# Experimental Improvements of Global Optimization Algorithms for Lipschitz Functions

Perceval Beja-Battais, Gaëtan Serré, Sophia Chirrane

# Experimental Improvements of Global Optimization Algorithms for Lipschitz Functions

Perceval Beja-Battais, Gaëtan Serré and Sophia Chirrane

Centre Borelli, ENS Paris-Saclay
CentraleSupélec
{perceval.beja-battais, gaetan.serre}@ens-paris-saclay.fr
sophia.chirrane@student-cs.fr

**Abstract**

In this paper, we define an experimental context in which we tested the performances of LIPO and AdaLIPO, two global optimization algorithms for Lipschitz functions, introduced in [10]. We provide experimental proofs of the efficiency of those algorithms, led numerical statistical analysis of our results, and suggested two intuitive improvements from the vanilla version of the algorithms, referred as LIPO-E and AdaLIPO-E. Within our test bench, these improvements allow the algorithms to converge significantly faster and whenever they struggle to find a better maximizer. Finally, we defined the scope of application of LIPO and AdaLIPO. We show that they are very prone to the curse of dimensionality and tend quickly to Pure Random Search when the dimension increases. We provide source code for LIPO, AdaLIPO, and our enhanced versions here[1].

**Keywords:** global optimization, Lipschitz applications, statistical analysis, convergence rate bounds, numerical analysis

## 1 General context

Global optimization refers to any method whose aim is to find the global maxima of a function over a set. This branch of applied mathematics is largely studied since numerous real life problems require it. One of the major fields where global optimization is widely used is shape optimization (Figure 1). Taking the example of optimizing the shape of an aircraft wing or a car, we can define an objective function by weighting several properties we desire (robustness, weight, wind penetration, etc.). The so defined objective function is highly complex and often requires to launch a whole simulation to be evaluated.

All authors have contributed equally to this work.

[1] https://github.com/gaetanserre/LIPO-E

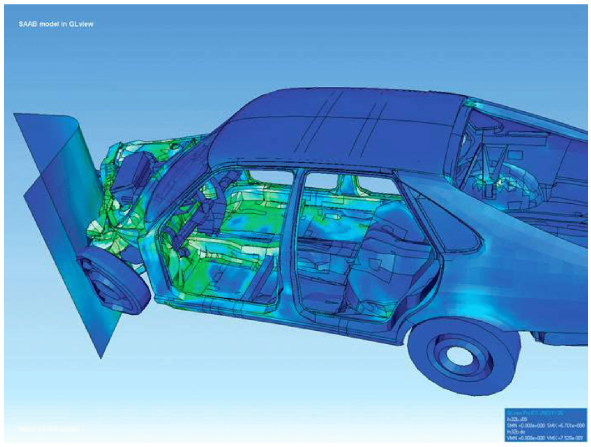


Figure 1: Global optimization for shape optimization: some finite element methods use global optimization algorithms to compute the updates [3].

Another major real life application of global optimization is hyperparameter tuning (Figure 2). Often done using cross validation in machine learning, this task requires the evaluation of very complex functions living in big dimensional space (according to the chosen models), making the hyperparameter selection very difficult. Global optimization methods can provide more clever ways to explore the space of hyperparameter than Grid Search or Pure Random Search.

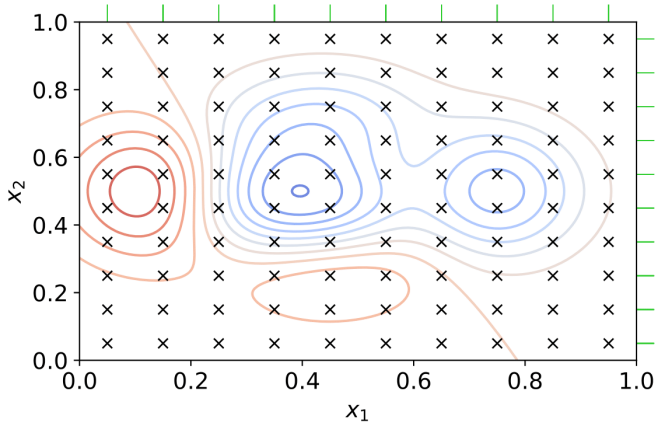



Figure 2: Global optimization for hyperparameters tuning. Traditional methods for hyperparameters tuning consist in a grid search on which the function is evaluated at every cross from the above graph. The resulting values of the objective function are represented with colored level lines.

Global optimization has countless other applications such as protein structure prediction, TSP, and object packing. These are challenging real-life applications that would benefit from a quick resolution. Most of the time, we aim to optimize an unknown function, living in a continuous space. We cannot assume strong properties like differentiability or convexity, which considerably reduces the number of usable algorithms. Furthermore, the evaluation of the studied function is often expensive: the less the optimizing method evaluates the function, the better.

The LIPO method and its adaptative variant AdaLIPO [10] aims at addressing all these raised issues. This global optimization method address the problem of optimizing a function $f : \mathcal{X} \to \mathbb{R}$, with $\mathcal{X} \subset \mathbb{R}^d$ a compact and convex set with non-empty interior such that $f$ is $k$-Lipschitz, i.e.

$$\exists k \geq 0, \quad |f(x) - f(x')| \leq k\|x - x'\|_2 \quad \forall (x, x') \in \mathcal{X}^2.$$

The Lipschitz assumption controls how fast variations of a function can change. This regularity constraint is enforced by the Lipschitz constant $k$. The bigger $k$ is, the less regular the function. As $k$ can be potentially huge this assumption offers a not too restrictive scope of application and thus can be posed in many real world problems.

In this paper we aim at providing a complete experimental analysis of LIPO and AdaLIPO. First, in Section 3 we proposed a reference implementation of these algorithms, fully reproducible and explained. Then, in Section 4 we designed experiments to evaluate the theoretical properties of the method introduced in the paper and compared LIPO with AdaLIPO and other methods. We, then, propose some improvements of both LIPO and AdaLIPO based on empirical observations in Section 5. Finally, in Section 6, we defined the application scope of LIPO.

**Notations.** We define some notations that we will use through this entire article. Most are taken directly from [10]. Let $\mathcal{X} \subset \mathbb{R}^d$ a compact and convex set with non-empty interior. Let $x^* \in \arg\max_{x \in \mathcal{X}} f(x)$ for $f : \mathcal{X} \to \mathbb{R}$. Let $B(x, r) = \{x' \in \mathbb{R}^d : \|x' - x\|_2 \leq r\}$ the ball centered in $x$ of radius $r \geq 0$. For any bounded set $\mathcal{X} \subset \mathbb{R}^d$, let $\text{rad}(x) = \sup\{r > 0 : \exists x \text{ s.t. } B(x, r) \subset X\}$ its radius and $\text{diam}(x) = \sup_{x,x' \in X} \|x - x'\|_2$ its diameter. We also define $\mu(\mathcal{X})$ its volume where $\mu(\cdot)$ stands for the standard Lebesgue measure or the counting measure if $\mathcal{X}$ is countable. Finally, let

$$\text{Lip}(k) = \left\{f : \mathcal{X} \to \mathbb{R} \; s.t. \; \forall x, x' \in \mathcal{X}^2, |f(x) - f(x')| \leq k\|x - x'\|_2\right\}$$

the set of $k$-Lipschitz functions defined on $\mathcal{X}$.

# 2 Related work

As stated in the introduction, global optimization is a very interesting field for solving industrial problems, so one can find a lot of papers describing variants and merges of algorithms or applications on new problems. One of the most known global optimization algorithm is Simulated Annealing [7]. It is based on the Metropolis-Hastings algorithm ([12] and [5]) as it samples random configurations and accepts it in a exploration-exploitation perspective. More generally, one can distinguish deterministic methods from stochastic ones.

Deterministic methods provide theoretical guarantee that the returned solution is the real global maximum. Cutting-plane methods [2] and branch and bounds methods [8] are two extremely popular deterministic algorithms for solving discrete NP-hard problems. One can also cite [1], the state of the art algorithm based on real algebraic geometry, also discrete.

