th} dimension of all $\mathbf{x}_{1:i}$, the α best function values were returned by the *black-box* function f . The sought solution is the vector **opti_x**, and the minimum attained a value of the objective function opti_f .

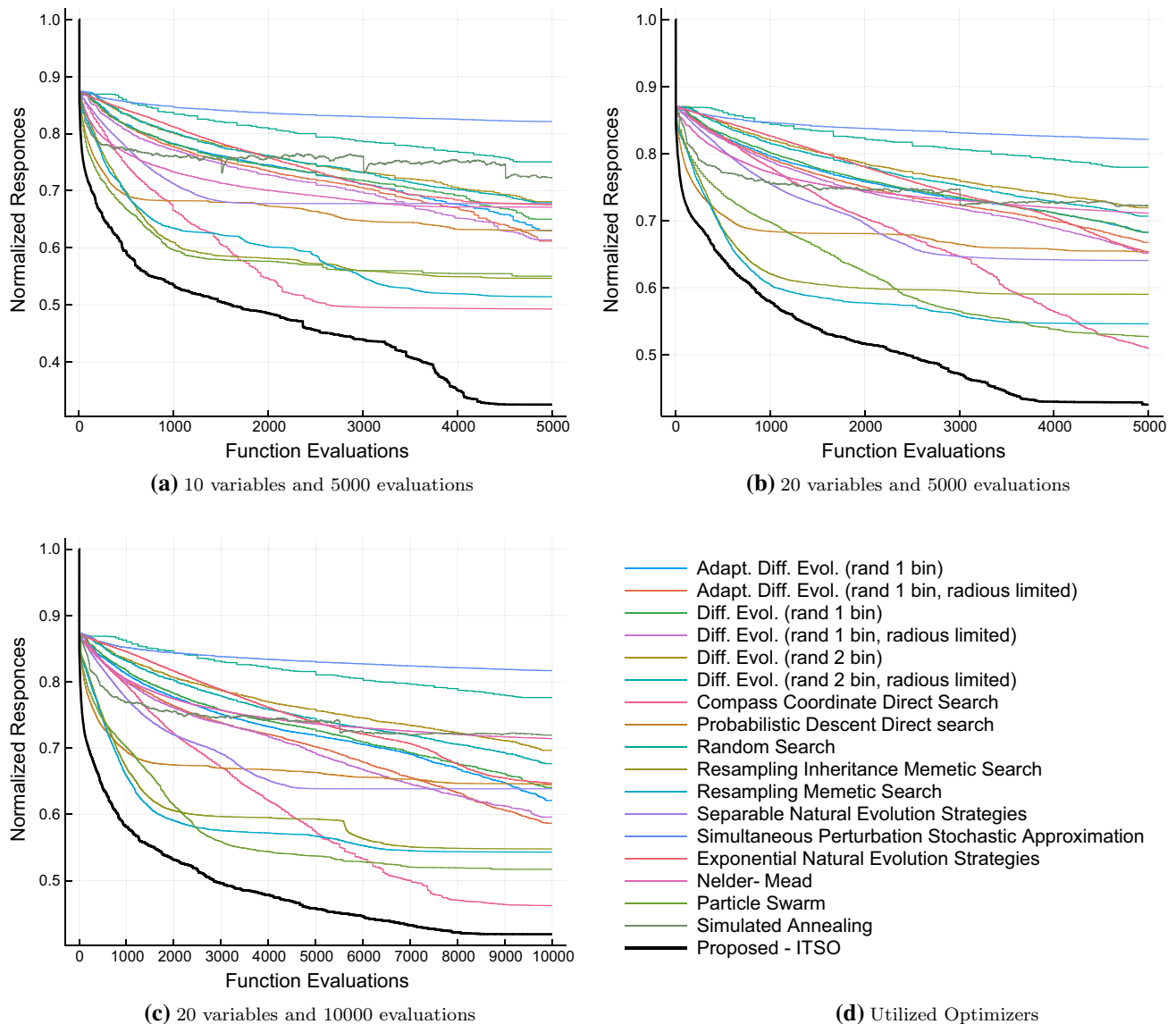


Fig. 3 Average optimization history h (Eq. 19, for all 13 Functions, repeated 10 times)

