A Unifying Framework for Global Optimization: From Theory to Formalization

Gaëtan Serré, Argyris Kalogeratos, Nicolas Vayatis

A Unifying Framework for Global Optimization: From Theory to Formalization

Gaëtan Serré [1*], Argyris Kalogeratos [1] and Nicolas Vayatis [1]

[1]Centre Borelli, École Normale Supérieure Paris-Saclay, Gif-Sur-Yvette, 91190, France.

*Corresponding author(s). E-mail(s): gaetan.serre@ens-paris-saclay.fr;

Contributing authors: argyris.kalogeratos@ens-paris-saclay.fr ; nicolas.vayatis@ens-paris-saclay.fr ;

Abstract

We introduce an abstract measure-theoretic framework that serves as a tool to rigorously study stochastic iterative global optimization algorithms as a unified class. The framework is formulated in terms of probability kernels, which, via the Ionescu–Tulcea theorem, induce probability measures on the space of sequences of algorithm iterations, endowed with two intuitive properties. This framework answers the need for a general, implementation-independent formalism in the analysis of such algorithms, providing a starting point for formalizing global optimization results in proof-assistants. To illustrate the relevance of our tool, we show that common algorithms fit naturally in the framework, and we also use it to give a rigorous proof of a general consistency theorem for stochastic iterative global optimization algorithms (Proposition 3 of [34]). This proof and the entire framework are formalized in the `Lean` proof assistant. This formalization both ensures the correctness of the definitions and proofs, and provides a basis for future machine-assisted formalizations in the field.

Keywords: global optimization, Markov kernel, Ionescu-Tulcea, probability theory, stochastic process, formal proof, proof assistant, Lean

1 Introduction

Inspired by the broad adoption of optimization techniques across fast-growing areas such as machine learning, alongside the concurrent rise of proof assistants like Lean [38] and its Mathlib library [37], this work tackles a core gap at the intersection of algorithmic analysis and formal verification. Although many stochastic iterative global optimization algorithms have been developed and studied individually [4, 34, 40, 45], their systematic formalization in proof assistants remains scarce. While a general kernel-based perspective has appeared in the literature (e.g., [19]), existing approaches lack the development required for mechanized verification; specifically, they do not build probability laws over sequence spaces nor provide a proof-assistant formalization, both indispensable for machine-assisted verification. This fragmentation impedes the formalization of general results across families of algorithms, forcing each to be treated in isolation.

Our contribution resolves this obstacle by proposing an abstract, measure-theoretic framework that unifies the analysis of all stochastic iterative global optimization algorithms within a single kernel-based formalism. This unified approach enables formal assistants to verify general theorems that hold for entire classes of algorithms rather than single instances, substantially reducing redundant proof work. Crucially, once results are formalized within our framework in a system like `Lean`, they become reusable components for subsequent formal developments thanks to the inherent modularity of proof assistants. As a result, definitions and properties established within this framework are automatically available for reuse in future formalizations. By providing a rigorous, machine-verifiable foundation for optimization algorithms, this work harnesses the inherent modularity of proof assistants, transforming algorithmic verification from an ad hoc, labor-intensive undertaking into a cumulative practice where each formal contribution strengthens the shared knowledge base.

Global optimization is the problem of finding the global optimizer (maximizer or minimizer) of a real-valued function f over a search space Ω, usually a subset of $\mathbb{R}^d$. Global optimization problems can be classified into two main categories depending on the countability of Ω: if Ω is finite or countably infinite, the problem is called *discrete global optimization*; if Ω is uncountable, the problem is called *continuous global optimization*. The strategies used in these two categories are very different, and in this paper we focus on continuous global optimization. In this setting, the function f is usually assumed to be continuous, and the search space Ω is often assumed to be compact, which ensures that f has a global maximum and minimum within Ω. The problem of continuous global optimization can be formally expressed as follows:

$$x^\star \in \arg\max_{x \in \Omega} f(x).$$

The above is expressed using the arg max operator, though it is equivalent to using its complementary arg min: $\arg\max_{x \in \Omega} f(x) = \arg\min_{x \in \Omega} (-f(x))$.

The problem of global optimization is equivalent to that of local optimization only when f is a convex function, which explains why global optimization is often called *non-convex optimization*. The strategies employed to solve global optimization problems are different from those used to solve local optimization problems, and the theoretical analysis of global optimization algorithms can be more challenging, as the class of functions considered is much larger and the algorithms often explore stochastically the search space [8, 16, 25, 40, 45].

The applications of global optimization are vast, ranging from machine learning and data science to operations research and engineering design [1, 13, 15, 27, 33, 41]. Although the study of global optimization algorithms dates decades back [31, 43], it remains an active research area with many recent contributions, notably on SDE-driven methods [3, 14, 17, 40, 44].

The strategies developed to tackle global optimization problems are highly diverse. To cite a few examples, we can mention SIMULATED ANNEALING [31], which is inspired by the annealing process in metallurgy; *genetic algorithms* [26], which mimic the process of natural selection; *particle swarm optimization* [30], which is based on the social behavior of birds and fish; COVARIANCE MATRIX ADAPTATION EVOLUTION STRATEGY [25], which adapts the covariance matrix of a multivariate normal distribution to sample new candidate solutions. More recent algorithms include *bayesian optimization* [28, 46], which uses probabilistic models to guide the search for the optimum; ADALIPO [34], which combines Lipschitz optimization with random exploration; CONSENSUS-BASED OPTIMIZATION [40] or STEIN BOLTZMANN SAMPLING [45], which uses a system of interacting particles to explore the search space.

Proof assistants are software for formalizing and mechanically checking mathematical proofs. Proof assistants are built on extensions of the calculus of constructions [9], where propositions are represented as types and proofs as terms inhabiting those types. Thus, proving a statement in a proof assistant amounts to constructing a term of the corresponding type. Multiple proof assistants exist, such as `Rocq` [48], `Isabelle/HOL` [39], and `Lean` [38]. The use of proof assistants in foundational mathematics has been growing in recent years, with notable successes including the formalization of the four-color theorem [21] and the Feit-Thompson theorem [20] in `Rocq`; the formalization of the Kepler conjecture [23] in `Isabelle/HOL`; the formalization of the sphere eversion theorem [50] and the independence of the continuum hypothesis [24] in `Lean`; and more [2, 10, 51]. Recently, `Lean` has gained popularity due to its user-friendly interface and powerful libraries, such as `Mathlib` [37], which provides a comprehensive collection of undergraduate and graduate-level mathematics, including real analysis, measure theory, and topology. Renowned mathematicians such as Terence Tao have also embraced `Lean` for formalizing their research, including Tao's work on the Polynomial Freiman-Ruzsa conjecture [22] and his analysis textbook [47]. Tao has documented his experience with formalization in a series of blog posts[1,2]. In an online article[3], Terence Tao advocates a collaborative role for proof assistants and machine-learning

[1] https://terrytao.wordpress.com/2023/11/18/formalizing-the-proof-of-pfr-in-lean4-using-blueprint-a-short-tour/
[2] https://terrytao.wordpress.com/2025/05/31/a-lean-companion-to-analysis-i/
[3] https://terrytao.wordpress.com/wp-content/uploads/2024/03/machine-assisted-proof-notices.pdf

algorithms: by grounding ML outputs in formal verification, proof assistants could curb model hallucinations, while ML could suggest promising proof directions and accelerate the formalization process.

Although proof assistants have proven effective in foundational mathematics, their adoption in applied areas such as optimization remains limited. This stems from the complexity of formalizing algorithms and their analyses, the steep learning curve of proof assistants, and the limited overlap between formal-methods and applied-mathematics communities. Paradoxically, applied fields stand to gain the most from mechanized verification. In applied mathematics, where empirical results and algorithmic performance take precedence, proofs are secondary artifacts that rarely undergo the detailed peer review that is standard in pure mathematics. Undetected errors in such proofs can compromise entire lines of research. Formal verification thus addresses a critical gap, offering systematic validation where traditional review processes fall short.

Contributions. To this end, we set out to formally verify a non-trivial result from the global optimization literature using `Lean`. After surveying the field, we identified Proposition 3 from [34] as a suitable candidate. This proposition establishes, for **any** stochastic iterative global optimization algorithm, an equivalence between the consistency of a stochastic global optimization algorithm and the property of sampling the entire search space.