On the other hand, stochastic methods are based on Monte-Carlo and can return inexact results. LIPO and AdaLIPO are stochastic methods, as Bayesian optimization

methods [11], that construct a gaussian process that best describes the function to be optimized and samples the next evaluation point given this posterior distribution. As LIPO and AdaLIPO, Bayesian optimization methods are advantageous when the function is difficult to evaluate. Another interesting stochastic algorithm is CMA-ES [4], it samples next evaluation points using a multivariate gaussian distribution with mean and covariance maximizing the likelihood of previous candidate solutions and search steps. Finally, in 2018 [9] established a relationship between any continuous function $f$ on a compact set $\Omega \subset \mathbb{R}^d$ and its global minima $f^*$:

$$\int_\Omega f(x)m^{(k)}(x)\mu(\mathrm{d}x) \xrightarrow[k\to\infty]{} f^* \text{ , where } m^{(k)}(x) = \frac{e^{-kf(x)}}{\int_\Omega e^{-kf(y)}\mu(\mathrm{d}y)}.$$

Note that:

$$\int_\Omega m^{(k)}(x)\mu(\mathrm{d}x) = 1 \text{ and } \lim_{k\to\infty} m^{(k)}(x) = \begin{cases} \frac{1}{\mu(X^*)} & \text{if } x \in X^* \\ 0 & \text{if } x \in \Omega/X^* \end{cases}$$

where $X^* \subseteq \Omega$ is the set of minimizers. Thus, $m^{(k)}$ is a probability density function, that one can think of as a kind of generalized softmax for infinite dimension, that pointwise converges to a uniform distribution on $X^*$.

# 3 Methods

In this section we provide a reference implementation of LIPO and AdaLIPO. First, we decided to use oriented object programming to define the objective function. We thus defined a class **Function** which has two attributes and one method:

- **bounds**: The bounds of the definition space of the function for each dimension *(attribute)*
- **k**: The Lipschitz constant of the function *(attribute)*
- **call**: The closed form of the objective function *(method)*

The two presented algorithms' pseudo code are available in Section A.

## 3.1 LIPO

The LIPO algorithm improvement, compared to Pure Random Search, is that it evaluates the objective function only when a certain condition is verified:

$$\min_{i=1..t} \left(f\left(X_i\right) + k \cdot \left\|X_{t+1} - X_i\right\|_2\right) \geq \max_{i=1...t} f\left(X_i\right). \tag{1}$$

The meaning of this condition can be understood in depth in [10], but it basically means that we won't explore a point for which, according to our observations, we can be sure that the evaluation will be below one of our other observations, thus it can't be the maximum.
This condition is implemented as followed in Algorithm 1. Then, we define the LIPO algorithm in Section A Algorithm 3.
With this algorithm, we have new confidence intervals on the value of the maximum, described below.

### 3.1.1 Naive LIPO Upper Bound

For any $f \in \mathrm{Lip}(k)$, any $n \in \mathbb{N}^\star$ and $\delta \in (0,1)$, we have with probability at least $1-\delta$,

$$\max_{x\in\mathcal{X}} f(x) - \max_{i=1\dots n} f(X_i) \leq k \cdot \mathrm{diam}(X) \cdot \left(\frac{\ln(1/\delta)}{n}\right)^{\frac{1}{d}}. \tag{2}$$

### 3.1.2 Fast LIPO Bounds

When the following conditions are satisfied:

**LIPO bounds conditions**

1. the global optimizer $x^\star \in \mathcal{X}$ is unique;
2. for some $\kappa \geq 1$, $c_\kappa \geq 0$, for all $x \in \mathcal{X}$:

$$f(x^\star) - f(x) \geq c_\kappa \cdot \|x - x^\star\|_2^\kappa. \tag{3}$$

we have faster rates for the same confidence intervals:

**Fast LIPO Bounds:** for any $n \in \mathbb{N}^\star$ and $\delta \in (0,1)$, we have with probability at least $1-\delta$ the following upper bound,

$$\begin{aligned}\max_{x\in\mathcal{X}} f(x) - \max_{i=1\dots n} f(X_i) \leq & k \cdot \mathrm{diam}(\mathcal{X}) \\ & \times \begin{cases} \exp\left(-C_{k,\kappa} \cdot \frac{n\ln(2)}{\ln(n/\delta)+2(2\sqrt{d})^d}\right) & \text{if } \kappa = 1 \\ \frac{2^\kappa}{2}\left(1 + C_{k,\kappa} \cdot \frac{n\left(2^{d(\kappa-1)}-1\right)}{2\ln(n/\delta)+2(2\sqrt{d})^d}\right)^{-\frac{\kappa}{d(\kappa-1)}} & \text{if } \kappa > 1 \end{cases}\end{aligned} \tag{4}$$

where $C_{k,\kappa} = (c_\kappa \max_{x\in\mathcal{X}} \|x - x^\star\|^{\kappa-1}/8k)^d$, and the following lower bound:

$$c_\kappa \cdot \mathrm{rad}(\mathcal{X})^\kappa \cdot e^{-\frac{\kappa}{d}\left(n + \sqrt{2n\ln(1/\delta)} + \ln(1/\delta)\right)} \leq \max_{x\in\mathcal{X}} f(x) - \max_{i=1\dots n} f(X_i). \tag{5}$$

## 3.2 AdaLIPO

The AdaLIPO algorithm is a variant for LIPO when the Lipschitz constant of the objective function is unknown. AdaLIPO provides a way to estimate this constant while optimizing the function. The estimated constant is given according to the Eq. 6:

$$\hat{k}_t \triangleq \inf\left\{k_{i\in\mathbb{Z}} : \max_{i\neq j} \frac{|f(X_i) - f(X_j)|}{\|X_i - X_j\|_2} \leq k_i\right\}. \tag{6}$$

We can choose any series $(k_i)_{i\in\mathcal{Z}} \in \mathbb{R}$. In this paper, we choose to use one series defined in [10]:

$$k_i = (1+\alpha)^i.$$

Thus,

$$\hat{k}_t = (1+\alpha)^{i_t}, \text{ where } i_t = \left\lceil \ln\left(\max_{i\neq j} \frac{|f(X_i) - f(X_j)|}{\|X_i - X_j\|_2}\right) \frac{1}{\ln(1+\alpha)} \right\rceil. \tag{7}$$

The estimation of the Lipschitz constant of AdaLIPO ensures that the constant $k$ cannot be overestimated of more than one term in the series. For example, if $k_i < k < k_{i+1}$, AdaLIPO ensures that $\hat{k}_t \leq k_{i+1}$.

The AdaLIPO algorithm is defined in Section A Algorithm 4.

For this algorithm, we also have "naives" rates if conditions defined in 3.1.2 are not satisfied.

**Naive AdaLIPO Upper Bound:**

$$\max_{x\in\mathcal{X}} f(x) - \max_{i=1\dots n} f(X_i) \leq k_{i^\star} \times \mathrm{diam}\mathcal{X} \times \left(\frac{5}{p} + \frac{2\ln(\delta/3)}{p\ln(1-\Gamma(f,k_{i^\star-1}))}\right)^{\frac{1}{d}} \times \left(\frac{\ln(3/\delta)}{n}\right)^{\frac{1}{d}} \tag{8}$$

where $\Gamma(f,k_{i^*-1}) \triangleq \mathbb{P}\Big(\frac{|f(X_1)-f(X_2)|}{\|X_1-X_2\|_2} > k_{i^*-1}\Big) > 0$ and $i^\star = \min\{i \in \mathbb{Z} : f \in \mathrm{Lip}(k_i)\}$.

When 3.1.2 are satisfied, we have a faster upper bound.

**Fast AdaLIPO Upper Bound:**

$$\max_{x\in\mathcal{X}} f(x) - \max_{i=1\dots n} f(X_i) \leq k_{i^\star} \times \mathrm{diam}(\mathcal{X}) \times \exp\left(\frac{2\ln(\delta/4)}{p\ln(1-\Gamma(f,k_{i^\star-1}))} + \frac{7\ln(4/\delta)}{p(1-p)^2}\right) \times \begin{cases} \exp\Big(-C_{k_{i^*},\kappa}\cdot\frac{n(1-p)\ln(2)}{2\ln(n/\delta)+4(2\sqrt{d})^d}\Big) & \text{if } \kappa = 1, \\ 2^\kappa\left(1 + C_{k_{i^*},\kappa}\cdot\frac{n(1-p)\left(2^{d(\kappa-1)}-1\right)}{2\ln(n/\delta)+4(2\sqrt{d})^d}\right)^{-\frac{\kappa}{d(\kappa-1)}} & \text{if } \kappa > 1 \end{cases} \tag{9}$$

where $C_{k_{i^\star},\kappa} = (c_\kappa \max_{x\in\mathcal{X}}\|x-x^\star\|_2^{\kappa-1}/8k_{i^\star})^d$.