5 Numerical experiments

In this section, we test the ITSO algorithm, for various objective functions as per Appendix 2, in comparison with two well known packages for Optimization, utilizing a variety of methods. Particularly, we present the results obtained by running the ITSO algorithm, as well as Adaptive Differential Evolution (rand 1 bin), Differential Evolution (rand 1 bin), and Differential Evolution (rand 2 bin) with and without radius limited, Compass Coordinate Direct Search, Probabilistic Descent Direct search, Random Search, Resampling Inheritance Memetic Search, Resampling Memetic Search, Separable Natural Evolution Strategies, Simultaneous Perturbation Stochastic Approximation, and Exponential Natural Evolution Strategies from

the Julia Package BlackBoxOptim.jl (Feldt 2013-2018), and Nelder-Mead, Particle Swarm, and Simulated Annealing from Optim.jl (Mogensen and Riseth 2018). To demonstrate the performance of each optimizer in attaining the minimum, we firstly run the algorithm $r = 10$ times, obtain $f_k(\mathbf{x}_i)$ for $k = \{1, 2, \dots, r\}$ and all iterations $i = \{1, 2, \dots, f_e\}$, and average the results

$$\hat{f}(\mathbf{x}_i) = \frac{\sum_{k=1}^r f_k(\mathbf{x}_i)}{r}. \quad (17)$$

Hence we obtain robust results for the evaluation, and as the values of the various functions f of Appendix 2 vary significantly, we seek to normalize the results in order to compare the historical values within a common domain; see e.g. Hansen et al. 2021, and Karaboga et al. 2004. It is important to underline that this procedure regards solely

Table 1 Average minimum values of h attained by each optimizer

	$n = 10$ $f_e = 5000$	$n = 20$ $f_e = 5000$	$n = 20$ $f_e = 10000$
Adapt. diff. evol. (rand 1 bin)	0.00991	0.02210	0.00852
Adapt. diff. evol. (rand 1 bin, radius limited)	0.00734	0.01762	0.00482
Diff. evol. (rand 1 bin)	0.01348	0.02196	0.01161
Diff. evol. (rand 1 bin, radius limited)	0.00760	0.01377	0.00564
Diff. evol. (rand 2 bin)	0.02140	0.03712	0.02687
Diff. evol. (rand 2 bin, radius limited)	0.02072	0.03110	0.02008
Compass coordinate direct search	0.00084	0.00118	0.00045
Probabilistic descent direct search	0.00984	0.01424	0.01239
Random search	0.05678	0.08307	0.07936
Resampling inheritance memetic search	0.00238	0.00513	0.00243
Resampling memetic search	0.00129	0.00236	0.00223
Separable natural evolution strategies	0.02038	0.01165	0.01127
Simultaneous perturbation stochastic approximation	0.13967	0.13998	0.13227
Exponential natural evolution strategies	0.01972	0.01415	0.01286
Nelder-mead	0.01858	0.03307	0.03452
Particle swarm	0.00254	0.00166	0.00136
Simulated annealing	0.03825	0.03812	0.03721
Proposed-ITSO	0.00001	0.00020	0.00017

the visualisation of the results into a common scale for *all* functions, as the final optima reached by *each* function f are not affected by the suggested operation. Accordingly, we normalize the vector of obtained function evaluations $\mathbf{v} = \{\hat{f}(\mathbf{x}_1), \hat{f}(\mathbf{x}_2), \dots, \hat{f}(\mathbf{x}_{f_e})\}$ in the domain $[0, 1]$ through

