ITSO: A novel Inverse Transform Sampling-based Optimization algorithm for stochastic search

Nikolaos P. Bakas,* Vagelis Plevris,† Andreas Langousis,‡ Savvas A. Chatzichristofis§

Abstract

Optimization algorithms appear in the core calculations of numerous Artificial Intelligence (AI) and Machine Learning methods, as well as Engineering and Business applications. Following recent works on the theoretical deficiencies of AI, a rigor context for the optimization problem of a *black-box* objective function is developed. The algorithm stems directly from the theory of probability, instead of a 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 only falling into the proposed framework, either directly or incidentally, any optimization algorithm converges in the fastest possible time. The numerical experiments, verify the theoretical results on the efficacy of the algorithm apropos reaching the optimum, as fast as possible.

Keywords: Stochastic Optimization, Inverse Transform Sampling, Black-box Function, Global Convergence.

1 Introduction

Despite the numerous research works and industrial applications of Artificial Intelligence algorithms, they have been criticized about lacking a solid theoretical background [1]. The empirical results demonstrate impressive performance, however, their theoretical foundation and analysis are often vague [2]. Machine Learning (ML) models, frequently aim at identifying an optimal solution [3, 4], which is computationally hard and attempts to explain the procedure often based on the evaluation of the function's Gradients [5, 6]. Accordingly, the identification of whether a stochastic algorithm will work or not remains an open question for many research and real-world applications [7, 8, 9, 10]. A vast number of optimization algorithms have been developed and applied for the solution of the corresponding problems [11, 12, 13, 14], as well as mathematical proofs regarding their algorithmic convergence [15, 16, 17], however, their basic formulation often stems from nature-inspired procedures [18, 19] and not solid mathematical frameworks. Accordingly, a vast number of research works have been published in order to investigate the performance of black-box algorithms [20, 21, 22].

*Computation-based Science and Technology Research Center, The Cyprus Institute, 20 Konstantinou Kavafi Street, 2121, Aglantzia Nicosia, Cyprus. e-mail: n.bakas@cyi.ac.cy

†Department of Civil Engineering and Energy Technology, OsloMet—Oslo Metropolitan University, Pilestredet 35, Oslo 0166, Norway. e-mail: vageli@oslomet.no

‡Department of Civil Engineering, University of Patras, 265 04 Patras, Greece. e-mail: andlag@alum.mit.edu

§Intelligent Systems Lab & Department of Computer Science, Neapolis University Pafos, 2 Danais Avenue, 8042 Pafos, Cyprus. e-mail: s.chatzichristofis@nup.ac.cy

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, $f : A \to \mathbb{R}^n$. 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. A is assumed a compact subset of the Euclidean space $\mathbb{R}^n$, where n is the number of dimensions of the set A, however, the proposed method applies similarly to discrete and continuous topological spaces. The unknown, black-box function f, returns values for the given input $x_{ij} = \mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{in})$ at each computational discrete time step $i = \{1, 2, \ldots, f_e\}$, 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 by

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

The purpose of this work is to provide a rigor context for the optimization problem of a *black-box* function, by adhering to the 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 Section 2, the proposed Inverse Transform Sampling Optimizer (ITSO) is presented. Additionally, the same Section provides in details the theoretical formulation of the algorithm as well as some programming and implementation techniques. Illustrative examples of the optimization history are also comprised. In Section 3 the theoretical proof of convergence is provided, as well as Lemma 1, deriving that the suggested optimization framework is the fastest possible. The numerical experiments are divided into three groups. Subsection 5 is about the comparison with 13 nonlinear loss functions, 17 optimization methods, for 10 and 20 dimensions of search space, and 5000 and 10000 iterations per dimension. Section 4, briefly presents the programming techniques that were investigated, in order to implement the proposed method into a computer code. Finally, the conclusions are drawn in Section 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 $\arg\min f$,

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

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 Equation 2, such as the Gaussian:

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

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

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

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

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 $1, 2, ..., i$, 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 $X' \sim \mathcal{U}(a, b) \forall k(f_{\hat{i}}) \in (a, b)$.