# 4 Experimental analysis

The main purpose of this paper is to provide improvements through experimental analysis for LIPO and AdaLIPO. We led several experiments on popular global optimization benchmark functions.

## 4.1 Selected benchmark functions

As global optimization algorithms such as LIPO can be used for low dimensional problems, we picked 2D functions for visualization purposes. Note that any of the code can be executed on higher dimensional problems also. As our optimization aims to find a global maximum here, some of the functions we selected are the opposite of "classical" benchmark functions.

#### 4.1.1 Himmelblau function

This functions writes

$$f(x,y) = -(x^2 + y - 11)^2 - (x + y^2 - 7)^2, \forall x, y \in \mathbb{R}. \tag{10}$$

This function has 4 global maxima, at

- $(3, 2)$
- $(-2.805118..., 3.131312...)$
- $(-3.779310, -3.283186...)$
- $(3.584428..., -1.848126...)$

such that $f = 0$ at those 4 points. It has a Lipschitz constant of approximately 283 over $[-4, 4]^2$, and does not satisfy the fast rate property 3.1.2.

#### 4.1.2 Hölder table function

This functions writes

$$f(x,y) = \left|\sin(x)\cos(y)\exp\left(\left|1 - \frac{\sqrt{x^2+y^2}}{\pi}\right|\right)\right|, \forall x, y \in \mathbb{R}. \tag{11}$$

This function has 4 global maxima, at

- $(8.05502, 9.66459)$
- $(-8.05502, 9.66459)$
- $(8.05502, -9.66459)$
- $(-8.05502, -9.66459)$

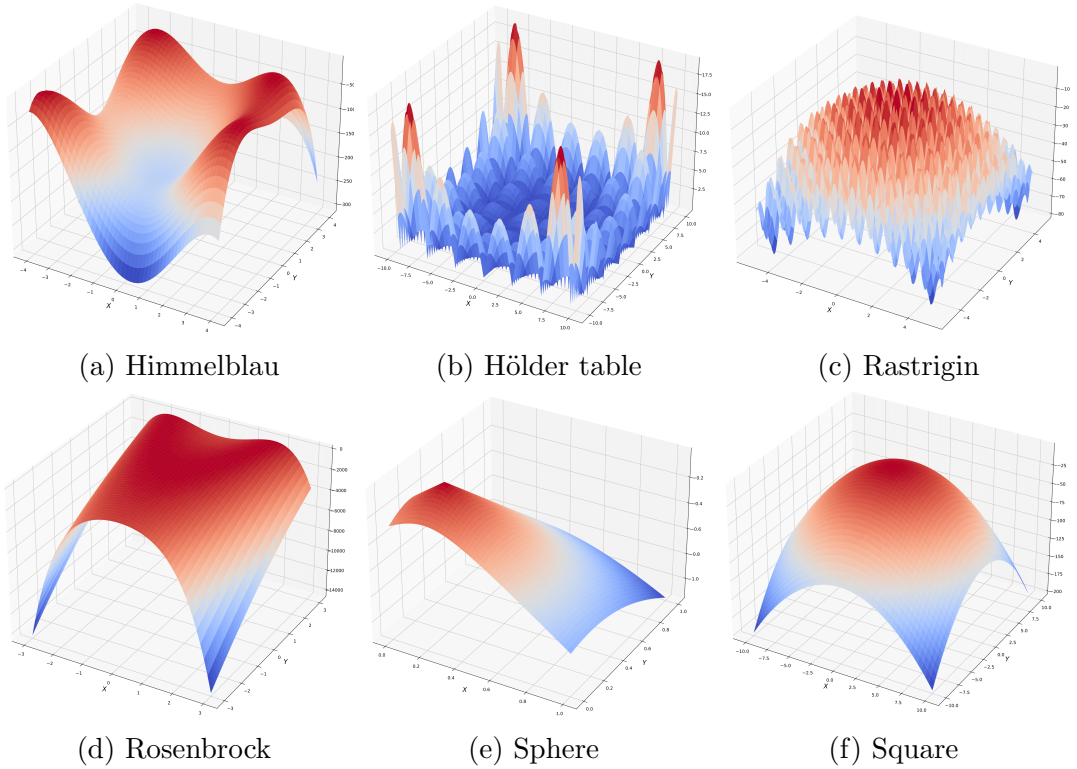
such that $f = 19.2085$ at those 4 points. It has a Lipschitz constant of approximately 30 over $[-10, 10]^2$, and does not satisfy the fast rate property 3.1.2.

#### 4.1.3 Rastrigin function

This functions writes, in dimension $n$,

$$f(x_1, ..., x_n) = -10n - \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right). \tag{12}$$

This function has 1 global maximum, at $(0, ..., 0)$, such that $f = 0$. In dimension 2, it has a Lipschitz constant of approximately 96 over $[-5.12, 5.12]^2$, and satisfies the fast rate property 3.1.2, with $k = 2, c_k = 1 + 20\pi^2$.



(a) Himmelblau (b) Hölder table (c) Rastrigin

(d) Rosenbrock (e) Sphere (f) Square

Figure 3: Graphs of benchmark functions in 2D.

#### 4.1.4 Rosenbrock function

This functions writes,

$$f(x, y) = -(1 - x)^2 - 100(y - x^2)^2. \tag{13}$$

This function has 1 global maximum, at $(1, 1)$, such that $f = 0$. It has a Lipschitz constant of approximately 14607 over $[-3, 3]^2$, and does not satisfy the fast rate property 3.1.2.

#### 4.1.5 Sphere function

This functions writes, in dimension $n$

$$f(x_1, ..., x_n) = -\sqrt{\left(x - \frac{\pi}{16}\right)^2 + \left(y - \frac{\pi}{16}\right)^2}. \tag{14}$$

This function has 1 global maximum, at $(0, 0)$, such that $f = 0$. In dimension 2, it has a Lipschitz constant of approximately 1.5 over $[0, 1]^2$, and satisfies the fast rate property 3.1.2, with $k = 1, c_k = 1$.

#### 4.1.6 Square function

This functions writes, in dimension $n$

$$f(x_1, ..., x_n) = -\sum_{i=1}^{n} x_i^2 = -\|x\|_2^2. \tag{15}$$

This function has 1 global maximum, at $(0, ..., 0)$, such that $f = 0$. In dimension 2, it has a Lipschitz constant of $20\sqrt{2}$ over $[-5.12, 5.12]^2$, and satisfies the fast rate property 3.1.2, with $k = 2, c_k = 1$.

## 4.2 Experiment setup

For both LIPO and AdaLIPO, we fixed a number of steps $n = 2000$ which represents the maximum number of times we will evaluate the function. Remember that what is supposed to be costly in this kind of problem is not the number of samples we draw, but the evaluation of the function (typically a simulation, costing a lot of time to be executed). We compared both algorithms with Pure Random Search (PRS) on the 6 functions defined in 4.1. For AdaLIPO, we set $p = 0.5$ and $\alpha = 0.01$ (needed in Eq. 7). Each algorithm is stopped when it reaches $n$ iterations or when it gets close enough to the known maximum. To evaluate the latter, we used a metric defined in [10]:

$$\text{Stop at iteration } i \text{ if } f(X_i) \geq f_{target}(t) \tag{16}$$

where

$$f_{target}(t) = \max_{x \in \mathcal{X}} f(x) - \left( \max_{x \in \mathcal{X}} f(x) - \int_{x \in X} f(x) \frac{\mathrm{d}x}{\mu(\mathcal{X})} \right)(1 - t), \ \forall t \in [0, 1]. \tag{17}$$

This condition allows us to stop the algorithm whenever we consider it is close enough to the true maximum, given the parameter $t$. The bigger $t$, the closer the algorithm must be to the true maximum. In our experiment, we fixed $t = 0.99$. To compute Eq. 17, we need the value of the true maximum (which we know because we chose the the benchmark functions accordingly), and the value of the integral, which we estimated using a Monte-Carlo method with $10^6$ samples.