$$\hat{f}_n(\mathbf{x}_i) = \frac{\hat{f}(\mathbf{x}_i) - \min \mathbf{v}}{\max \mathbf{v} - \min \mathbf{v}}, \quad (18)$$

and finally use the optimization history h

$$h(\mathbf{x}_i) = \left(\frac{\sum_{l=1}^m \hat{f}_n(\mathbf{x}_i)}{m} \right)^{\frac{1}{10}}, \quad (19)$$

where m indicates the number of *black-box* functions used for the evaluation (Appendix 2). Equation 19 was selected as a performance metric, in order to obtain a clear representation of the various optimizers utilized, as powers smaller than one (in our case $\frac{1}{10}$), have the property to magnify the attained values at the final steps of the optimization history. In Fig. 3 and Table 1, the numerical experiments for $m = 13$ functions are presented. Each line corresponds to the normalized average optimization history h (Eq. 19, for all functions, which were repeated 10 times). We may see a prevalence of the proposed framework, in terms of convergence performance, as expected by the theoretical investigation. The numerical experiments for

the comparison with other optimizers, may be reproduced by running the file “__run.jl”, provided in the supplementary Computer Code (Appendix 1).

6 Discussion and conclusions

In this work, a novel approach was presented for the well-known problem of finding the argument that minimizes a *black-box*, function or system. A vast volume of approximation algorithms has been proposed, mainly heuristic, such as genetic, evolutionary, particle swarm, and variations. However, they stem from *nature-inspired* procedures, and hence their converge is investigated a-posteriori. Despite their efficiency, researchers are often questioning their rigorousness, in terms of mathematical formulation, while the complexity of embodying such algorithms to larger computer codes is often complex and not straightforward. To the contrary, the proposed algorithm initiates its formulation from well established probabilistic definitions and theorems, and its implementation demands a few lines of computer code. Furthermore, the convergence properties were found stable, as presented in the corresponding theoretical proof, and the numerical experiments validate the theoretical results.

Appendix 1: Programming code

The corresponding computer code is available at GitHub <https://github.com/nbakas/ITSO.jl>. The examples of Figure 3 may be reproduced by running `__run.jl`. The short version of the Algorithm 1 is available in Julia (Bezanson et al. 2017) Language, and particularly in the file *ITSO-short.jl*, along with Octave (Contributors 2020), in the file *ITSOshort.m*, and Python (Contributors 2020), in the file *ITSO-short.py*. The implementation of the framework is integrated in a few lines of computer code, which can be easily adapted for case specific applications with high efficiency.

Appendix 2: Black-Box functions

The following functions were used for the numerical experiments. Equations 20, 21 (Elliptic, Cigar), were utilized from Liang et al. 2013, Cigtab (Eq. 22), Griewank (Eq. 23) from Au and Leung 2012, Quartic (Eq. 24) from Jamil and Yang 2013, Schwefel (Eq. 25), Rastrigin (Eq. 26), Sphere (Eq. 27), and Ellipsoid (Eq. 28) from Finck et al. 2010; Feldt 2013-2018, and Alpine (Eq. 29) from Hussain et al. 2017. Equations 30, 31, 32, were developed by the authors. The code implementation for the selected equations appears in file *functions_opti.jl* in the supplementary computer code.

The exact variation used in this work is as follows. We have adopted the notation presented in the Nomenclature section, where i denotes the optimization history step, and j the dimension of the design variable x_{ij} .