Proposition 1 (Proposition 3 from [34]) Let $\mathscr{A}$ be a stochastic iterative global optimization algorithm. The sequence $(X_0, \ldots, X_n)$ denotes a sequence of $n+1$ points sampled by $\mathscr{A}$ over a function f. The following statements are equivalent:

i. For any Lipschitz continuous function f defined on a compact measurable metric space Ω,
$$\sup_{x \in \Omega} \min_{i=0\ldots n} \operatorname{dist}(X_i, x) \xrightarrow{p} 0.$$

ii. For any Lipschitz continuous function f defined on a compact measurable metric space Ω,
$$\max_{i=0\ldots n} f(X_i) \xrightarrow{p} \max_{x \in \Omega} f(x).$$

Such general results are rare, and the proof is sufficiently intricate to make it an ideal test case for formal verification. However, the formalization effort revealed a fundamental obstacle: the absence of a rigorous, unifying definition of stochastic iterative global optimization algorithms. Given the diversity of approaches in the field, no general framework existed to support a systematic study, making it difficult to even state the proposition precisely. Some related works have started to explore this direction (e.g. [5, 7, 42]), but they are either limited to specific algorithm classes or lack the formalism needed for implementation-independent reasoning.

To overcome this challenge, we develop an abstract mathematical framework that serves as a tool to rigorously study any stochastic iterative global optimization algorithm, in a formal way. This framework is based on probability kernels, which are mathematical objects describing the probabilistic transitions between states in a stochastic process. The idea of using probability kernels has the advantage that we can capture the internal mechanisms of any stochastic iterative global optimization algorithm without explicitly modeling it. Our framework allows the construction of probability measures on the spaces of finite and infinite sequences of algorithm iterations, via the Ionescu–Tulcea theorem [49]. We prove two intuitive theorems about these measures that establish key properties useful for rigorous analysis of global optimization algorithms; they also act as sanity checks for the soundness of our framework and serve as exemplars of reasoning within it. Finally, we demonstrate how our framework enables a rigorous proof of Proposition 3 from [34], which we fully formalize in `Lean`, providing machine-assisted verification of both the framework and the proposition's proof. We emphasize that the general framework introduced in [19] is closely related in proposing a kernel-based formalism, but it does not construct probability laws on sequence spaces nor provide a formalization or concrete use case, which are key distinctions of our contribution.

Notations. Let $B(x, \varepsilon)$ be the open ball with center x and radius $\varepsilon > 0$ for some metric space. Let $\mathcal{P}(\Omega)$ be the set of probability measures over a measurable space $(\Omega, \mathscr{F})$. For a metric space specified by the context, we denote by $\mathrm{dist}(\cdot, \cdot)$ the distance function on this space. Depending on the context, the symbol "$\bigotimes$" denotes the product of σ-algebras, the product between probability kernels, or the product between measures and probability kernels. The former operation is defined as the σ-algebra generated by the Cartesian product of measurable sets from each σ-algebra. The second one is defined by:

$$(\kappa_1 \otimes \kappa_2)(x, A) = \int_y \kappa_2\left((x, y), \{z \mid (y, z) \in A\}\right) \mathrm{d}\kappa_1(x, y),$$

where κ_1 is a probability kernel from a measurable space (E_1) to another (E_2), κ_2 is a probability kernel from the product of these two measurable spaces $(E_1 \otimes E_2)$ to another (E_3), and A is a measurable set in the product measurable space $E_2 \otimes E_3$. The last operation, the product measures and probability kernels, is defined by:

$$(\mu \otimes \kappa)(A) = \int_x \kappa\left(x, \{y \mid (x, y) \in A\}\right) \mathrm{d}\mu(x),$$

where μ is a measure on a specific measurable space, κ is a probability kernel from this measurable space to another one, and A is a measurable set in the product measurable space. For more details on these operations, we refer the reader to [29]. Finally, we denote by $(X_i)_{i \in I}$ a sequence of points sampled by a stochastic iterative global optimization algorithm applied on a function (clear from the context), for some countable set I (finite or infinite).

2 A unifying framework

Given a continuous function $f : \Omega \to \beta$, any stochastic iterative global optimization algorithm $\mathscr{A}$ operates the same way. At first, it samples a point $X_0 \in \Omega$. This point is chosen according to the internal structure of the algorithm, potentially at random. Since f has not been evaluated yet, the distribution of X_0 is the same for any function. Then, the algorithm iteratively samples points X_n in Ω for $n > 0$. Apart from the structure of $\mathscr{A}$, the distribution of X_n only depends on the previous iterations and the evaluations of the function f at these points. Thus, $\mathscr{A}$ can be reduced to two elements in which the internal structure is encapsulated.

Definition 1 (Stochastic iterative global optimization algorithm) Given a measurable search space Ω and an evaluation space β (usually $\mathbb{R}$) associated with a measurable function $f : \Omega \to \beta$ to optimize, a stochastic iterative global optimization algorithm $\mathscr{A}$ is defined by:

i. an initial probability measure $\nu \in \mathcal{P}(\Omega)$ that describes the distribution of the first point X_0 sampled by the algorithm, and

ii. a sequence $(\kappa_n : \Omega^{n+1} \times \beta^{n+1} \rightsquigarrow \Omega)_{n\in\mathbb{N}}$, such that, for any $n \in \mathbb{N}$, κ_n is a probability kernel on the product space $\Omega^{n+1} \times \beta^{n+1}$ that describes the distribution of the next point X_{n+1} given the previous points $X_0, \ldots, X_n$ and their evaluations $f(X_0), \ldots, f(X_n)$. Given f, this corresponds to the conditional probability $\mathbb{P}(X_{n+1}|X_0, \ldots, X_n)$.

To assess the validity of this definition, we encompass several well-known algorithms in the literature into this framework.

Pure Random Search is one of the simplest stochastic iterative global optimization algorithm, which samples points uniformly at random in the search space Ω. Expressing this algorithm in the frame of Definition 1 is possible by an initial measure ν that is the uniform measure on Ω, and kernels κ_n that are the same for all n: they are the constant kernels that return the uniform measure on Ω for any previous points and evaluations.

AdaLIPO [34] is designed to find the global maximum of a real-valued Lipschitz function f. It uses an estimation of the Lipschitz constant of f to adaptively sample the search space. Expressing this algorithm in the frame of Definition 1 can be done by setting the initial measure ν to be the uniform measure on Ω, and the kernels κ_n to be defined as follows:

$$\kappa_n(e) = \mathcal{U}\left(\left\{x \in \Omega \;\middle|\; \max_{i=0\ldots n} f(X_i) \leq \min_{i=0\ldots n} f(X_i) + L(e) \cdot \mathrm{dist}(x, X_i)\right\}\right),$$

where $e \overset{\text{def}}{=} ((X_0, ..., X_n), (f(X_0), ..., f(X_n)))$, $L(e)$ is an estimation of the Lipschitz constant of f based on the previous evaluations, and $\mathcal{U}(\cdot)$ is the uniform measure. This kernel ensures that the next point X_{n+1} is sampled in a region where $f(X_{n+1})$ is above a linear upper bound based on the previous evaluations.

CMA-ES [25] is a well-known continuous global optimization algorithm. At each iteration, it samples k points according to a multivariate normal distribution where the mean and the covariance matrix depend on the previous iterations. Given a continuous function $f : \mathbb{R}^d \to \mathbb{R}$, to encompass this algorithm by Definition 1 we construct $f_k : \Omega^k \to \mathbb{R}^k$ by:

$$f_k(x_1, \ldots, x_k) = (f(x_1), \ldots, f(x_k)).$$

This new definition lets us view CMA-ES as a procedure that generates samples in $\mathbb{R}^{d\times k}$, where f_k is employed to construct the transition kernels.

The initial measure ν is then the k-product of multivariate normal distributions with mean m and covariance matrix C, two parameters of the algorithm. The kernels κ_n are defined as follows:

$$\kappa_n(e) = \prod_{i=1}^{k} \mathcal{N}(\mu(e), \Sigma(e)),$$

where $e \stackrel{\text{def}}{=} ((X_0, ..., X_n), (f_k(X_0), ..., f_k(X_n)))$ and where $\mu(e) \in \mathbb{R}^d$ and $\Sigma(e) \in \mathbb{R}^{d\times d}$ are functions of e that computes a mean vector and a covariance matrix of a multivariate normal distribution in $\mathbb{R}^d$. These functions use all the previous samples and evaluations; their definitions are intrinsic to CMA-ES and should not be interpreted as the empirical mean or covariance of e.