## 4.3 Results

We ran 100 times the experiment for each algorithm on each benchmark function. The results are shown in Table 1. As stated in the original paper, both algorithms are way better than PRS and the AdaLIPO quickly converges to the same results as LIPO.

| | Himmelblau | Hölder | Rastrigin | Rosenbrock | Sphere | Square |
|---|---|---|---|---|---|---|
| PRS | 184 ± 185 | 1245 ± 686 | 1950 ± 236 | 13 ± 13 | 1811 ± 436 | 188 ± 152 |
| LIPO | 100 ± 86 | 508 ± 217 | **670 ± 183** | **11 ± 10** | 46 ± 10 | **43 ± 22** |
| AdaLIPO | **97 ± 77** | **319 ± 201** | 913 ± 297 | 12 ± 11 | **28 ± 8** | 62 ± 47 |

Table 1: Results of the numerical experiments. The table shows the total number of function evaluations (mean ± standard deviation) required to meet the condition stated in Eq. 16. We clearly see that LIPO and AdaLIPO outperform PRS. On some functions, AdaLIPO converges faster than LIPO which can be due to an underestimation of the Lipschitz constant around the maximum.

For each function, we observed the same behavior of LIPO and AdaLIPO. While PRS simply tests uniformly the whole set, LIPO and AdaLIPO first sample uniformly

before sampling mostly in the region of interest. This behavior is even more pronounced for AdaLIPO when we reduce the parameter $p$ for exploration. However, reducing $p$ too much might be problematic for the algorithm to properly estimate the Lipschitz constant of the function, as it won't have a representation of the function's graph precise enough. A too large underestimation of the Lipschitz constant could lead the LIPO's condition (Eq. 1) to erase regions where the maximizers live. On the contrary, setting $p$ close to 1 will make AdaLIPO behave like PRS. Figure 5 illustrates this behavior on the Himmelblau function. Moreover, as we can see on Figure 6, AdaLIPO tends to underestimate the real Lipschitz constants when $p$ decreases. This is due to the fact that the constants are estimated within local regions where the sparseness is proportional to $p$. Within these regions, the functions can be more regular than when considering the entire compact sets $\mathcal{X}$. However, this behavior can lead to faster convergence, as we only are interested in the neighborhood of $x^*$.

Thus, we need to find a good compromise between enough exploration to have a good estimation of the Lipschitz constant and fast convergence through exploitation.

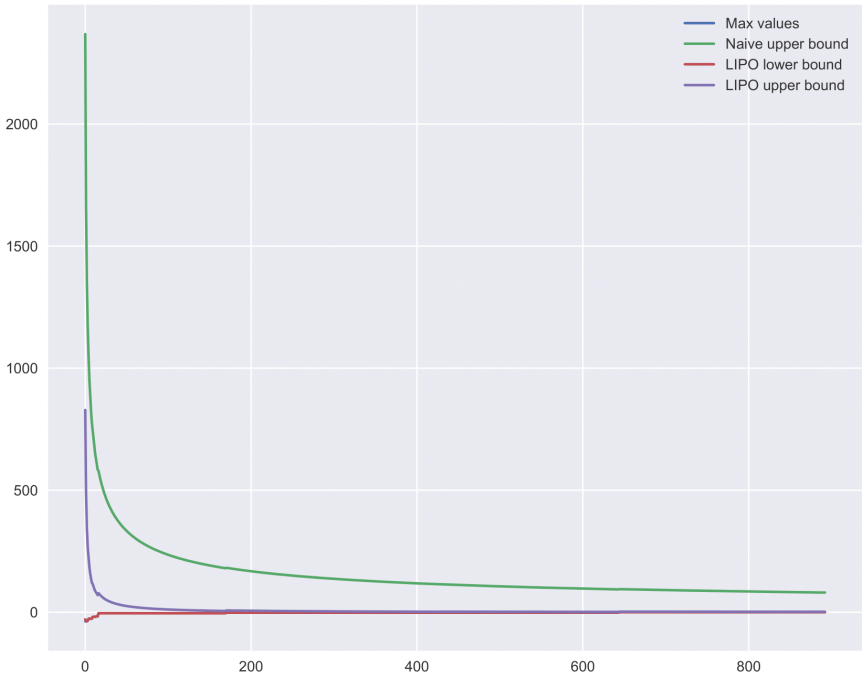
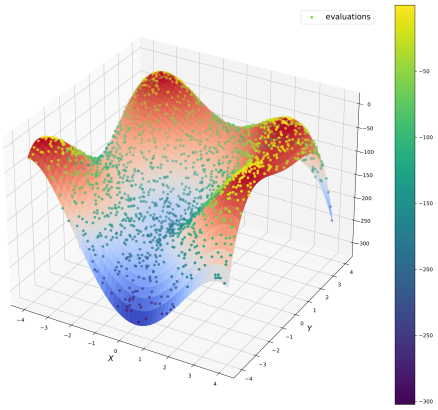
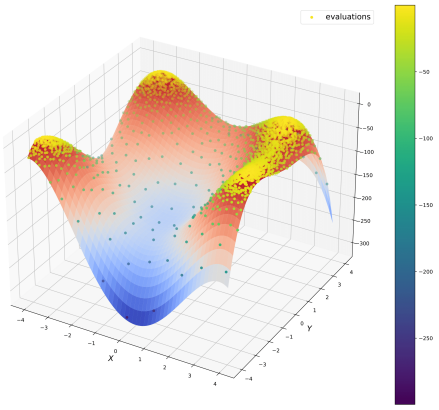


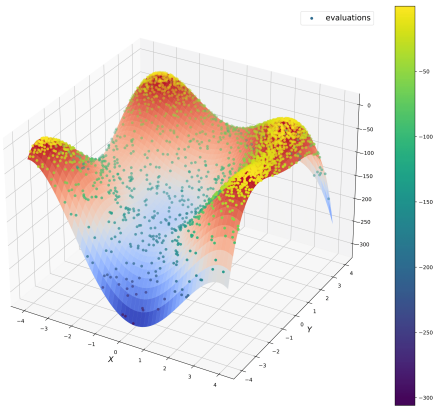

Figure 4: Lower and upper bounds estimation for the maximum value for Rastrigin function through the evaluations given by Eqs. 2 and 4. Fast rates are here much closer to the true value of the maximum (which is 0) than naive rates. The maximum value is very close thus hidden by the LIPO lower bound curve.



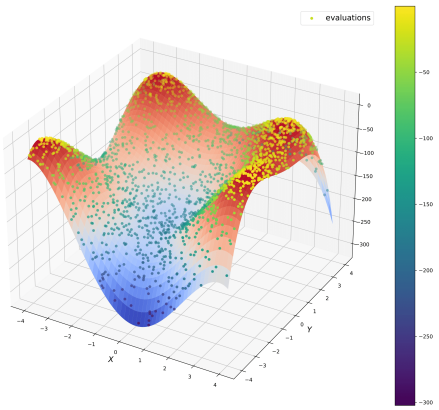

(a) PRS algorithm




(b) LIPO algorithm

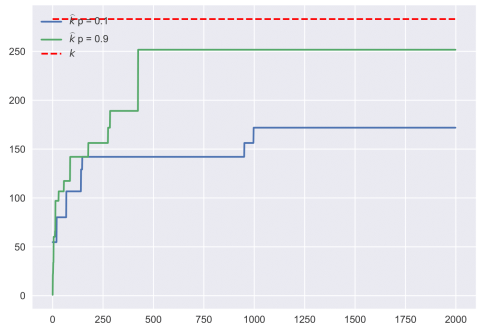



(c) AdaLIPO algorithm with $p = 0.5$

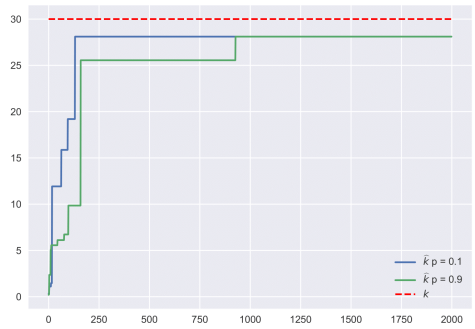



(d) AdaLIPO algorithm with $p = 0.9$

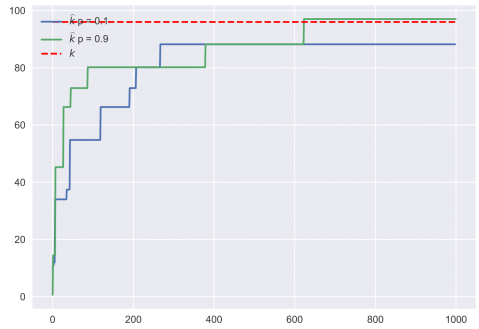
Figure 5: Visual comparison between each algorithm on the Himmelblau function. As PRS (a) explores uniformly $\mathcal{X}$, LIPO (b) and AdaLIPO (c & d) methods tend to explore only the interesting areas to find the maximum. Also, we see the impact of the exploration rate $p$ from AdaLIPO: the higher $p$, the more we explore and the more AdaLIPO will behave like PRS.