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:

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

$P_{X_j}(x_{ij})dx$ 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 (CDF) F_{X_j}, can be calculated by:

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

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 pseudo-code 1, the algorithmic implementation of the Inverse Transform Sampling method is demonstrated. The symbols are also noted in the Nomenclature section.

A graphical demonstration of the evolution of the Probability Density, as well as the corresponding Cumulative Distribution Functions, is presented in Figure 1, for $f(x) = (x-5)^2$, which is function with one extreme value and in Figure 2, for $f(x) = sin(x + 0.7) + 0.01 * (x + 0.7)^2$, 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 $f(x)$ attends its lows. Figures 1 and 2, offer an intuitive representation of the procedure for finding the minimum of the 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 Equation 6, and $F_{X_j}(x_{ij})$ by Equation 7, for all x_{ij}.

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

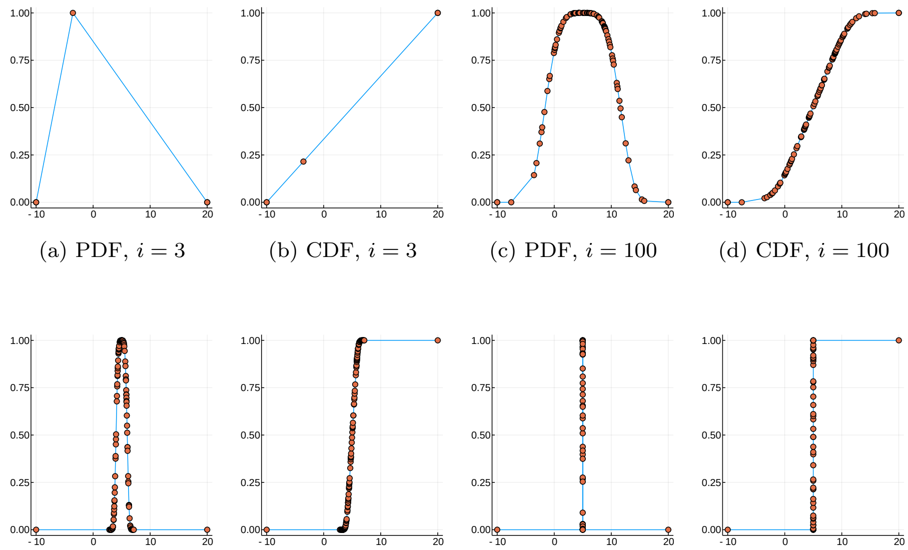
In iteration i, the algorithm will have searched the space $\hat{A} \subset A$, within the limits lb_j, ub_j. By definition, P is the probability density (likelihood) that the optimum occurs in a region $\mathbf{x} \pm \mathbf{dx}$.

Algorithm 1: ITSO-Mathematical Framework

Data: A, f_e, n

Result: $\mathbf{x}^* = \arg\min f,\ f^* = f(\mathbf{x}^*)$

while $i \leq f_e$ **do**
    $j \leftarrow \mathcal{U}\{1, n\}$;
    $r_i \leftarrow \mathcal{U}(0, 1)$;
    SORT $x_{ij}\quad \forall i$;
    $x_{ij} \leftarrow F_j^{-1}(r_i)$;
    $f_i = f(\mathbf{x}_i)$;
    if $f_i \leq f^*$ **then**
        $f^* \leftarrow f_i$;
        $x_j^* \leftarrow x_{ij}$;
    else
        $x_{ij} \leftarrow x_j^*$;
    end
end
return $\mathbf{x}^* = \arg\min f$