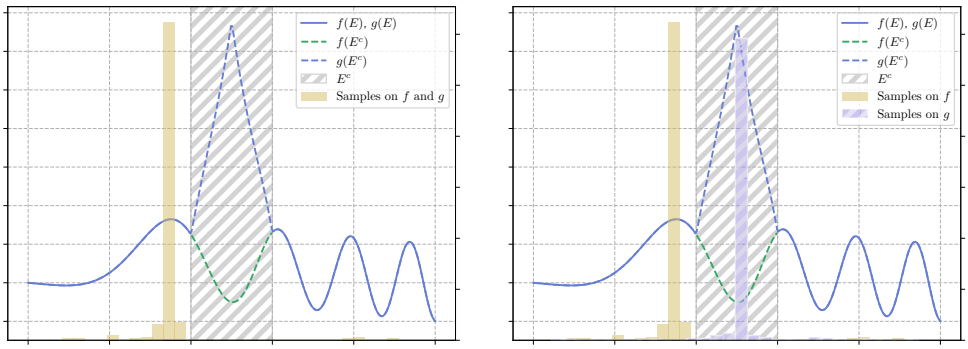
SDE-based algorithms. Easy to see that any global optimization algorithm that is based on a system of stochastic differential equations (e.g. [11, 40, 45, 52]), is trivially encompassed by the presented Definition 1: the initial measure ν is the initial distribution of the system and, for any $n \in \mathbb{N}$, the kernel κ_n is the transition kernel coming from the Markov process associated to the system (see [6, 32]). Moreover, if the step size of the numerical scheme used to solve the system is fixed (such as in the Unadjusted Langevin algorithm [12]), then the kernels κ_n are the same for any n.

2.1 Link with stochastic process

The foundational idea upon which this definition builds is that an iterative stochastic global optimization algorithm can be seen as a stochastic process. Indeed, provided the search space Ω is measurable, with $\mathscr{F}$ being the associated σ-algebra, one could construct the probability measure $\mathbb{P}$ on $\Omega^{\mathbb{N}}$ that describes the law of infinite sequences of points sampled by the algorithm. The iterations of the algorithm is then a collection of random variables, which can be written as

$$\left\{X_n : \Omega^{\mathbb{N}} \to \Omega \mid n \in \mathbb{N}\right\},$$

and can be seen as a stochastic process with respect to the probability space $\left(\Omega^{\mathbb{N}}, \mathscr{F}^{\otimes\mathbb{N}}, \mathbb{P}\right)$. However, constructing $\mathbb{P}$ (a measure on infinite sequences of Ω) solely from the internal structure of the algorithm (which only provides the distribution of the next point sampled by the algorithm given the previous points and their evaluations) is not straightforward. One way to proceed is to invoke the Ionescu–Tulcea theorem [49], which operates on a sequence of probability kernels, exactly the data provided by Definition 1.



(a) Samples restricted to lie inside E.

(b) No restriction on the samples.

Figure 1: Distribution of samples generated by CMA-ES, when optimizing two different functions f and g that are equal inside the set E (outside the highlighted area). In **(a)**, we force all samples to lie inside E, and we observe that the distributions of samples for both functions are identical. In **(b)**, we allow samples to lie anywhere in the search space, and we observe that the distributions of samples for both functions differ. This illustrates Theorem 1.

The construction of the measure on infinite sequences, denoted $\mu_\infty^{(f)}$ (where f is the function being optimized), is detailed in Section 4.1, where we show that it arises from a family of finite-dimensional laws $\left(\mu_n^{(f)}\right)_{n\in\mathbb{N}}$, each $\mu_n^{(f)}$ being a probability measure on the space of finite sequences of length $n+1$ generated by the algorithm. These measures are built from the initial distribution ν and the kernels $(\kappa_n)_{n\in\mathbb{N}}$ and satisfy two intuitive properties useful for rigorous analysis of global optimization algorithms.

2.2 Properties of the measures

In this section, we present two intuitive properties of the measures $\left(\mu_n^{(f)}\right)_{n\in\mathbb{N}}$ that arise from our framework. They are expected from the way stochastic iterative global optimization algorithms operate, yet their formal proofs require careful measure-theoretic reasoning. They act as sanity checks for the soundness of our framework, confirming that it aligns with intuitive expectations about algorithm behavior, and they also serve as exemplars of reasoning within it. The properties are needed in the formal proof of Proposition 1, which is detailed in Section 4.4.

2.2.1 Equality of restricted measures

Given two functions $f, g : \Omega \to \mathbb{R}$ that are equal on a set $E \subseteq \Omega$, a stochastic iterative global optimization algorithm $\mathscr{A}$ will behave identically if restricted to E. More precisely, the measures describing the sequences of points sampled by the algorithm for both functions coincide when we only consider sequences that lie entirely within E. This property is stated in the following theorem (see Figure 1 for an illustration), and its proof is provided in Section 4.3.

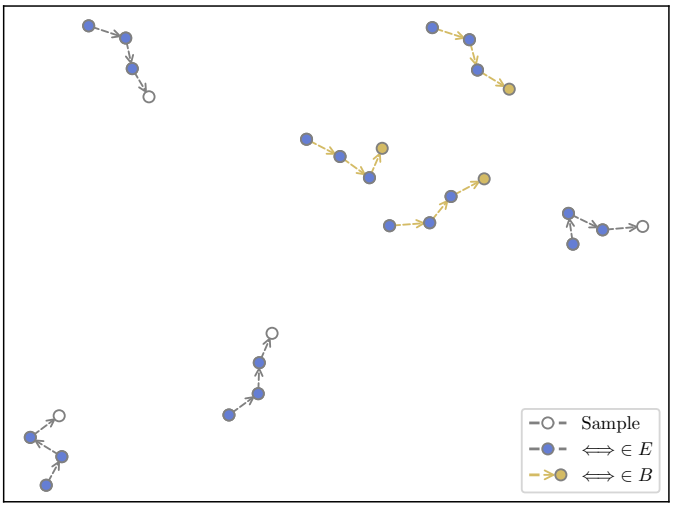


Figure 2: Symbolic illustration of sequences of length 4, representing algorithmic iterations starting from different initial conditions. For each sequence lying in (B connected by a yellow dashed line), its truncated sequence belongs to E (subsequence with blue dots), but the converse does not hold. As a result, there is at least as many sequences of length 3 in E as there are sequences of length 4 in B. This illustrates Theorem 2.

Theorem 1 (Equality of restricted measures) Let $\mathscr{A}$ be a stochastic iterative global optimization algorithm as in Definition 1, $E \subseteq \Omega$ a measurable set, $n \in \mathbb{N}$, and $f, g : \Omega \to \mathbb{R}$ two measurable functions such that $f(x) = g(x)$ for any $x \in E$. The following holds:

$$\mu_n^{(f)}|_{E^{n+1}} = \mu_n^{(g)}|_{E^{n+1}}.$$

2.2.2 Monotonicity by truncation

Another intuitive property is that, if an event defined on the first m iterations implies an event defined on the first n iterations (with $m \leq n$), then the probability of the former cannot exceed that of the latter. This intuitive monotonicity is stated in the following theorem (see Figure 2 for an illustration).

Theorem 2 (Monotonicity by sequence truncation) Let $\mathscr{A}$ be a stochastic iterative global optimization algorithm as in Definition 1, $f : \Omega \to \mathbb{R}$ a measurable function, $n, m \in \mathbb{N}$ such that $n \leq m$, a measurable set $E \subseteq \Omega^{n+1}$ and a measurable set $B \subseteq \Omega^{m+1}$. The following holds:

$$B \subseteq \left\{(X_i)_{0\leq i\leq m} \;\middle|\; (X_i)_{0\leq i\leq n} \in E\right\} \implies \mu_m^{(f)}(B) \leq \mu_n^{(f)}(E).$$

The proof is provided in Section 4.2.

3 Lean formalization

The entire framework presented in this paper (including Definition 1 and Theorems 1 and 2) has been fully formalized in the `Lean` proof assistant [38]. To provide intuition for how to use the framework, we also implement two algorithms as instances of our abstract definition. The framework formalization and these illustrative instances are available in the `LeanGO` repository[4] with documentation[5]. Building on LeanGO, the proof of Proposition 1 is formalized in the `LipoCons` repository[6], with documentation[7].

3.1 Overview of Lean

`Lean` [38] is an open-source proof assistant based on dependent type theory. It is a programming language with a strong type system that allows users to define mathematical objects, state theorems, and construct proofs in a formal language that can be checked by a computer. It is based on the *calculus of inductive constructions* [9], which consider propositions as types and proofs as terms inhabiting those types. Thus, proving a statement in `Lean` reduces to constructing a term of the corresponding type. To help users constructing such terms, `Lean` provides a "tactic mode", where users can apply high-level commands (called tactics) to manipulate the current proof state. Tactics correspond to common proof techniques in mathematics, such as induction, case analysis, and contradiction. To prove complex mathematical statements, users often combine `Lean` with the `Mathlib` library [37], which provides a comprehensive collection of definitions, theorems, and proofs in various areas of mathematics, including real analysis, measure theory, and topology.