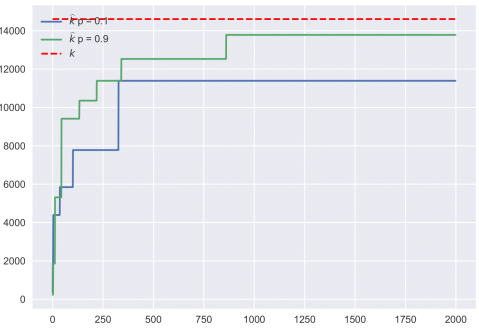

(a) Himmelblau



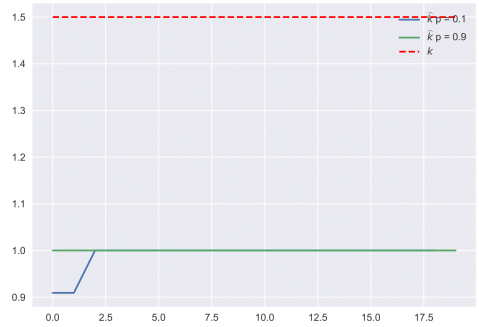

(b) Hölder



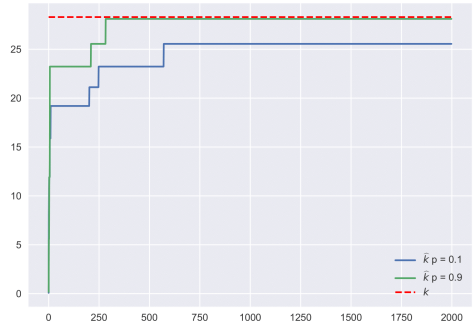

(c) Rastrigin




(d) Rosenbrock




(e) Sphere




(f) Square

Figure 6: Examples of Lipschitz constant estimations of AdaLIPO with $p = 0.1$ and $p = 0.9$ on the benchmark functions. The graphs represent the evolution of the estimated Lipschitz constant during the optimization process. We observe that AdaLIPO often tends to underestimate the Lipschitz constant, which can be explained because the "true" Lipschitz constant around the maximum is smaller than the one on $\mathcal{X}$. The bigger is $p$, the closer the estimate is to the real value.

### 4.3.1 Convergence analysis

We led statistical analysis on the convergence of both LIPO and AdaLIPO. Using Eqs. 2, 4, and 9, we obtain for each function graphics showing the 95% confidence interval in which $f(x^*)$ should be. In most cases, when the conditions for Eq.4 are

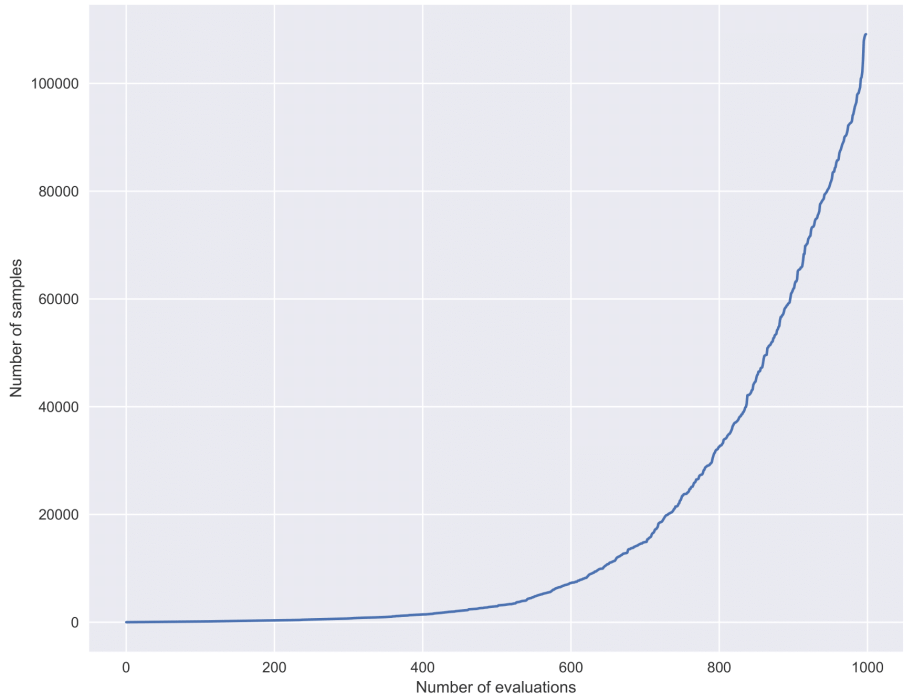



Figure 7: Number of samples with respect to the number of function evaluations for LIPO on the Rastrigin function. The growth appears to be exponential, and in fact is hyperbolic as the more we evaluate, the less we are likely to accept a new point (to the point where we won't accept any new point).

satisfied, we see in Figure 4 that the convergence of the confidence interval is way faster than using the boundaries given by Eq.2.

# 5 Empirical improvements

## 5.1 Stopping criterion

While experimenting with both LIPO and AdaLIPO, we noticed, as expected, that the number of samples drawn from a uniform law grows exponentially (in fact probably hyperbolically) with respect to the number of function evaluations as shown in Figure 7. This makes us want to define a stopping criterion that is different from `while t < n` with fixed `n`, because it might take a really long time to converge, or not converge at all (as the condition Eq.1 will never be satisfied). The improvement we propose in this paper consists in stopping the algorithm as soon as the slope of the plot in Figure 7 exceeds a threshold $\gamma$. The bigger $\gamma$, the longer but the more precise the result will be. In practice, we approximate $\gamma$ with finite differences on several iterations (because from the one to the other, there might be too much randomization and a moment of bad luck can stop the algorithm too soon). The pseudocode for this stopping criterion is described in 2.

## 5.2 Decreasing Bernoulli parameter for AdaLIPO

For AdaLIPO, as explained in Section 3, it has two possibilities at each iteration: exploration or exploitation. The choice of the path it will take is given by the result of a Bernoulli random variable $\mathcal{B}(p)$. By intuition and by experience, we can think that exploring a lot at the beginning of the algorithm is a good way to approximate correctly the Lipschitz constant $k$, whereas the exploitation phase is the one where we will really use the observations we already have to find the maximum. That's why we propose to decrease progressively the parameter $p$ during the optimization, for it to be high enough in the beginning, and to be small enough at the end so we can focus on exploiting our observations. In a way, one can think of this as a first phase where the algorithm focuses on estimating the Lipschitz constant, and a second phase where it basically applies LIPO to this estimated Lipschitz constant. Of course, the two phases are not really separated (we propose a continuous decrease of $p = p(t)$), but that's the idea behind it.
For instance, we empirically suggest to take $p(t) = \min\left(1, \frac{1}{\ln(t)}\right)$, with the convention $\frac{1}{0} = +\infty$.

These two improvements resulted in new versions of LIPO and AdaLIPO, referred to as LIPO-E and AdaLIPO-E in the following.