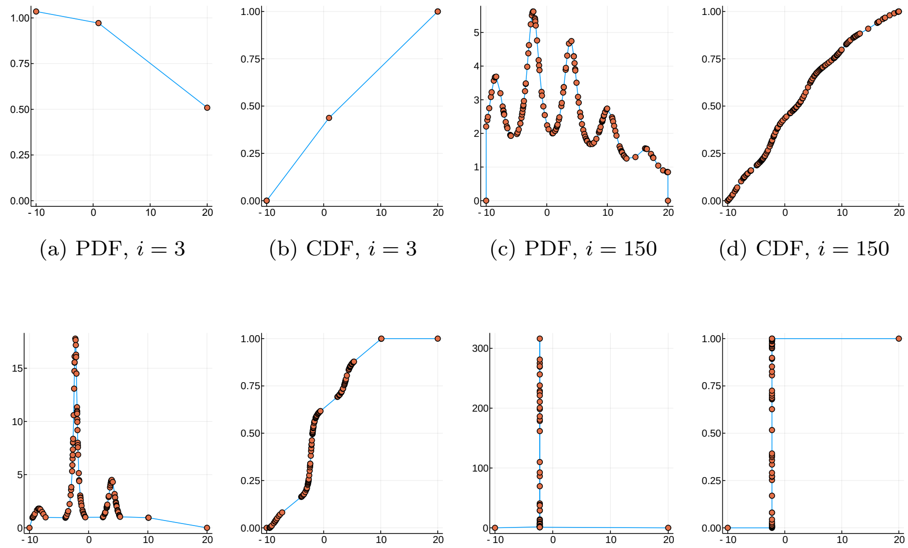
(a) PDF, $i = 3$ (b) CDF, $i = 3$ (c) PDF, $i = 100$ (d) CDF, $i = 100$

(e) PDF, $i = 350$ (f) CDF, $i = 350$ (g) PDF, $i = 500$ (h) CDF, $i = 500$

Figure 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 $x_m = 5$

Hence,

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



(a) PDF, $i = 3$ (b) CDF, $i = 3$ (c) PDF, $i = 150$ (d) CDF, $i = 150$

(e) PDF, $i = 250$ (f) CDF, $i = 250$ (g) PDF, $i = 500$ (h) CDF, $i = 500$

Figure 2: Probability Density and Cumulative Distribution Functions, utilizing ITSO search strategy, for function $f(x) = sin(x + 0.7) + 0.01 * (x + 0.7)^2$. The shape sequentially tends to the Heaviside function, centered at $x_m = -2.24$

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

Theorem 1. *If P_{X_j} tends to Dirac δ_m, then ITSO will converge to x_m*

Proof. For each dimension j, we may write

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

and integrating by parts, we obtain

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

If we apply the theorem of the antiderivative of inverse functions [23], we obtain

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

Hence, for Equations 10 and 11 we deduce that

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

With subscript m denoting that the Dirac function is centered at the argument that minimizes f, i.e.

$$\delta_m(x) = \begin{cases} +\infty, & x = \arg\min f \\ 0, & x \neq \arg\min f \end{cases}, \tag{13}$$

and

$$H_m(x) := \int_{-\infty}^{x} \delta_m(s) ds. \tag{14}$$

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

$$E[X_j] \xrightarrow{i \to f_e} \int_0^1 {H_m}^{-1}(y) dy, \tag{15}$$

and by Equation 11

$$E[X_j] \to ub_j - \int_0^1 H_m(y) dy = ub_j - (ub_j - x_m) * 1 = x_m \tag{16}$$

□

Lemma 1. (ITSO Convergence Speed) *ITSO is the fastest possible optimization framework*

Proof. Any distribution that doesn't tend to Dirac, could be considered as a Dirac plus a positive function of x. In this case, the algorithmic framework would search through a strategy that produces a P' over $\mathbf{x}$, as

$$P' = P^* \pm \delta_m. \tag{17}$$

Hence, with F^* indicating the CDF corresponding to P^*, Equations 15, and 12, result in

$$E[X] \xrightarrow{i \to f_e} \int_0^1 {H_m}^{-1}(y) dy + \int_{lb}^{ub} F^*(x) dx, \tag{18}$$

.

and thus, by Equations 17, and 16, we obtain

$$E[X] \to x_m \pm \epsilon. \tag{19}$$

Hence the algorithm would converge to a point different than x_m

□

4 Programming techniques

A variety of programming techniques were investigated, in order to implement the proposed method into a computer code. To keep the algorithm simple and reduce the computational time, we applied inverse transform sampling by keeping in each iteration i the best function evaluations, and randomly sampling among 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.