To give a sense of how `Lean` code looks like, we provide a simple example below. The following `Lean` code states and proves that the function $x : \mathbb{R} \mapsto x + 1$ is strictly monotone:

```
1 import Mathlib
2 example : StrictMono (fun (x : ℝ) ↦ x + 1) := by
3   intro x y hxy
4   /-
5     x y : ℝ
6     hxy : x < y
7     ⊢ x + 1 < y + 1
8   -/
9   exact (add_lt_add_iff_right 1).mpr hxy
```

Let us briefly explain this code. The first line uses the `import` command to import the definitions of real numbers and of strict monotonicity from `Mathlib`. The second line states the example, where `StrictMono` is the proposition that a function is strictly monotone. The third line begins the proof, where `intro` is a tactic that introduces two arbitrary real numbers x and y such that $x < y$. The proof state is then updated

[4] https://github.com/gaetanserre/LeanGO
[5] https://gaetanserre.fr/LeanGO/
[6] https://github.com/gaetanserre/LipoCons
[7] https://gaetanserre.fr/LipoCons/

to show that $x + 1 < y + 1$. The fourth line uses a lemma from `Mathlib` that states that, for any real numbers x, y, a, $x + a < y + a \iff x < y$.

3.2 Formalization of the proposed framework

The first step of our formalization is to implement the framework presented in Definition 1. To do so, we need to have `Lean` definitions of measurable spaces, measures, and kernels. While these definitions are already available in `Mathlib`, we detail them here for completeness.

Measurable space. A measurable space is a set X equipped with a σ-algebra $\mathscr{F}$, which is a collection of subsets of X that includes the empty set, is closed under complements, and is closed under countable unions. In `Mathlib`, a measurable space is defined as *typeclass*, which is a way to attach additional structure to a type. The `Mathlib` definition of a measurable space is as follows:

```
class MeasurableSpace (X : Type) where
  /-- Predicate saying that a given set is measurable. -/
  MeasurableSet' : Set X → Prop
  -- For a set E ⊆ X, MeasurableSet' E is the proposition "E is measurable"

  /-- The empty set is a measurable set. -/
  measurableSet_empty : MeasurableSet' ∅
  /-- The complement of a measurable set is a measurable set. -/
  measurableSet_compl : ∀ s, MeasurableSet' s → MeasurableSet' sᶜ
  /-- The union of a sequence of measurable sets is a measurable set. -/
  measurableSet_iUnion : ∀ (f : ℕ → Set X),
    (∀ i, MeasurableSet' (f i)) → MeasurableSet' (⋃ i, f i)
```

This typeclass attaches to a type X (which corresponds to a set in classical mathematics) a σ-algebra by providing a predicate `MeasurableSet'` that determines whether a subset of X is measurable. It also provides the necessary properties to ensure that `MeasurableSet'` defines a σ-algebra.

Measure. A measure on a measurable space $(X, \mathscr{F})$ is a function $\mu : \mathscr{F} \to \overline{\mathbb{R}}_+$ that assigns a non-negative extended real number to each measurable set, satisfying the properties of non-negativity, null empty set, and countable additivity. In `Mathlib`, a measure is defined as a *structure*, which is a way to group related data together (such as a function and its properties). The `Mathlib` definition of a measure is rather involved: it is defined as an outer measure (a function from sets to $\overline{\mathbb{R}}_+$ that sends $\varnothing$ to 0 and is countably subadditive) that is countably additive on measurable sets. The `Mathlib` definition of a measure is as follows:

```
structure Measure (X : Type*) [MeasurableSpace X] extends OuterMeasure X where
  -- σ-additivity
  m_iUnion ⦃f : ℕ → Set X⦄ : (∀ i, MeasurableSet (f i)) → Pairwise (Disjoint on f) →
    toOuterMeasure (⋃ i, f i) = ∑' i, toOuterMeasure (f i)
  -- The outer measure is the canonical extension of its restriction to measurable sets
  trim_le : toOuterMeasure.trim ≤ toOuterMeasure
```

This structure defines a new type "`Measure` X" for measures on a measurable space X. It can be used to create an object μ that represents a measure on X:

```
variable {X : Type} [MeasurableSpace X] (μ : Measure X)
```

Kernel. A kernel from a measurable space $(X, \mathscr{F})$ to another measurable space $(Y, \mathscr{G})$ is a function $\kappa : X \times \mathscr{G} \to \overline{\mathbb{R}}_+$ such that, **i)** for each fixed $x \in X$, $\kappa(x, \cdot)$ is a measure on $(Y, \mathscr{G})$, and **ii)** for each fixed $A \in \mathscr{G}$, $\kappa(\cdot, A)$ is a measurable function on $(X, \mathscr{F})$. In `Mathlib`, a kernel is also defined as a *structure*. The `Mathlib` definition of a kernel is as follows:

```
structure Kernel (X Y : Type*) [MeasurableSpace X] [MeasurableSpace Y] where
  /-- The underlying function of a kernel. -/
  toFun : X → Measure Y
  /-- A kernel is a measurable map. -/
  measurable' : Measurable toFun
```

Note that in this definition, **ii)** is expressed using the fact that the space of measures on $(Y, \mathscr{G})$ is itself a measurable space [18].

Formalization of Definition 1. With these definitions in place, we can now formalize Definition 1 in `Lean`. A stochastic iterative global optimization algorithm is represented as a `Lean` structure that encapsulates the initial measure and the sequence of kernels:

```
structure Algorithm (Ω β : Type*) [MeasurableSpace Ω] [MeasurableSpace β] where
  v : Measure Ω
  prob_measure : IsProbabilityMeasure v
  kernel_iter (n : ℕ) : Kernel (prod_iter_image Ω β n) Ω
  markov_kernel (n : ℕ) : IsMarkovKernel (kernel_iter n)
```

Here, Ω is the search space, β is the evaluation space (most likely $\mathbb{R}$), and ν is the initial measure. The instances `prop_measure` and `markov_kernel` ensure that ν is a probability measure and that, for any $n \in \mathbb{N}$, `kernel_iter n` is a probability kernel.

To define the probability measure $\mu_\infty^{(f)}$ of an `Algorithm`, we used the existing `Lean` formalization of the Ionescu-Tulcea theorem [36]:

```
def measure (hf : Measurable f) (A : Algorithm Ω β) : Measure (ℕ → Ω) :=
  (Kernel.traj (A.iter_comap hf) 0).avg A.v_mequiv
```

We do not go into the details of this `Lean` definition, but it essentially corresponds to the measure constructed using the Ionescu-Tulcea theorem, where `Kernel.traj` is the infinite product of the algorithm's kernels, and where `iter_comap` is a technical construction that adapts the kernels to the function f being optimized (see Section 4.1).

Then, we define the finite-dimensional measures $\left(\mu_n^{(f)}\right)_{n \in \mathbb{N}}$ by pushing forward $\mu_\infty^{(f)}$ along the function that truncates an infinite sequence to its first $n+1$ elements:

```
def fin_measure {n : ℕ} (hf : Measurable f) (A : Algorithm Ω β) : Measure (iter Ω n) :=
  (A.measure hf).map (frestrictLe n)
```

where `iter Ω n` is the Lean type that represents the product space Ω^{n+1}, and where `frestrictLe n` is the truncation function.

Some examples. To give a sense of how these definitions look like in practice, we provide the Lean formalization of the PURE RANDOM SEARCH (PRS) and the LIPO [34] algorithms. The implementation of these algorithms is available in the `LeanGO` repository[8].

PRS. The initial measure ν of PRS is the uniform measure on Ω, and the kernels κ_n are the constant kernels that return the uniform measure on Ω for any previous points and evaluations:

```
def PRS : Algorithm α β where
  ν := uniform univ
  prob_measure := uniform_is_prob_measure (by simp [NeZero.ne ℙ])
  kernel_iter _ := Kernel.const _ (uniform (univ) : Measure α)
  markov_kernel _ := ⟨fun _ ⇒ uniform_is_prob_measure (by simp [NeZero.ne ℙ])⟩
```

where Ω is a measure space with finite and non-zero measure (e.g. a compact subset of $\mathbb{R}^d$), and where `uniform` is the function that returns the uniform measure on a given set.