## 5.3 Empirical comparison

We decided to compare both versions of LIPO and AdaLIPO, with and without the proposed improvements. Regarding the stopping criterion, we set $\gamma = 800$ and the slope is computed over the 5 last number of samples. We ran the 4 algorithms on some benchmark functions defined in Section 4.1. We fixed a maximum number of evaluations for each function, depending on their difficulties. We removed the target stopping condition defined in Eq. 16, therefore, the original version of LIPO and AdaLIPO exhausted the maximum number of evaluations allowed. However, the enhanced versions could stop earlier if the slope condition defined in Section 5.1 is reached. The other hyperparameters of the experiment are the same as in Section 4.2, except that we ran 10 times the experiment instead of 100 times. The results are shown in Table 2. We chose to display only the 3 benchmark functions where the slope stopping criterion is met. We can see that, with significantly fewer iterations, the distances from the real global maximum of our enhanced algorithms are in the same order of magnitude and sometimes less than the original versions.
As expected, our stopping criterion allows to totally emancipate from the maximum number of evaluations, which is an arbitrary parameter. Indeed, the original versions of LIPO and AdaLIPO stop when they evaluate the function $n$ times. However, sampling a potential maximizer (a point verifying Eq. 1) can be very difficult; for instance if the Lipschitz constant is small or if the algorithm already has a good approximation of the maximum. Our stopping criterion ensures that, if the algorithm struggles too much to sample a potential maximizer, it stops. One can run LIPO or AdaLIPO until it met the slope stopping criterion, the resulting value will be the best value one can obtain using the algorithm in a reasonable amount of time.

| | Hölder | | Rastrigin | | Sphere | |
|---|---|---|---|---|---|---|
| | *# evals* | $d_{max}$ | *# evals* | $d_{max}$ | *# evals* | $d_{max}$ |
| LIPO | 2000 | **0.0018** | 1000 | **0.0512** | 25 | 0.0137 |
| LIPO-E | 1505 ± 104 | **0.0018** | 869 ± 34 | 0.1282 | 25 | 0.0395 |
| AdaLIPO | 2000 | 0.003 | 1000 | 0.4106 | 25 | 0.0227 |
| AdaLIPO-E | **719 ± 457** | 0.023 | **753 ± 133** | 0.0569 | **20 ± 5** | **0.0063** |

Table 2: Comparison between the original algorithms and our empirically improved versions. *# evals* represents the number of function evaluations and $d_{max}$ the distance from the real maximum. For AdaLIPO-E, $\gamma = 800$. We can see that, with significantly less evaluation, LIPO-E and AdaLIPO-E compete with the original version in terms of distance to the real maximum.

We also compared AdaLIPO using only the decreasing Bernoulli parameter (referred as AdaLIPO-B) with LIPO and AdaLIPO in the exact same experimental setup as in Section 4.2. The results are summarized in Table 3. As we can see, AdaLIPO-B significantly outperforms the original AdaLIPO on this benchmark. It even succeeded to beat LIPO on almost every problem. In the next section, we highlight how our empirical improvements allows such a performance gain.

| | Himmelblau | Hölder | Rastrigin | Rosenbrock | Sphere | Square |
|---|---|---|---|---|---|---|
| LIPO | 100 ± 86 | 508 ± 217 | 670 ± 183 | **11 ± 10** | 46 ± 10 | **43 ± 22** |
| AdaLIPO | 97 ± 77 | 319 ± 201 | 913 ± 297 | 12 ± 11 | 28 ± 8 | 62 ± 47 |
| AdaLIPO-B | **65 ± 46** | **228 ± 136** | **616 ± 187** | **11 ± 10** | **22 ± 6** | 51 ± 36 |

Table 3: Empirical performance of our enhanced AdaLIPO. The table shows the total number of function evaluations (mean ± standard deviation) required to meet the condition stated in Eq. 16. AdaLIPO-B outperforms LIPO and AdaLIPO in almost every problem.

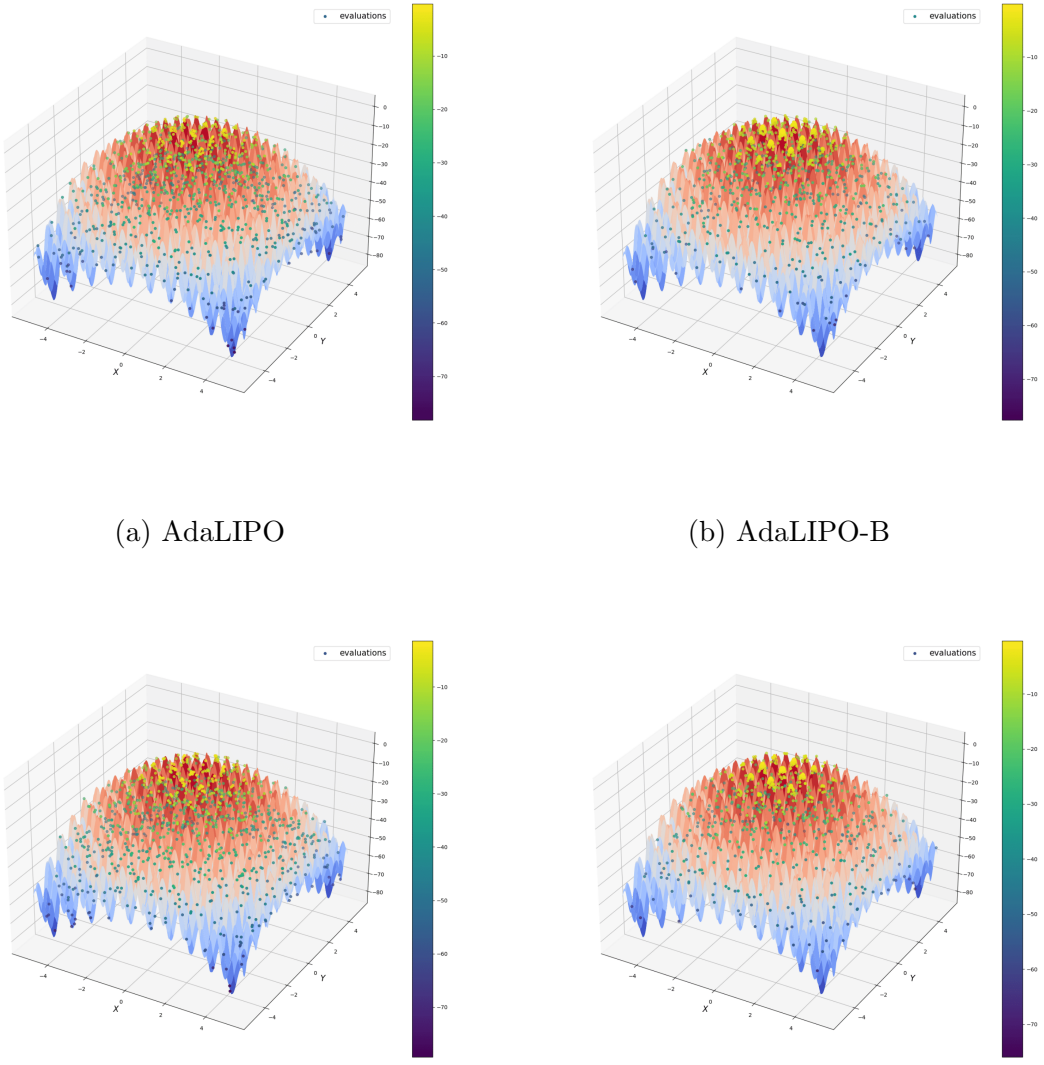
## 5.4 Understanding the improvements' contributions

This section aims at understanding the contribution of the two improvements introduced in Section 2. Indeed, the enhanced version of AdaLIPO includes the two improvements (decreasing Bernoulli parameter and Stopping criterion) and leads to better performance for some benchmark functions, both in terms of the number of evaluations and distance to the maximum (i.e. the difference between the real maximum value and our best observation). To understand better these results we need to identify the separate contribution of the improvements. To this end, we conducted an experiment where we compared the original AdaLIPO, AdaLIPO-E and two other AdaLIPO versions one called AdaLIPO-B which includes only the Bernoulli parameter and AdaLIPO-slope which only includes the stopping criterion. We studied the behavior of these 4 algorithms for 4 benchmark functions. We fixed the number of evaluations to 2000 (except from Rastrigin function where we fixed this parameter to 1000) and ran 100 times each experiment. The results are summarized in Table 4. We fixed the following hyperparameters: slope's size to 5, maximum slope to $\gamma = 1000$ and $p = 0.5$.