$$f_{\text{elliptic}}(\mathbf{x}_i) = \sum_{j=1}^n c_j \left(x_{ij} + \frac{3}{2}\right)^2, \text{ where} \quad (20)$$

$$\mathbf{c} = 10^3 \left\{0, \frac{1}{n-1}, \dots, 1\right\}.$$

$$f_{\text{cigar}}(\mathbf{x}_i) = x_1^2 + \sum_{j=2}^n |x_{ij}|. \quad (21)$$

$$f_{\text{cigtab}}(\mathbf{x}_i) = x_1^2 + \sum_{j=2}^{n-1} |x_{ij}| + x_n^2. \quad (22)$$

$$f_{\text{griewank}}(\mathbf{x}_i) = 1 + \frac{1}{4000} \sum_{j=1}^n x_{ij}^2 - \prod_{j=1}^n \cos\left(\frac{x_{ij}}{\sqrt{j}}\right). \quad (23)$$

$$f_{\text{quartic}}(\mathbf{x}_i) = \sum_{j=1}^n j(x_{ij} - 2)^4. \quad (24)$$

$$f_{\text{schwefel}}(\mathbf{x}_i) = \sum_{j=1}^n c_j^2, \text{ where} \quad (25)$$

$$c_j = \sum_{k=1}^j (x_{ik} - 9).$$

$$f_{\text{rastrigin}}(\mathbf{x}_i) = 10n + \sum_{j=1}^n \left(x_{ij} + \frac{7}{10}\right)^2 - 10 \sum_{j=1}^n \cos\left(2\pi\left(x_{ij} + \frac{7}{10}\right)^2\right). \quad (26)$$

$$f_{\text{sphere}}(\mathbf{x}_i) = \sum_{j=1}^n \left(x_{ij} - \frac{13}{10}\right)^2. \quad (27)$$

$$f_{\text{ellipsoid}}(\mathbf{x}_i) = \sum_{j=1}^n (x_{ij} - \sqrt{2})^2. \quad (28)$$

$$f_{\text{alpine}}(\mathbf{x}_i) = \sum_{j=1}^n |x_{ij} \sin x_{ij} + \frac{1}{10} x_{ij}|. \quad (29)$$

$$f_{x_j}(\mathbf{x}_i) = \sum_{j=1}^n \left(x_{ij} - j - \frac{21}{10}\right)^2. \quad (30)$$

$$f_{x_5}(\mathbf{x}_i) = \sum_{j=1}^n (x_{ij} - 5)^2 - 5. \quad (31)$$

$$f_{\sin_x}(\mathbf{x}_i) = \sum_{j=1}^n \left(\sin\left(x_{ij} + \frac{7}{10}\right) + \frac{\left(x_{ij} + \frac{7}{10}\right)^2}{100}\right). \quad (32)$$

Acknowledgements The Authors would like to acknowledge the insightful comments and suggestions of two anonymous Reviewers and the Editor, which significantly helped them to improve the quality of the presented work.

Funding The contribution of Andreas Langousis has been conducted within the project PerManeNt, which has been co-financed by the European Regional Development Fund of the European Union and Greek National Funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (project code: T2EDK-04177).

Code availability Available as per Appendix 1.

Declarations

Conflicts of interest No.

References

Au CK, Leung HF (2012) Eigenspace sampling in the mirrored variant of $(1, \lambda)$ -cma-es. In: 2012 IEEE Congress on Evolutionary Computation, IEEE, pp 1–8