LIPO. The initial measure ν of LIPO is also the uniform measure on Ω, but the kernels κ_n are defined as follows:

$$\kappa_n(e) = \mathcal{U}\left(\left\{x \in \Omega \;\middle|\; \max_{i=0\ldots n} f(X_i) \leq \min_{i=0\ldots n} f(X_i) + \kappa \cdot \mathrm{dist}(x, X_i)\right\}\right),$$

where $e \overset{\text{def}}{=} ((X_0, ..., X_n), (f(X_0), ..., f(X_n)))$ and κ is the Lipschitz constant of f. The Lean formalization of LIPO is more involved as PRS because of the definition of the kernels, but it follows the same structure as the formalization of PRS:

```
def LIPO : Algorithm Ω ℝ where
  ν := uniform Set.univ
  prob_measure := by
    have := i, mes_Ω mΩ,
    refine uniform_is_prob_measure ?_
    rwa [Measure.Subtype.volume_univ mes_Ω.nullMeasurableSet]
  kernel_iter _ := potential_max_kernel mes_Ω mΩ, κ
  markov_kernel n := by
    refine ⟨fun data ⇒ ?_⟩
    have := i, mes_Ω mΩ,
    exact uniform_is_prob_measure ◁ h n data
```

where Ω is a subset of $\mathbb{R}^d$ with finite and non-zero measure, κ is a positive real number, and where `potential_max_kernel` is the function that returns the kernel defined by the formula above. We advise the reader to consult the Lean code for the precise

[8] https://github.com/gaetanserre/LeanGO

definition of `potential_max_kernel`, as it involves several technical constructions to ensure that it is indeed a kernel.

Formalization of the properties. Finally, we formalized and proved Theorem 1 and Theorem 2 in `Lean`, both of which were essential for establishing Proposition 1.

The `Lean` statement of Theorem 2 is as follows:

```
theorem fin_measure_mono {n m : ℕ} {s : Set (iter Ω n)} (hs : MeasurableSet s)
    {e : Set (iter Ω m)} (he : MeasurableSet e) (hmn : n ≤ m)
    (hse : e ⊆ {u | subTuple hmn u ∈ s}) {f : Ω → β} (hf : Measurable f) :
    A.fin_measure hf e ≤ A.fin_measure hf s := by ...
```

The `Lean` proof essentially follows the same reasoning as the proof presented in Section 4.2.

The `Lean` statement of Theorem 1 is as follows:

```
theorem eq_restrict {f g : Ω → β} (hf : Measurable f) (hg : Measurable g)
    {s : Set Ω} (hs : MeasurableSet s) (h : s.EqOn f g) (n : ℕ) :
    (A.fin_measure hf).restrict (univ.pi (fun (_ : Finset.Iic n) ⇒ s)) =
    (A.fin_measure hg).restrict (univ.pi (fun (_ : Finset.Iic n) ⇒ s)) := by ..
```

Analogously, the `Lean` proof mirrors the proof presented in Section 4.3, although the decomposition of the product of kernels is carried out with considerably greater technical precision.

3.3 Formalization of Proposition 1

We do not delve into the details of the formalization of the equivalent properties in Proposition 1. However, we provide a detailed sketch of the proof in Section 4.4, that can be followed alongside the `Lean` formalization. The `Lean` statement of Proposition 1 is as follows:

```
variable {Ω : Type*} [MeasurableSpace Ω] [NormedAddCommGroup Ω] [NormedSpace ℝ Ω]
  [CompactSpace Ω] [Nonempty Ω] [OpensMeasurableSpace Ω]

theorem sample_iff_consistent (A : Algorithm Ω ℝ) :
    (∀ {f : Ω → ℝ}, (hf : Lipschitz f) → sample_whole_space A hf.continuous)
    ↔
    (∀ {f : Ω → ℝ}, (hf : Lipschitz f) → is_consistent A hf) := by ...
```

In this statement, `sample_whole_space` corresponds to property **(i)** of Proposition 1, while `is_consistent` corresponds to property **(ii)**. The convergence in probability is expressed using `fin_measure`, defined in the previous section.

One notable aspect of this formalization is the use of various typeclasses to represent the mathematical structures involved. The search space Ω is equipped with several typeclasses to ensure it has the necessary properties (measurable space, normed space, compactness, etc.). This approach allows us to work in a general setting without committing to a specific type for Ω.

4 Formal constructions and proofs

This section details the construction of the infinite-sequence measure $\mu_\infty^{(f)}$ introduced in Section 2, gives rigorous proofs of Theorem 1 and Theorem 2, and closes with a sketch of the proof of Proposition 1.

4.1 Construction of finite and infinite dimensional laws

We construct probability measures on finite and infinite sequences sampled by an algorithm $\mathscr{A}$ as in Definition 1. To do so, we invoke the Ionescu–Tulcea theorem [49], which allows us to build such measures from the initial distribution ν and the sequence of kernels $(\kappa_n)_{n\in\mathbb{N}}$.

Theorem 3 (Ionescu-Tulcea theorem [49]) Let $(\Omega_0, \mathcal{A}_0, P_0)$ be a probability space and $(\Omega_n, \mathcal{A}_n)_{n\in\mathbb{N}}$ a sequence of measurable spaces. For each $n \in \mathbb{N}^*$, let $\kappa_n : \left(\Omega^{n-1}, \mathcal{A}^{n-1}\right) \rightsquigarrow (\Omega_n, \mathcal{A}_n)$ be a probability kernel, where:

$$\Omega^n \overset{\text{def}}{=} \prod_{k=0}^{n} \Omega_k \quad \text{and} \quad \mathcal{A}^n \overset{\text{def}}{=} \bigotimes_{k=0}^{n} \mathcal{A}_k.$$

Then, for each $n \in \mathbb{N}$, there exists a probability measure P_n defined on $(\Omega^n, \mathcal{A}^n)$ by the following formula:

$$P_n \overset{\text{def}}{=} P_0 \otimes \bigotimes_{k=1}^{n} \kappa_k,$$

and there exists a uniquely defined probability measure $\mathbb{P}$ on $\left(\Omega^{\mathbb{N}}, \bigotimes_{k=1}^{\infty} \mathcal{A}_k\right)$, so that

$$P_n(A) = \mathbb{P}\left(A \times \prod_{k=n+1}^{\infty} \Omega_k\right).$$

Thus, the measure $\mathbb{P}$ has conditional probabilities equal to the stochastic kernels κ.

In Definition 1 we are given the initial law of X_0 and kernels κ_n defined on $\Omega^{n+1}\times\mathbb{R}^{n+1}$. The Ionescu–Tulcea theorem, however, requires kernels on Ω^{n+1}. To bridge this gap we precompose each κ_n along the evaluation map

$$e_f : \Omega^{n+1} \to \Omega^{n+1} \times \mathbb{R}^{n+1}, \qquad e_f(x_0, \ldots, x_n) = ((x_0, \ldots, x_n), (f(x_0), \ldots, f(x_n))),$$

obtaining pullback kernels on Ω^{n+1} which are then used in the Ionescu–Tulcea construction.

Definition 2 (Pullback kernels on finite sequences) Given a topological space Ω endowed with a σ-algebra $\mathscr{F}$, a natural number n, and a measurable function $f : \Omega \to \mathbb{R}$. We define $\kappa_n'^{(f)} : \Omega^{n+1} \rightsquigarrow \Omega$ as the pullback of κ_n along e_f, i.e., for any $x \in \Omega^{n+1}$,

$$\kappa_n'^{(f)}(x, \cdot) \overset{\text{def}}{=} \kappa_n(e_f(x), \cdot).$$

These new kernels enable the construction of probability measures on finite sequences of points sampled by an algorithm driven by a target function f. Indeed, for any $n \in \mathbb{N}$, a probability measure $\mu_n^{(f)}$ (the notation here emphasizes the dependence on f) can be defined on $\left(\Omega^{n+1}, \bigotimes_{k=0}^{n} \mathscr{F}\right)$ by:

$$\mu_n^{(f)} \stackrel{\text{def}}{=} \nu \otimes \bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}.$$

Moreover, using Theorem 3, we can obtain $\mu_\infty^{(f)}$, which is the probability measure on infinite sequences of points sampled by the algorithm over the function f such that:

$$\forall n \in \mathbb{N}, A \in \bigotimes_{k=0}^{n} \mathscr{F}, \mu_n^{(f)}(A) = \mu_\infty^{(f)}\left(A \times \prod_{k=n+2}^{\infty} \Omega\right). \tag{1}$$

Therefore, each measure $\mu_n^{(f)}$ is the restriction of $\mu_\infty^{(f)}$ to the first $n+1$ elements of infinite sequences.