On the one hand we observed that the Bernoulli parameter allows a better estimation of the function's maximum while slightly degrading the Lipschitz constant estimation compared to the classic AdaLIPO. This improvement also increases the number of samples however, this step is not expensive. On the other hand, the stopping criterion, as expected, reduced the number of function evaluations which is the expensive step of the method. Combined these two improvements lead to AdaLIPO-E. Thus, we can see that our version of this algorithm inherits the properties of the two improvements. Indeed, the maximum estimation is better while having in general a slightly worse Lipschitz constant estimation but proceeding to less evaluation. It corroborates what we assumed in Section 4.3, i.e. underestimating the Lipschitz constant can lead to faster convergence. We also provide with a visual illustration of the improvements' behaviors in Figure 8.

| | | Himmelblau | Hölder | Rosenbrock | Square | Rastrigin |
|---|---|---|---|---|---|---|
| AdaLIPO | $d_{max}$ | 0.0188 ± 0.0188 | 0.0024 ± 0.0029 | 0.0529 ± 0.0487 | 0.0028 ± 0.0025 | 0.4066 ± 0.3439 |
| | # samples | 9846 ± 1286 | 133217 ± 61770 | 2793.01 ± 56.25 | 42806 ± 2753 | 21028 ± 9583 |
| | $\kappa$ | 224.9 ± 19.7 | 27.84 ± 2.36 | 13358.31 ± 671.16 | 28.1024 ± 0.00 | 94.46 ± 4.00 |
| | # *evals* | 2000 | 2000 | 2000 | 2000 | 1000 |
| AdaLIPO-B | $d_{max}$ | 0.0061 ± 0.0058 | 0.0010 ± 0.0017 | 0.034 ± 0.034 | 0.001 ± 0.0008 | 0.0632 ± 0.0960 |
| | # samples | 22745 ± 3975 | 519322 ± 410188 | 3458 ± 119 | 113937 ± 9827 | 178520 ± 98026 |
| | $\kappa$ | 201.74 ± 20.09 | 26.36 ± 4.09 | 12151.9 ± 977.6 | 27.03 ± 1.26 | 90.8434 ± 4.0417 |
| | # *evals* | 2000 | 2000 | 2000 | 2000 | 1000 |
| AdaLIPO-slope | $d_{max}$ | 0.0136 ± 0.0111 | 0.003 ± 0.003 | 0.0391 ± 0.0361 | 0.0029 ± 0.0027 | 0.4040 ± 0.3923 |
| | # samples | 9870 ± 1187 | 109004 ± 34252 | 2779 ± 58 | 42582 ± 2342 | 19086 ± 9630 |
| | $\kappa$ | 221.24 ± 18.98 | 28.08 ± 2.43 | 13358 ± 677.8 | 28.10 ± 0.00 | 95.6237 ± 3.7340 |
| | # *evals* | 2000 | 1976.22 ± 126.23 | 2000 | 2000 | 999.5400 ± 4.5769 |
| AdaLIPO-E | $d_{max}$ | 0.0062 ± 0.0056 | 0.0027 ± 0.0041 | 0.0352 ± 0.0295 | 0.001 ± 0.0009 | 0.0825 ± 0.0819 |
| | # samples | 23596 ± 4171 | 168210 ± 66780 | 3463 ± 127 | 114184 ± 9295 | 72081 ± 16582 |
| | $\kappa$ | 199.52 ± 22.47 | 24.02 ± 5.84 | 12302 ± 1070 | 27.05 ± 1.26 | 90.2581 ± 4.3964 |
| | # *evals* | 2000 | 1399 ± 567 | 2000 | 2000 | 894.8700 ± 124.5155 |

Table 4: Empirical performance of the two proposed improvements of our enhanced AdaLIPO. The table shows the contributions of each improvement by evaluating several criteria. The total number of function evaluations (mean ± standard deviation), the distance from the real maximum $d_{max}$, the estimation of the Lipschitz constant $\kappa$ and the number of samples. We see that AdaLIPO-B has the best maximum estimation in general while AdaLIPO-slope stops earlier but with a worse estimation. AdaLIPO-E is a combination of the two improvements and has the best performance in terms of maximum estimation and the number of function evaluations.




(a) AdaLIPO

(b) AdaLIPO-B

(c) AdaLIPO-slope

(d) AdaLIPO-E

Figure 8: Visual understanding of the contributions of the proposed improvements. Compared to AdaLIPO (a), AdaLIPO-B (b) tends to explore more the region of interest (which can be seen with a higher number of evaluations at the center of $\mathcal{X}$, in yellow). AdaLIPO-slope (c) reduces its overall number of function evaluations, without modifying its exploration-exploitation parameter, and AdaLIPO-E (d) combines those two behaviors.

# 6 Application scope of LIPO

As many optimization algorithms, LIPO is subject to the curse of dimensionality. In fact, we can show that in some sense, LIPO converges towards PRS in high dimension. Indeed, if we want to optimize $f$ on a "non-small" subset $\mathcal{X}$ of $\mathbb{R}^d$, by looking at our algorithm 3, we will at iteration $t+1$ *erase* from our search space, at best, the

ball $B\left(x_{t+1}, \frac{\alpha}{k}\right)$ with $k$ being the Lipschitz constant of the objective function, and $\alpha \geq \max_{x\in\mathcal{X}} f(x) - \min_{x\in\mathcal{X}} f(x)$. For LIPO to be efficient we need:

- to reduce the search space, so we need to have a control on $k$ (for it not to be too big);
- to have a control on $\mu(\mathcal{X})$ (for it not to be too big as well).

The main problem is that classical functions in high dimensions are usually defined on quite big subspaces $\mathcal{X}$, with high Lipschitz constants $k$. To summarize, in the best case, we will have a search space of volume

$$V_t = \mu(\mathcal{X}) - t\frac{\pi^{d/2}\alpha^d}{k^d\Gamma(\frac{d}{2}+1)}$$

at iteration $t+1$. Thus, the probability to reject the point at iteration $t+1$ will be in the best scenario:

$$\mathbb{P}(\text{Reject}(x_{t+1})) = t\frac{\pi^{d/2}\alpha^d}{k^d\Gamma\left(\frac{d}{2}+1\right)\mu(\mathcal{X})} \tag{18}$$

i.e., the probability for $x_{t+1}$ to be in the erased space. Most of the time, $t\frac{\pi^{d/2}\alpha^d}{k^d\Gamma\left(\frac{d}{2}+1\right)\mu(\mathcal{X})}$ will be tiny.

To properly see it, let's take a simple example. Define $f$ as the square function 15. Let's take $\mathcal{X} \triangleq [-1,1]^d$ which verifies $\mu(\mathcal{X}) = 2^d$. For this subset, the Lipschitz constant will be $k = 2^d$ as well, and $\alpha = d$. Then, using 18, we will have

$$\mathbb{P}(\text{Reject}(x_{t+1})) \leq t\frac{\pi^{d/2}d^d}{(2^d)^d\Gamma\left(\frac{d}{2}+1\right)2^d} \triangleq tC_d. \tag{19}$$

For instance, the value of $C_d$ is 0.78 for $d = 2$, 0.22 for $d = 3$, $4.9 \times 10^{-4}$ for $d = 5$, $2.0 \times 10^{-20}$ for $d = 10$ etc.

Therefore, LIPO and, by extension, AdaLIPO, are not good choices of algorithms for maximizing a high dimensional function.

# 7 Conclusion

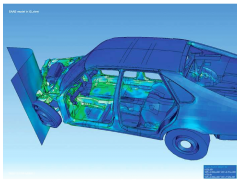
In this paper, we provide a reproducible implementation of both LIPO and AdaLIPO and we tested them on classical benchmark functions for global optimization. Through numerical experiments, we also propose two intuitive improvements: a decreasing exploration-exploitation parameter for AdaLIPO, and a stopping criterion based on the number of rejected samples for both algorithms. We compared original and enhanced versions of both algorithms on our benchmark with respect to the number of evaluations of the objective function and the distance to the real maximum. Those experiments show significant improvements of the two methods (LIPO and AdaLIPO), with respect to both criteria. The stopping criterion allows the algorithm to reduce the number of evaluations of the function in exchange of a small decrease of the precision,

whereas the progressive decrease of the Bernoulli parameter often increases precision due to an evolving exploration-exploitation trade-off. We also discussed on the application scope of LIPO and AdaLIPO, stating that these algorithms quickly tend to PRS with respect to dimension. LIPO, AdaLIPO, and our enhanced versions are not suitable for maximizing high dimensional functions in a reasonable amount of time.

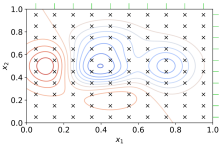
Note that the benchmark functions we used in this paper are fast to evaluate, which is not generally the case in the industry. This choice allowed us to perform a complete statistical analysis of the algorithms, which required a lot of function evaluations to be significant. However, these functions are as hard to optimize as most industrial objective functions (e.g. [13], [6], [15], [14]). Thus, our results are fully extendable to the latter case, without loss of generality.