The Algorithm 2 represents a simple version of the supplementary code in the appendix, which may easily be 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

Algorithm 2: ITSO-Short Pseudocode

```
1  Initialize: x = rand(n),   opti_x = x,   opti_f = f(opti_x);
2  for i = 1 : f_e do
3      for j = 1 : n do
4          indsBEST = sortperm(opti_evals)[1 : α];
5          inp_x = x_evals[indsBEST, j];
6          inp_y = opti_evals[indsBEST];
7          rr = min{inp_x}
8              +rand(0, 1)(max{inp_x} − min{inp_x});
9          x[j] = rr;
10         f_i = f(x);
11         if f_i ≤ opti_f then
12             opti_f = f_i;
13             opti_x = x;
14         else
15             x = opti_x;
16         end
17     end
18 end
19 return   opti_f,   opti_x
```

$i \times n$ 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 in order to randomly sample among them; for example if $f_e = 10^4$, we may select $\alpha = 10^2$. 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 mimimum attained value of the objective function *opti_f*.

5 Numerical Experiments

In this section, 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 [24], and Nelder-Mead, Particle Swarm, and Simulated Annealing from Optim.jl [25]. 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 $k = \{1, 2, \ldots, r\}$ and all iterations $i = \{1, 2, \ldots, f_e\}$, and average the results

$$\hat{f}(\mathbf{x}_i) = \frac{\sum_{k=1}^{r} f_k(\mathbf{x}_i)}{r}. \tag{20}$$

Then, we normalize the vector of obtained function evaluations $\mathbf{v} = \{\hat{f}(\mathbf{x}_1), \hat{f}(\mathbf{x}_1), \ldots, \hat{f}(\mathbf{x}_{f_e})\}$ in the domain $[0, 1]$ through

$$\hat{f}_n(\mathbf{x}_i) = \frac{\hat{f}(\mathbf{x}_i) - \min \mathbf{v}}{\max \mathbf{v} - \min \mathbf{v}}, \tag{21}$$

and finally use the optimization history h

$$h(\mathbf{x}_i) = \left(\frac{\sum_{l=1}^{m} \hat{f}_n^l(\mathbf{x}_i)}{m} \right)^{\frac{1}{10}}, \tag{22}$$

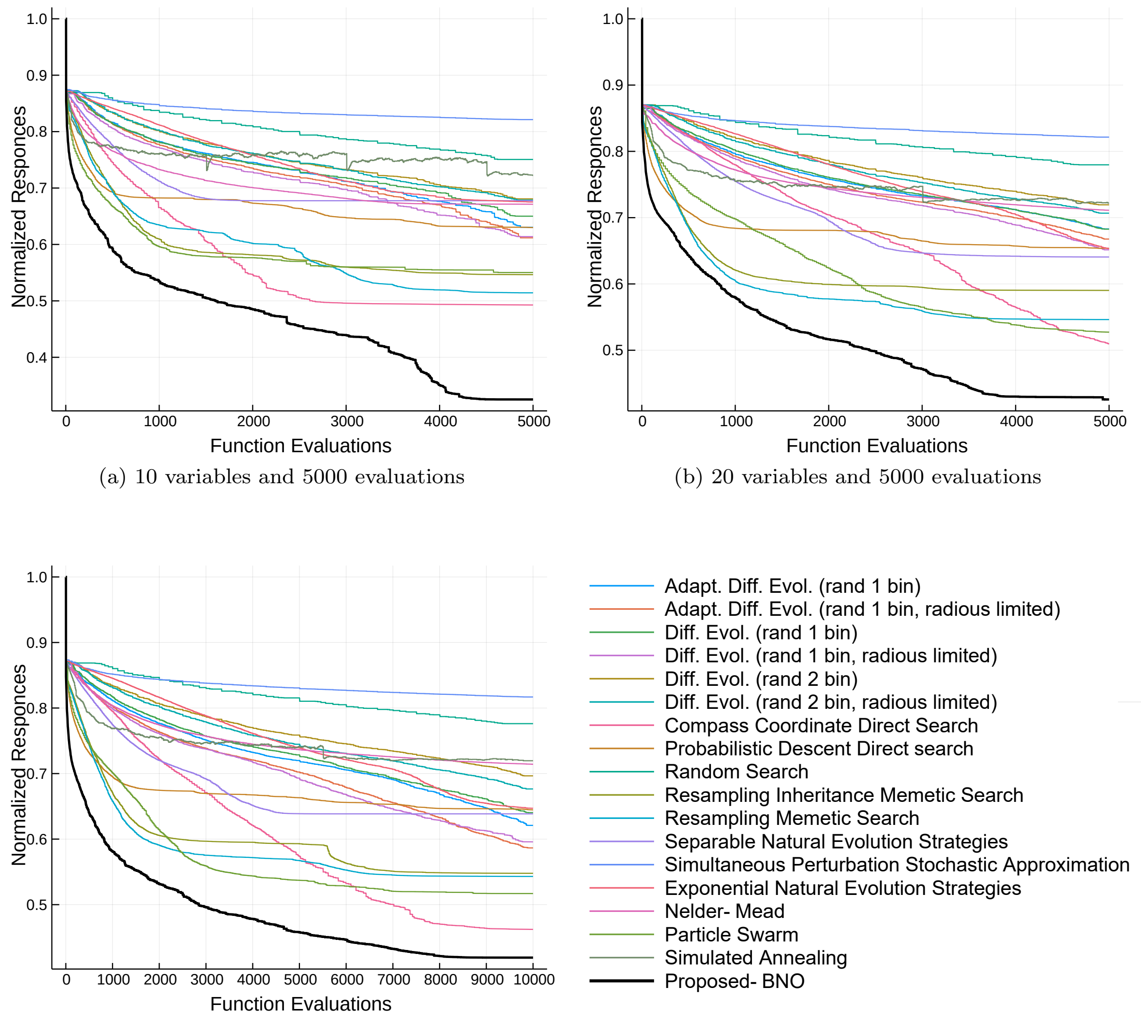
where m indicates the number of *black-box* functions used for the evaluation. Equation 22 was selected as a performance metric, in order to obtain a clear representation of the various optimizers utilized, as powers smaller than one (in our case $\frac{1}{10}$), have the property to magnify the attained values at the final steps of the optimization history. In Figure 3 the numerical experiments for $m = 13$ functions are presented. Each line corresponds to the normalized average optimization history h (Equation 22, for all functions, which were repeated 10 times). We may see a clear 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*.