4.2 Proof of Theorem 2

Proof. The proof is trivial using Equation (1). We have that:

$$\begin{aligned}\mu_m^{(f)}(B) &= \mu_\infty^{(f)}\left(B \times \prod_{k=m+1}^{\infty} \Omega\right) \\ &= \mu_\infty^{(f)}\left(\{(X_i)_{i\in\mathbb{N}} \mid (X_i)_{0\leq i\leq m} \in B\}\right).\end{aligned}$$

Moreover, for any $X \in \Omega^{\mathbb{N}}$ such that $(X_i)_{0\leq i\leq m} \in B$, we have by hypothesis that $(X_i)_{0\leq i\leq n} \in E$. Thus,

$$\{(X_i)_{i\in\mathbb{N}} \mid (X_i)_{0\leq i\leq m} \in B\} \subseteq \{(X_i)_{i\in\mathbb{N}} \mid (X_i)_{0\leq i\leq n} \in E\},$$

and therefore

$$\begin{aligned}\mu_\infty^{(f)}\left(\{(X_i)_{i\in\mathbb{N}} \mid (X_i)_{0\leq i\leq m} \in B\}\right) &\leq \mu_\infty^{(f)}\left(\{(X_i)_{i\in\mathbb{N}} \mid (X_i)_{0\leq i\leq n} \in E\}\right) \\ &= \mu_n^{(f)}(E).\end{aligned}$$

□

4.3 Proof of Theorem 1

Proof.
$\mathbf{n = 0}$. We have that $\mu_0^{(f)} = \nu = \mu_0^{(g)}$, and thus the property holds.

$\mathbf{n \geq 1}$. It is sufficient to show that the two measures are equal on all measurable rectangles of the form $\prod_{i=0}^{n} B_i$ where $(B_i)_{0\leq i\leq n}$ are measurable subsets of Ω. Indeed, the

σ-algebra $\bigotimes_{i=0}^{n} \mathscr{F}$ is generated by these rectangles. Moreover, these rectangles form a π-system. Thus, the Sierpiński–Dynkin's π-λ theorem ensures that two measures are equal on this σ-algebra if and only if they are equal on all these rectangles.

Let $(B_i)_{0\leq i\leq n}$ be measurable subsets of Ω. The goal reduces to show that

$$\begin{aligned}\mu_n^{(f)}|_{E^{n+1}}\left(\prod_{i=0}^{n} B_i\right) &= \mu_n^{(g)}|_{E^{n+1}}\left(\prod_{i=0}^{n} B_i\right)\\ \mu_n^{(f)}\left(\prod_{i=0}^{n} B_i \cap E\right) &= \mu_n^{(g)}\left(\prod_{i=0}^{n} B_i \cap E\right).\end{aligned}$$

For any $i \in [\![0,n]\!]$, let $C_i \overset{\text{def}}{=} B_i \cap E$. Using the definition of $\mu_n^{(f)}$, we have that

$$\begin{aligned}\mu_n^{(f)}\left(\prod_{i=0}^{n} B_i \cap E\right) &= \left(\nu \otimes \bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}\right)\left(\prod_{i=0}^{n} C_i\right)\\
&= \int_x \bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}\left(x, \left\{(y_i)_{1\leq i\leq n} \;\middle|\; (x,y) \in \prod_{i=0}^{n} C_i\right\}\right) \mathrm{d}\nu(x)\\
&= \int_{x\in C_0} \bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}\left(x, \left\{(y_i)_{1\leq i\leq n} \;\middle|\; (x,y) \in \prod_{i=0}^{n} C_i\right\}\right) \mathrm{d}\nu(x)\\
&\quad + \int_{x\notin C_0} \bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}\left(x, \left\{(y_i)_{1\leq i\leq n} \;\middle|\; (x,y) \in \prod_{i=0}^{n} C_i\right\}\right) \mathrm{d}\nu(x)\\
&= \int_{x\in C_0} \bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}\left(x, \left\{(y_i)_{1\leq i\leq n} \;\middle|\; (x,y) \in \prod_{i=0}^{n} C_i\right\}\right) \mathrm{d}\nu(x)\\
&\quad + \int_{x\notin C_0} \bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}\left(x, \varnothing\right) \mathrm{d}\nu(x)\\
&= \int_{x\in C_0} \bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}\left(x, \left\{(y_i)_{1\leq i\leq n} \;\middle|\; (x,y) \in \prod_{i=0}^{n} C_i\right\}\right) \mathrm{d}\nu(x) + 0\end{aligned}$$

and similarly,

$$\mu_n^{(g)}\left(\prod_{i=0}^{n} B_i \cap E\right) = \int_{x\in C_0} \bigotimes_{k=0}^{n-1} \kappa_k'^{(g)}\left(x, \left\{(y_i)_{1\leq i\leq n} \;\middle|\; (x,y) \in \prod_{i=0}^{n} C_i\right\}\right) \mathrm{d}\nu(x).$$

Thus, it is sufficient to show that the integrands are equal for any $x \in C_0$. Let $A_n \overset{\text{def}}{=} \prod_{i=1}^{n} C_i$. For any $x \in C_0$, We have that

$$\left\{ (y_i)_{1 \leq i \leq n} \;\middle|\; (x, y) \in \prod_{i=0}^{n} C_i \right\} = A_n. \tag{2}$$

It is sufficient to show that the restriction to A_n of these two kernels products are equal, i.e.

$$\left(\bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}(x, \cdot) \right) \Bigg|_{A_n} = \left(\bigotimes_{k=0}^{n-1} \kappa_k'^{(g)}(x, \cdot) \right) \Bigg|_{A_n}.$$

Indeed, if this holds, then

$$\bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}(x, A_n) = \bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}(x, A_n).$$

Combining this with Equation (2), we obtain the desired result.

We proceed by induction on n. For $n = 1$, we have that

$$\kappa_0'^{(f)}(x, \cdot)|_{A_1} = \kappa_0(f(x), \cdot)|_{A_1} \text{ and } \kappa_0'^{(g)}(x, \cdot)|_{A_1} = \kappa_0(g(x), \cdot)|_{A_1}$$

As $C_0 \subseteq E$ and f and g agree on E, we have that $f(x) = g(x)$, and thus

$$\kappa_0'^{(f)}(x, \cdot)|_{A_1} = \kappa_0'^{(g)}(x, \cdot)|_{A_1}.$$

Now, let $n \in \mathbb{N}$ such that the property holds. We want to show that it also holds for $n + 1$. For any measurable set $B \subseteq \Omega^{n+1}$, we have that

$$\begin{aligned}
&\left(\bigotimes_{k=0}^{n} \kappa_k'^{(f)}(x, B) \right) \Bigg|_{A_{n+1}} \\
&\qquad = \int_y \kappa_n'^{(f)}((x, y), \{z \mid (y, z) \in B\})|_{C_{n+1}} \, \mathrm{d} \left(\bigotimes_{k=0}^{n-1} \kappa_k'^{(f)}(x, y) \right) \Bigg|_{A_n} \\
&\qquad = \int_{y \in A_n} \kappa_n'^{(f)}((x, y), \{z \mid (y, z) \in B\})|_{C_{n+1}} \, \mathrm{d} \left(\bigotimes_{k=0}^{n-1} \kappa_k'^{(g)}(x, y) \right) \Bigg|_{A_n}.
\end{aligned}$$

Moreover, as $x \in C_0$, for any $y \in A_n$, we have that $(x, y) \in E^{n+1}$, and thus $f((x, y)) = g((x, y))$. Therefore,

$$\kappa_n'^{(f)}((x, y), \{z \mid (y, z) \in B\})|_{C_{n+1}} = \kappa_n'^{(g)}((x, y), \{z \mid (y, z) \in B\})|_{C_{n+1}},$$

which concludes the induction and the proof. □

4.4 Proof sketch of Proposition 1

In this section, we briefly discuss the strategy of the proof of Proposition 1 developed in the preprint [35] of the original paper [34], considering that stochastic and iterative global optimization algorithms are defined as in Definition 1. We do not detail the entire proof, which is already well explained in that paper, but rather highlight the key steps where the framework developed in the present work is essential.