In the future, it would be interesting to make a study of our versions of LIPO and AdaLIPO within a complex industrial framework in order to assess the performance of these algorithms in a more realistic problem. We also would like to develop a library containing efficient implementations of these algorithms for making them usable by as many people as possible, from all fields.

# Image Credits



MrMambo (Wikipedia)[2]



Alexander Elvers (Wikipedia), CC-BY-SA[3]

# References

[1] D. S. Arnon, G. E. Collins, and S. McCallum, *Cylindrical algebraic decomposition i: The basic algorithm*, SIAM Journal on Computing, 13 (1984), pp. 865–877. https://doi.org/10.1137/0213054.

[2] P. C. Gilmore and R. E. Gomory, *A linear programming approach to the cutting-stock problem*, Operations Research, 9 (1961), pp. 849–859. https://doi.org/10.1287/opre.9.6.849.

[2] https://commons.wikimedia.org/wiki/File:FAE_visualization.jpg

[3] https://commons.wikimedia.org/wiki/File:Hyperparameter_Optimization_using_Grid_Search.svg

[3] M. Girardi, C. Padovani, D. Pellegrini, and L. Robol, *A finite element model updating method based on global optimization*, Mechanical Systems and Signal Processing, 152 (2021), p. 107372.

[4] N. Hansen, *The CMA evolution strategy: A comparing review*, in Towards a New Evolutionary Computation, Springer Berlin Heidelberg, 2006, pp. 75–102. https://doi.org/10.1007/3-540-32494-1_4.

[5] W. K. Hastings, *Monte carlo sampling methods using markov chains and their applications*, Biometrika, 57 (1970), pp. 97–109. https://doi.org/10.1093/biomet/57.1.97.

[6] M. Hovd, *A brief introduction to model predictive control*, URL= http://www.itk. ntnu. no/fag/TTK4135/viktig/MPCkompendium% 20HOvd. pdf, (2004).

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing*, Science, 220 (1983), pp. 671–680. https://doi.org/10.1126/science.220.4598.671.

[8] A. H. Land and A. G. Doig, *An automatic method of solving discrete programming problems*, Econometrica, 28 (1960), p. 497. https://doi.org/10.2307/1910129.

[9] X. Luo, *Minima distribution for global optimization*, 2018. https://doi.org/10.48550/arxiv.1812.03457.

[10] C. Malherbe and N. Vayatis, *Global optimization of lipschitz functions*, 2017. https://doi.org/10.48550/arxiv.1703.02628.

[11] R. Martinez-Cantin, *Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits*, 2014. https://doi.org/10.48550/arxiv.1405.7430.

[12] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of state calculations by fast computing machines*, The Journal of Chemical Physics, 21 (1953), pp. 1087–1092. https://doi.org/10.1063/1.1699114.

[13] I. Papamichail and C. S. Adjiman, *Global optimization of dynamic systems*, Computers & chemical engineering, 28 (2004), pp. 403–415.

[14] K. E. Parsopoulos and M. N. Vrahatis, *Unified particle swarm optimization for solving constrained engineering optimization problems*, Lecture notes in computer science, 3612 (2005), p. 582.

[15] R. Z. Ríos-Mercado and C. Borraz-Sánchez, *Optimization problems in natural gas transportation systems: A state-of-the-art review*, Applied Energy, 147 (2015), pp. 536–555.

# A Algorithms

**Algorithm 1** LIPO's condition

**Require:**
$x$: the candidate point $X_{t+1}$
*values*: set of previous evaluated values $f(X_i)$
$k$: Lipschitz constant
*points*: set of previous point of evaluation $X_i$

max_val $\leftarrow max(values)$ $\triangleright$ $\max_{i=1\dots t} f(X_i)$
left_min $\leftarrow min(values + k \cdot \|x - points\|_2)$ $\triangleright$ $\min_{i=1\dots t} (f(X_i) + k \cdot \|X_{t+1} - X_i\|_2)$
**return** left_min $\geq$ max_val

**Algorithm 2** Stopping criterion for LIPO and AdaLIPO

**Require:**
$K$: the number of consecutive iterations to compute the slope with finite differences
*nb_samples*: a list of the number of samples on $K$ consecutive iterations
$\gamma$: threshold for the stopping criterion

slope $\leftarrow \frac{nb\_samples[K-1] - nb\_samples[0]}{K}$
**if** slope $> \gamma$ **then return** True
**else return** False
**end if**

**Algorithm 3** LIPO

**Require:**
$f$: the objective function
$n$: The maximum number of iteration

**Initialization**
$t \leftarrow 1$ ▷ iteration counter
nb_samples $\leftarrow 0$ ▷ sample counter to compute the difference between sampling and iterating
$X_1 \sim \mathcal{U}(f.bounds)$ ▷ sample from a uniform distribution over the compact set
points $\leftarrow [X_1]$ ▷ stored the points where $f$ is evaluated, needed for condition algorithm 1
values $\leftarrow [f(X_1)]$ ▷ stored the evaluations of $f$, needed for condition algorithm 1

**while** $t < n$ [and $\neg(2)$] **do** ▷ [optional stopping criterion].
  $X_{t+1} \sim \mathcal{U}(f.bounds)$
  **if** $condition(X_{t+1}, values, f.k, points)$ **then** ▷ apply the algorithm condition 1
    points.append($X_{t+1}$)
    values.append($f(X_{t+1})$) ▷ evaluate the function only if the condition is valid
    $t \leftarrow t + 1$ ▷ increase the iteration counter only when the function is evaluated
  **end if**
**end while**
**return** arg max(values) ▷ $\arg\max_{i=1\dots n} f(X_i)$

**Algorithm 4** AdaLIPO

**Require:**
$f$: The objective function
$n$: The maximum number of iteration
$p$: parameter of a Bernoulli distribution. A constant for AdaLIPO, a function of $t$ for AdaLIPO-E
$\alpha$: Needed to compute Eq. 7
**Initialization**
$t \leftarrow 1$ ▷ iteration counter
nb_samples $\leftarrow 0$ ▷ sample counter to compute the difference between sampling and iterating
$X_1 \sim \mathcal{U}(f.bounds)$ ▷ sample from a uniform distribution
$\hat{k} \leftarrow 0$ ▷ initialized the estimation of the Lipschitz constant to 0
nb_samples $\leftarrow$ nb_samples $+ 1$
points $\leftarrow [X_1]$ ▷ stored the points where $f$ is evaluated, needed for condition algorithm 1
value $\leftarrow f(X_1)$
values $\leftarrow [value]$ ▷ stored the evaluations of $f$, needed for condition algorithm 1

**while** $t < n$ **do**
  $B_{t+1} \sim \mathcal{B}(p)$
  **if** $B_{t+1} = 1$ **then**
    $X_{t+1} \sim \mathcal{U}(f.bounds)$
    nb_samples $\leftarrow$ nb_samples $+ 1$
    points.append($X_{t+1}$)
    value $\leftarrow f(X_{t+1})$
  **else**
    **while** $f$ is not evaluated [and $\neg(2)$] **do** ▷ [optional stopping criterion].
      **Apply LIPO with the estimated Lipschitz constant**
      $X_{t+1} \sim \mathcal{U}(f.bounds)$
      nb_samples $\leftarrow$ nb_samples $+ 1$
      **if** $condition(X_{t+1}, values, f.k, points)$ **then** ▷ Algo. 1
        points.append($X_{t+1}$)
        value $\leftarrow f(X_{t+1})$
      **end if**
    **end while**
  **end if**
  values.append(value)
  $t \leftarrow t + 1$ ▷ increase the iteration counter only when the function is evaluated

  **Update the estimated Lipschitz constant**
  ratios $\leftarrow [\,]$
  **for** $i = 0$ to $i = t$ **do**
    ratios.append$\left(\frac{|value - values[i]|}{\|X_{t+1} - points[i]\|_2}\right)$ ▷ apply Eq. 6
  **end for**
  $i_t \leftarrow \left\lceil \frac{\ln(\max_i \text{ratios})}{\ln(1+\alpha)} \right\rceil$
  $\hat{k} \leftarrow (1+\alpha)^{i_t}$ ▷ update the estimated Lipschitz as stated in Eq. 7
**end while**
**return** *points*, *values*, *nb_samples*