- Audet C, Hare W (2017) *Derivative-free and blackbox optimization*. Springer, New York
- Audet C, Kokkolaras M (2016) Blackbox and derivative-free optimization: theory, algorithms and applications. <https://doi.org/10.1007/s11081-016-9307-4>
- Bezanson J, Edelman A, Karpinski S, Shah VB (2017) Julia: a fresh approach to numerical computing. *SIAM Rev* 59(1):65–98
- Bottou L, Curtis FE, Nocedal J (2018) Optimization methods for large-scale machine learning. *Siam Rev* 60(2):223–311
- Bull AD (2011) Convergence rates of efficient global optimization algorithms. *J Mach Learn Res* 12:2879–2904
- Chang BC, Ratnaweera A, Halgamuge SK, Watson HC (2004) Particle swarm optimisation for protein motif discovery. *Genet Programm Evol Mach* 5(2):203–214
- Clayton AD, Manson JA, Taylor CJ, Chamberlain TW, Taylor BA, Clemens G, Bourne RA (2019) Algorithms for the self-optimisation of chemical reactions. *React Chem Eng* 4(9):1545–1554
- Clerc M, Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans Evol Comput* 6(1):58–73
- Contributors (2020) Python 3.8.2. <https://www.python.org/>
- Contributors (2020) Gnu octave. <http://hg.savannah.gnu.org/hgweb/octave/file/tip/doc/interpreter/contributors.in>
- Cui H, Guo P, Li M, Guo S, Zhang F (2019) A multi-risk assessment framework for agricultural land use optimization. *Stoch Environ Res Risk Assess* 33(2):563–579. <https://doi.org/10.1007/s00477-018-1610-5>
- De S, Dey S, Bhattacharyya S (eds) (2020) *Recent advances in hybrid metaheuristics for data clustering*. ISBN: 978-1-119-55159-1, Wiley. <https://www.wiley.com/en-us/Recent+Advances+in+Hybrid+Metaheuristics+for+Data+Clustering-p-9781119551591>
- Degasperi A, Fey D, Kholodenko BN (2017) Performance of objective functions and optimisation procedures for parameter estimation in system biology models. *NPJ Syst Biol Appl* 3(1):1–9
- Doerr B (2020a) Probabilistic tools for the analysis of randomized optimization heuristics. In: Doerr B, Neumann F (eds) *Theory of evolutionary computation*. Springer, New York, pp 1–87. https://doi.org/10.1007/978-3-030-29414-4_1
- Doerr C (2020b) Complexity theory for discrete black-box optimization heuristics. In: Doerr B, Neumann F (eds) *Theory of evolutionary computation*. Springer, New York, pp 133–212. https://doi.org/10.1007/978-3-030-29414-4_3
- Feldt R (2013-2018) Blackboxoptim.jl. <https://github.com/robertfeldt/BlackBoxOptim.jl>
- Finck S, Hansen N, Ros R, Auger A (2010) Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Tech. rep, Penn State College of Information Sciences and Technology (<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.232.650&rep=rep1&type=pdf>)
- Gilli M, Schumann E (2012) Heuristic optimisation in financial modelling. *Ann Oper Res* 193(1):129–158
- Hansen N, Auger A, Ros R, Mersmann O, Tušar T, Brockhoff D (2021) Coco: a platform for comparing continuous optimizers in a black-box setting. *Optim Methods Softw* 36(1):114–144
- Hibbert DB (1993) Genetic algorithms in chemistry. *Chemom Intell Lab Syst* 19(3):277–293
- Holden N, Freitas AA (2005) A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In: *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, IEEE, pp 100–107
- Hussain K, Salleh MNM, Cheng S, Naseem R (2017) Common benchmark functions for metaheuristic evaluation: A review. *JOIV Int J Inform Vis* 1(4–2):218–223
- Hutson M (2018) AI researchers allege that machine learning is alchemy. *Science* 360:961. DOI: 10.1126/science.aau0577
- Jamil M, Yang XS (2013) A literature survey of benchmark functions for global optimization problems. arXiv preprint arXiv:13084008
- Kapoutsis AC, Chatzichristofis SA, Doitsidis L, de Sousa JB, Pinto J, Braga J, Kosmatopoulos EB (2016) Real-time adaptive multi-robot exploration with application to underwater map construction. *Auton Robots* 40(6):987–1015. <https://doi.org/10.1007/s10514-015-9510-8>