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 have been proposed, mainly heuristic, such as genetic, evolutionary, particle swarm, as well as their variations. However, they stem from *nature-inspired* procedures, and hence their converge is investigated a-posteriori. Despite their efficiency, they are often deprecated by researchers, due to the lack of rigorous mathematical formulation, as well as complexity of implementation. 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 a proof that the suggested framework attains the best possible solution in the fewest possible iterations. The numerical examples validate the theoretical results and may be reproduced by the provided computer code. We consider the suggested method as a powerful framework which may easily be adopted to the sought solution of any problem involving the minimization of a *black-box* function.

Appendix A Programming Code

The corresponding computer code is available on GitHub `https://github.com/nbakas/ITSO.jl`. The examples of Figure 3 may be reproduced by running *__run.jl*. The sort version of the Algorithm 1 is available in Julia [26] (file *ITSO-short.jl*), Octave [27] (*ITSOshort.m*), and Python [28] (*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.



(a) 10 variables and 5000 evaluations

(b) 20 variables and 5000 evaluations

(c) 20 variables and 10000 evaluations

(d) Utilized Optimizers

Figure 3: Average Optimization history h (Equation 22, for all 13 Functions, repeated 10 times.

Appendix B *Black-Box* Functions

The following functions were used for the numerical experiments. Equations 23, 24 (Elliptic, Cigar), were utilized from [29], Cigtab (Eq. 25), Griewank 26 from [30], Quartic (Eq. 27 from [31], Schwefel (Eq. 28), Rastrigin (Eq. 29), Sphere (Eq. 30), and Ellipsoid (Eq. 31) from [32, 24], and Alpine (Eq. 32) from [33]. Equations 33, 34, 35, 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, where we have adopted the notation presented in the Nomenclature section, where i denotes the step of the optimization history, and j the dimension

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, radious limited)	0.00734	0.01762	0.00482
Diff. Evol. (rand 1 bin)	0.01348	0.02196	0.01161
Diff. Evol. (rand 1 bin, radious limited)	0.00760	0.01377	0.00564
Diff. Evol. (rand 2 bin)	0.02140	0.03712	0.02687
Diff. Evol. (rand 2 bin, radious 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

of the design variable x_{ij}.