Modus ponens. The right implication of Proposition 1 is quite intuitive. Indeed, if the algorithm samples the search space, then it means that, for any point in this space, the probability for a point sampled by the algorithm to not be arbitrarily close to this point goes to 0. Therefore, for any $\varepsilon > 0$, it suffices to select a $\delta > 0$ such that, for any point x in the search space such that the distance between x and the maximizer of f (noted $x^\star$) is lower than δ, then $f(x^\star) - f(x) \leq \varepsilon$. This δ can be obtained using the continuity of f. Thus, for any $n \in \mathbb{N}$, we have the following set inclusion:

$$\left\{(X_i)_{0\leq i\leq n} \;\middle|\; f(x^\star) - \max_{i=0\dots n} f(X_i) > \varepsilon\right\} \subseteq \left\{(X_i)_{0\leq i\leq n} \;\middle|\; \sup_{x\in\Omega} \min_{i=0\dots n} \operatorname{dist}(X_i, x) > \delta\right\}.$$

This allows us to show that, for any $n \in \mathbb{N}$,

$$\begin{aligned}\mu_n^{(f)}\left(\left\{(X_i)_{0\leq i\leq n} \;\middle|\; f(x^\star) - \max_{i=0\dots n} f(X_i) > \varepsilon\right\}\right) \leq \\ \mu_n^{(f)}\left(\left\{(X_i)_{0\leq i\leq n} \;\middle|\; \sup_{x\in\Omega} \min_{i=0\dots n} \operatorname{dist}(X_i, x) > \delta\right\}\right),\end{aligned}$$

by the monotonicity of measures. Then, we can conclude that the left-hand side also goes to 0 as n goes to infinity using the squeeze theorem, as we know by hypothesis that

$$\mu_n^{(f)}\left(\left\{(X_i)_{0\leq i\leq n} \;\middle|\; \sup_{x\in\Omega} \min_{i=0\dots n} \operatorname{dist}(X_i, x) > \delta\right\}\right) \xrightarrow[n\to\infty]{} 0.$$

Reverse modus ponens. The left implication is more involved and is proven by contradiction, i.e. we prove that the facts that the algorithm is consistent on any Lipschitz function and that there exists a Lipschitz function f and an $\varepsilon_1 > 0$ such that

$$\exists \varepsilon_2 > 0, \forall N \in \mathbb{N}, \exists n \geq N, \varepsilon_2 < \mu_n^{(f)}\left(\left\{(X_i)_{0\leq i\leq n} \;\middle|\; \sup_{x\in\Omega} \min_{i=0\dots n} \operatorname{dist}(X_i, x) > \varepsilon_1\right\}\right)$$

are incompatible.

First, we use the fact that Ω is a compact space to construct an $(\varepsilon_1/2)$-cover of Ω, i.e. a finite collection of points $\{c_1, \dots, c_{N_1}\}$ in Ω such that, $\bigcup_{i=1\dots N_1} B(c_i, \varepsilon_1/2) = \Omega$.

Next, we use the contradiction hypothesis, Theorem 2, and the monotonicity of measures to show that there exists a $c^\star \in \{c_1, \dots, c_{N_1}\}$ such that, for any $n \in \mathbb{N}$,

$$0 < \frac{\varepsilon_2}{2N_1} \leq \mu_n^{(f)}\left(\{(X_i)_{0\leq i\leq n} \mid \forall i \in [\![0, n]\!], X_i \notin B(c^\star, \varepsilon_1/2)\}\right). \tag{3}$$

Proof. To do so, we again reason by contradiction, assuming that for any c_i in the cover, there exists an $n_i \in \mathbb{N}$ such that

$$\mu_{n_i}^{(f)}\left(\{(X_j)_{0\leq j\leq n_i} \mid \forall j \in [\![0, n_i]\!], X_j \notin B(c_i, \varepsilon_1/2)\}\right) < \frac{\varepsilon_2}{2N_1}. \tag{4}$$

Then, we select $n^\star$, the maximum of all these n_i's. Using Theorem 2, we show that, for any c_i in the cover,

$$\mu_{n^\star}^{(f)}\left(\{(X_j)_{0\leq j\leq n^\star} \mid \forall j \in [\![0, n^\star]\!], X_j \notin B(c_i, \varepsilon_1/2)\}\right) < \frac{\varepsilon_2}{2N_1}.$$

Indeed, as, for any c_i, there exists a $n_i \leq n^\star$ such that Equation (4) holds and as the following set inclusion holds:

$$\begin{aligned}\{(X_j)_{0\leq j\leq n^\star} \mid \forall j \in [\![0, n^\star]\!], X_j \notin & B(c_i, \varepsilon_1/2)\} \subseteq \\ & \{(X_j)_{0\leq j\leq n_i} \mid \forall j \in [\![0, n_i]\!], X_j \notin B(c_i, \varepsilon_1/2)\},\end{aligned}$$

we can use Theorem 2 to show that, for any c_i in the cover,

$$\mu_{n^\star}^{(f)}(\{(X_j)_{0\leq j\leq n^\star} \mid \forall j \in [\![0, n^\star]\!], X_j \notin B(c_i, \varepsilon_1/2)\}) < \frac{\varepsilon_2}{2N_1}. \tag{5}$$

Finally, showing that

$$\begin{aligned}\left\{(X_i)_{0\leq i\leq n^\star} \;\middle|\; \sup_{x\in\Omega} \min_{i=0\ldots n^\star} \mathrm{dist}(X_i, x) > \varepsilon_1\right\} \subseteq & \\ \bigcup_{i=1\ldots N_1} \{(X_j)_{0\leq j\leq n^\star} \mid \forall j \in [\![0, n^\star]\!], X_j \notin & B(c_i, \varepsilon_1/2)\}\end{aligned}$$

and using Equation (5) with the σ-additivity of measures, we obtain the contradiction:

$$\mu_{n^\star}^{(f)}\left(\left\{(X_i)_{0\leq i\leq n^\star} \;\middle|\; \sup_{x\in\Omega} \min_{i=0\ldots n^\star} \mathrm{dist}(X_i, x) > \varepsilon_1\right\}\right) \leq \varepsilon_2.$$

□

Next, we construct a Lipschitz function $\tilde{f}$ that is indistinguishable from f outside the ball $B(c^\star, \varepsilon_1/2)$ and such that its maximum is strictly greater than the maximum of f over Ω. This function is constructed as follows:

$$\tilde{f}(x) = \begin{cases} f(x) + 2\left(1 - \frac{\mathrm{dist}(c^\star, x)}{\varepsilon_1/2}\right) \times \left(\max_{x\in\Omega} f(x) - \min_{x\in\Omega} f(x) + 1\right) & \text{if } x \in B(c^\star, \varepsilon_1/2); \\ f(x) & \text{otherwise.} \end{cases}$$

Note that our expression for $\tilde{f}$ is slightly simpler than the one in [34], while serving the same purpose, and it also corrects an issue in the original proof, which fails when f is constant. A visual representation of this function is given in Figure 3.

Finally, let $\delta \overset{\text{def}}{=} \max_{x\in\Omega} \tilde{f}(x) - \max_{x\in\Omega} f(x) > 0$. By construction of $\tilde{f}$, for any sequence of points $(X_i)_{0\leq i\leq n}$ such that, for any $i \in [\![0, n]\!]$, $X_i \notin B(c^\star, \varepsilon_1/2)$ implies that

$$\max_{x\in\Omega} \tilde{f}(x) - \max_{i=0\ldots n} \tilde{f}(X_i) > \delta.$$

Therefore, using Equation (3) and Theorem 1, we have that

$$0 < \frac{\varepsilon_2}{2N_1} \leq \mu_n^{(\tilde{f})}\left(\left\{(X_i)_{0\leq i\leq n} \;\middle|\; \max_{x\in\Omega} \tilde{f}(x) - \max_{i=0\ldots n} \tilde{f}(X_i) > \delta\right\}\right),$$

and thus

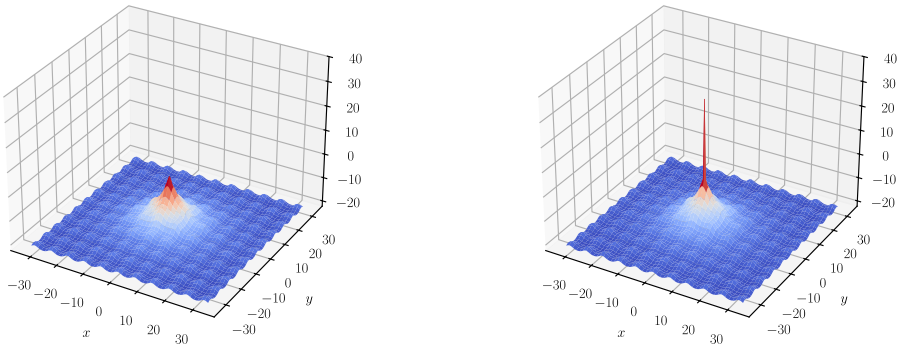
$$\mu_n^{(\tilde{f})}\left(\left\{(X_i)_{0\leq i\leq n} \middle| \max_{x\in\Omega} \tilde{f}(x) - \max_{i=0\ldots n} \tilde{f}(X_i) > \delta\right\}\right) \xrightarrow[n\to\infty]{\not\longrightarrow} 0,$$

which contradicts the fact that the algorithm is consistent on $\tilde{f}$.