- Kapoutsis AC, Chatzichristofis SA, Kosmatopoulos EB (2019) A distributed, plug-n-play algorithm for multi-robot applications with a priori non-computable objective functions. *Int J Robot Res*. <https://doi.org/10.1177/0278364919845054>
- Karaboga N, Kalinli A, Karaboga D (2004) Designing digital iir filters using ant colony optimisation algorithm. *Eng Appl Artif Intell* 17(3):301–309
- Lagaros ND, Papadrakakis M, Bakas NP (2006) Automatic minimization of the rigidity eccentricity of 3D reinforced concrete buildings. *J Earthq Eng* 10(4):533–564. <https://doi.org/10.1080/13632460609350609>
- Lagaros ND, Bakas N, Papadrakakis M (2009) Optimum design approaches for improving the seismic performance of 3D RC buildings. *J Earthq Eng* 13(3):345–363. <https://doi.org/10.1080/13632460802598594>
- Liang J, Qu B, Suganthan P, Hernández-Díaz AG (2013) Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201212(34):281–295
- Lin L, Cao L, Wang J, Zhang C (2004) The applications of genetic algorithms in stock market data mining optimisation. *Manag Inf Syst*. <https://www.witpress.com/elibrary/wit-transactions-on-information-and-communication-technologies/33/14241>
- Moayyeri N, Gharehbaghi S, Plevris V (2019) Cost-based optimum design of reinforced concrete retaining walls considering different methods of bearing capacity computation. *Mathematics* 7(12):1232
- Mogensen PK, Riset AN (2018) Optim: a mathematical optimization package for Julia. *J Open Source Softw* 3(24):615
- Muñoz MA, Smith-Miles KA (2017) Performance analysis of continuous black-box optimization algorithms via footprints in instance space. *Evolut Comput* 25(4):529–554
- Opara KR, Arabas J (2019) Differential evolution: a survey of the theoretical analyses. *Swarm Evolut Comput* 44:546–558
- Papadrakakis M, Lagaros ND, Plevris V (2001) Optimum design of space frames under seismic loading. *Int J Struct Stab Dyn* 1(01):105–123
- Papadrakakis M, Lagaros ND, Plevris V (2005) Design optimization of steel structures considering uncertainties. *Engi Struct* 27(9):1408–1418
- Parker FD (1955) Integrals of inverse functions. *Am Math Mon* 62(6):439. <https://doi.org/10.2307/2307006>
- Plevris V, Papadrakakis M (2011) A hybrid particle swarm-gradient algorithm for global structural optimization. *Comput Aided Civ Infrastruct Eng* 26(1):48–68
- Ponomareva K, Roman D, Date P (2015) An algorithm for moment-matching scenario generation with application to financial portfolio optimisation. *Eur J Oper Res* 240(3):678–687
- Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat Mach Intell* 1(5):206–215. <https://doi.org/10.1038/s42256-019-0048-x>
- Rudolph G (1994) Convergence analysis of canonical genetic algorithms. *IEEE Trans Neural Netw* 5(1):96–101
- Sculley D, Snoek J, Rahimi A, Wiltschko A (2018) Winner's Curse? On Pace, Progress, and Empirical Rigor. ICLR Workshop track

- Siddique N, Adeli H (2017) Nature-inspired chemical reaction optimisation algorithms. *Cogn Comput* 9(4):411–422
- Sra S, Nowozin S, Wright SJ (2012) Optimization for machine learning. Mit Press, Cambridge
- Wang CF, Hu MC, Lee CH, Yu HL (2019a) Optimization of air quality monitoring network based on a spatiotemporal-spectrum manifold analysis. *Stoch Environl Res Risk Assess* 33(10):1835–1849. <https://doi.org/10.1007/s00477-019-01730-x>
- Wang Y, Liu L, Guo P, Zhang C, Zhang F, Guo S (2019b) An inexact irrigation water allocation optimization model under future climate change. *Stoch Environ Res Risk Assess* 33(1):271–285. <https://doi.org/10.1007/s00477-018-1597-y>
- Wu J, Poloczek M, Wilson AG, Frazier P (2017) Bayesian optimization with gradients. *Adv Neural Inf Process Syst* pp 5267–5278. <https://papers.nips.cc/paper/2017/hash/64a08e5f1e6c39faeb90108c430eb120-Abstract.html>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.