$$\begin{gathered} f_{elliptic}(\mathbf{x}_i) = \sum_{j=1}^{n} c_j (x_{ij} + \frac{3}{2})^2, \text{where} \\ \mathbf{c} = 10^3 \{0, \frac{1}{n-1}, \ldots, 1\}. \end{gathered} \tag{23}$$

$$f_{cigar}(\mathbf{x}_i) = x_1^2 + \sum_{j=2}^{n} |x_{ij}|. \tag{24}$$

$$f_{cigtab}(\mathbf{x}_i) = x_1^2 + \sum_{j=2}^{n-1} |x_{ij}| + x_n^2. \tag{25}$$

$$f_{griewank}(\mathbf{x}_i) = 1 + \frac{1}{4000} \sum_{j=1}^{n} x_{ij}^2 - \prod_{j=1}^{n} \cos\left(\frac{x_{ij}}{\sqrt{j}}\right). \tag{26}$$

$$f_{quartic}(\mathbf{x}_i) = \sum_{j=1}^{n} j(x_{ij} - 2)^4. \tag{27}$$

$$\begin{aligned} f_{schwefel}(\mathbf{x}_i) &= \sum_{j=1}^{n} c_j^2, \text{where} \\ c_j &= \sum_{k=1}^{j} (x_{ik} - 9). \end{aligned} \tag{28}$$

$$\begin{aligned} f_{rastrigin}(\mathbf{x}_i) = 10n + \sum_{j=1}^{n} (x_{ij} + \frac{7}{10})^2 \\ -10 \sum_{j=1}^{n} \cos(2\pi(x_{ij} + \frac{7}{10})^2). \end{aligned} \tag{29}$$

$$f_{sphere}(\mathbf{x}_i) = \sum_{j=1}^{n} (x_{ij} - \frac{13}{10})^2. \tag{30}$$

$$f_{elipsoid}(\mathbf{x}_i) = \sum_{j=1}^{n} (x_{ij} - \sqrt{2})^2. \tag{31}$$

$$f_{alpine}(\mathbf{x}_i) = \sum_{j=1}^{n} |x_{ij} \sin x_{ij} + \frac{1}{10} x_{ij}|. \tag{32}$$

$$f_{x\_j}(\mathbf{x}_i) = \sum_{j=1}^{n} (x_{ij} - j - \frac{21}{10})^2. \tag{33}$$

$$f_{x\_5}(\mathbf{x}_i) = \sum_{j=1}^{n} (x_{ij} - 5)^2 - 5. \tag{34}$$

$$f_{sin\_x}(\mathbf{x}_i) = \sum_{j=1}^{n} (\sin(x_{ij} + \frac{7}{10}) + \frac{(x_{ij} + \frac{7}{10})^2}{100}). \tag{35}$$

References

[1] M. Hutson, "AI researchers allege that machine learning is alchemy," *Science*, may 2018. [Online]. Available: http://www.sciencemag.org/news/2018/05/ai-researchers-allege-machine-learning-alchemy

[2] D. Sculley, J. Snoek, A. Rahimi, and A. Wiltschko, "Winner's Curse? On Pace, Progress, and Empirical Rigor," *ICLR Workshop track*, 2018.

[3] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for machine learning.* Mit Press, 2012.

[4] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[5] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019. [Online]. Available: https://doi.org/10.1038/s42256-019-0048-x

[6] J. Wu, M. Poloczek, A. G. Wilson, and P. Frazier, “Bayesian optimization with gradients,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5267–5278.

[7] B. Doerr, “Probabilistic tools for the analysis of randomized optimization heuristics,” in *Theory of Evolutionary Computation*. Springer, 2020, pp. 1–87.

[8] C. Doerr, “Complexity theory for discrete black-box optimization heuristics,” in *Theory of Evolutionary Computation*. Springer, 2020, pp. 133–212.

[9] M. Papadrakakis, N. D. Lagaros, and V. Plevris, “Design optimization of steel structures considering uncertainties,” *Engineering Structures*, vol. 27, no. 9, pp. 1408–1418, 2005.

[10] ——, “Optimum design of space frames under seismic loading,” *International Journal of Structural Stability and Dynamics*, vol. 1, no. 01, pp. 105–123, 2001.

[11] N. Moayyeri, S. Gharehbaghi, and V. Plevris, “Cost-based optimum design of reinforced concrete retaining walls considering different methods of bearing capacity computation,” *Mathematics*, vol. 7, no. 12, p. 1232, 2019.

[12] V. Plevris and M. Papadrakakis, “A hybrid particle swarm—gradient algorithm for global structural optimization,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 26, no. 1, pp. 48–68, 2011.

[13] N. D. Lagaros, N. Bakas, and M. Papadrakakis, “Optimum Design Approaches for Improving the Seismic Performance of 3D RC Buildings,” *Journal of Earthquake Engineering*, vol. 13, no. 3, pp. 345–363, mar 2009. [Online]. Available: http://www.tandfonline.com/doi/full/10.1080/13632460802598594

[14] N. D. Lagaros, M. Papadrakakis, and N. P. Bakas, “Automatic minimization of the rigidity eccentricity of 3D reinforced concrete buildings,” *Journal of Earthquake Engineering*, vol. 10, no. 4, pp. 533–564, jul 2006. [Online]. Available: http://www.tandfonline.com/doi/full/10.1080/13632460609350609

[15] A. D. Bull, “Convergence rates of efficient global optimization algorithms,” *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2879–2904, 2011.

[16] G. Rudolph, “Convergence analysis of canonical genetic algorithms,” *IEEE transactions on neural networks*, vol. 5, no. 1, pp. 96–101, 1994.

[17] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[18] K. R. Opara and J. Arabas, “Differential evolution: A survey of theoretical analyses,” *Swarm and evolutionary computation*, vol. 44, pp. 546–558, 2019.

[19] N. Razmjooy, V. V. Estrela, H. J. Loschi, and W. Fanfan, "A comprehensive survey of new meta-heuristic algorithms," *Recent Advances in Hybrid Metaheuristics for Data Clustering, Wiley Publishing*, 2019.

[20] M. A. Muñoz and K. A. Smith-Miles, "Performance analysis of continuous black-box optimization algorithms via footprints in instance space," *Evolutionary computation*, vol. 25, no. 4, pp. 529–554, 2017.

[21] C. Audet and W. Hare, *Derivative-free and blackbox optimization.* Springer, 2017.

[22] C. Audet and M. Kokkolaras, "Blackbox and derivative-free optimization: theory, algorithms and applications," 2016.

[23] F. D. Parker, "Integrals of Inverse Functions," *The American Mathematical Monthly*, vol. 62, no. 6, p. 439, jun 1955. [Online]. Available: http://www.jstor.org/stable/2307006?origin=crossref

[24] R. Feldt, "Blackboxoptim.jl," 2013-2018. [Online]. Available: https://github.com/robertfeldt/BlackBoxOptim.jl

[25] J. Myles White, T. Holy, O. Contributors (2012), P. Kofod Mogensen, J. Myles White, T. Holy, P. Contributors (2016), Other. Kofod Mogensen, A. Nilsen Riseth, J. Myles White, T. Holy, and O. Contributors (2017), "Optim.jl," 2012,2016,2017. [Online]. Available: https://github.com/JuliaNLSolvers/Optim.jl

[26] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017.

[27] Contributors, "Gnu octave," http://hg.savannah.gnu.org/hgweb/octave/file/tip/doc/interpreter/contributors.in, 28 Feb 2020.

[28] ——, "Python 3.8.2," https://www.python.org/, 2020.

[29] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz, "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*, vol. 201212, no. 34, pp. 281–295, 2013.

[30] C.-K. Au and H.-F. Leung, "Eigenspace sampling in the mirrored variant of (1, λ)-cma-es," in *2012 IEEE Congress on Evolutionary Computation.* IEEE, 2012, pp. 1–8.

[31] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," *arXiv preprint arXiv:1308.4008*, 2013.

[32] S. Finck, N. Hansen, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions," Citeseer, Tech. Rep., 2010.

[33] K. Hussain, M. N. M. Salleh, S. Cheng, and R. Naseem, "Common benchmark functions for metaheuristic evaluation: A review," *JOIV: International Journal on Informatics Visualization*, vol. 1, no. 4-2, pp. 218–223, 2017.