The central role of our framework is evident in this proof, as Theorem 1 and Theorem 2 are indispensable tools for the formal verification of this result.

5 Conclusion

This work presents an abstract, measure-theoretic framework that serves as a practical tool for reasoning about stochastic iterative global optimization algorithms. Built



(a) The reverse Ackley function.

(b) $\tilde{f}$ for the reverse Ackley function.

Figure 3: Visualization of $\tilde{f}$ applied to the reverse Ackley function with $c \overset{\text{def}}{=} (0, 0)^\top$ and $\varepsilon_1 \overset{\text{def}}{=} 1$.

solely from an initial law and a sequence of probability kernels, the framework leverages the Ionescu–Tulcea theorem to produce probability measures on finite and infinite iteration sequences and yields useful and intuitive properties. This framework answers the need for a general, implementation-independent formalism in the analysis of such algorithms, providing the foundation for formalizing general results in proof assistants.

To demonstrate its practical applicability, we showed that common algorithms (e.g., ADALIPO, CMA-ES, SDE-based methods) fit naturally within the framework, and we used it to establish a clear, implementation-independent proof of a general consistency theorem (Proposition 3 of [34]). Critically, the entire framework and the proof of the proposition were formalized in the `Lean` proof assistant, ensuring the correctness of the definitions and proofs. This formalization effort demonstrates that rigorous verification of optimization theory is achievable and establishes a reusable foundation for future developments in the field.

By providing machine-verified building blocks, this work enables the inherent modularity of proof assistants in the optimization domain: definitions and theorems formalized here can be directly leveraged in subsequent formalizations, transforming algorithmic verification from isolated, ad hoc efforts into cumulative contributions that strengthen the collective infrastructure. Future work includes formalizing additional algorithms as concrete instances of the framework, establishing convergence rates and other algorithmic properties, and extending the framework to handle stochastic optimization in more general settings, such as unbounded domains or non-compact search spaces.

Acknowledgment

We thank Étienne Marion for his assistance with the formalization of Theorem 1 and for his work on the formalization of the Ionescu–Tulcea theorem in Mathlib, as well as all members of the `Lean` Zulip chat[9] for their support. The authors acknowledge the support from the Industrial Data Analytics and Machine Learning Chair hosted at ENS Paris-Saclay.

References

[1] Pranjal Awasthi, Anqi Mao, Mehryar Mohri, and Yutao Zhong. DC-programming for neural network optimizations. *Journal of Global Optimization*, 2024.

[2] Maxwell P. Bobbin, Samiha Sharlin, Parivash Feyzishendi, An Hong Dang, Catherine M Wraback, and Tyler R Josephson. Formalizing chemical physics using the Lean theorem prover. *Digital Discovery*, 2024.

[3] Leon Bungert, Tim Roith, and Philipp Wacker. Polarized consensus-based dynamics for optimization and sampling. *Mathematical Programming*, 211, 2024.

[9] https://leanprover.zulipchat.com/

[4] José A Carrillo, Shi Jin, Lei Li, and Yuhua Zhu. A consensus-based global optimization method for high dimensional machine learning problems. *ESAIM: Control, Optimisation and Calculus of Variations*, 2021.

[5] Philippe Casgrain. A latent variational framework for stochastic optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[6] Louis-Pierre Chaintron and Antoine Diez. Propagation of chaos: A review of models, methods and applications. I. Models and methods. *Kinetic and Related Models*, 2022.

[7] Chi-Ning Chou, Juspreet Singh Sandhu, Mien Brabeeba Wang, and Tiancheng Yu. A General Framework for Analyzing Stochastic Dynamics in Learning Algorithms. *preprint arXiv:2006.06171*, 2020.

[8] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013.

[9] Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 1988.

[10] Viviana del Barco, Gustavo Infanti, Exequiel Rivas, and Paul Schwahn. Formalizing a classification theorem for low-dimensional solvable Lie algebras in Lean. In *International Conference on Intelligent Computer Mathematics*, 2025.

[11] Aymeric Dieuleveut, Alain Durmus, and Francis Bach. Bridging the gap between constant step size stochastic gradient descent and Markov chains. *The Annals of Statistics*, 2020.

[12] Alain Durmus and Éric Moulines. Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. *The Annals of Applied Probability*, 2017.

[13] Meryeme El Yadari, Saloua El Motaki, Ali Yahyaouy, Philippe Makany, Khalid El Fazazy, Hamid Gualous, and Stéphane Le Masson. Taxonomy of optimization algorithms combined with CNN for optimal placement of virtual machines within physical machines in data centers. *Energy Informatics*, 2024.

[14] Massimo Fornasier, Timo Klock, and Konstantin Riedl. Consensus-based optimization methods converge globally. *SIAM Journal on Optimization*, 2024.

[15] Luca Franceschi, Michele Donini, Valerio Perrone, Aaron Klein, Cédric Archambeau, Matthias Seeger, Massimiliano Pontil, and Paolo Frasconi. Hyperparameter optimization in machine learning. *Preprint arXiv:2410.22854*, 2024.

[16] Peter I. Frazier. A Tutorial on Bayesian Optimization, 2018.

[17] Nicolai Jurek Gerber, Franca Hoffmann, and Urbain Vaes. Mean-field limits for consensus-based optimization and sampling. *ESAIM: Control, Optimisation and Calculus of Variations*, 31:74, 2025.

[18] Michèle Giry. *A categorical approach to probability theory*. 1982.

[19] Jonatan Gomez. Stochastic global optimization algorithms: A systematic formal approach. *Information Sciences*, 2019.

[20] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, et al. A machine-checked proof of the odd order theorem. In *International Conference on Interactive Theorem Proving*, pages 163–179, 2013.

[21] Georges Gonthier et al. Formal proof–the four-color theorem. *Notices of the AMS*, 2008.

[22] William Timothy Gowers, Ben Green, Freddie Manners, and Terence Tao. On a conjecture of Marton. *Annals of Mathematics*, 201(2):515–549, 2025.

[23] Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, et al. A formal proof of the Kepler conjecture. In *Forum of Mathematics, Pi*, volume 5, page e2, 2017.

[24] Jesse Michael Han and Floris van Doorn. A formal proof of the independence of the continuum hypothesis. In *ACM SIGPLAN International Conference on Certified Programs and Proofs*, 2020.

[25] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *IEEE International Conference on Evolutionary Computation*, 1996.

[26] John H. Holland. Genetic Algorithms and Adaptation. *Adaptive Control of Ill-Defined Systems*, 1984.

[27] Essam H. Houssein, Mahmoud Khalaf Saeed, Gang Hu, and Mustafa M. Al-Sayed. Metaheuristics for Solving Global and Engineering Optimization Problems: Review, Applications, Open Issues and Challenges. *Archives of Computational Methods in Engineering*, 2024.

[28] Donald R. Jones. *Direct Global Optimization Algorithm*. 2001.

[29] Olav Kallenberg. *Foundations of Modern Probability*. Probability Theory and Stochastic Modelling. 2021.

[30] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks*, 1995.

[31] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by Simulated Annealing. *Science*, 1983.

[32] Jean-François Le Gall. *Brownian Motion, Martingales, and Stochastic Calculus*. Graduate Texts in Mathematics. 2016.

[33] Juyong Lee, In-Ho Lee, InSuk Joung, Jooyoung Lee, and Bernard R Brooks. Finding multiple reaction pathways via global optimization of action. *Nature Communications*, 2017.

[34] Cédric Malherbe and Nicolas Vayatis. Global optimization of Lipschitz functions. In *International Conference on Machine Learning*, 2017.

[35] Cédric Malherbe and Nicolas Vayatis. Global optimization of Lipschitz functions, 2017.

[36] Etienne Marion. A Formalization of the Ionescu-Tulcea Theorem in Mathlib. *preprint arXiv:2506.18616*, 2025.

[37] The mathlib Community. The lean mathematical library. In *ACM SIGPLAN International Conference on Certified Programs and Proofs*, 2020.

[38] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *International Conference on Automated Deduction*, 2021.

[39] Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. Isabelle/HOL: a proof assistant for higher-order logic. 2002.

[40] René Pinnau, Claudia Totzeck, Oliver Tse, and Stephan Martin. A consensus-based model for global optimization and its mean-field limit. *Mathematical Models and Methods in Applied Sciences*, 2017.

[41] János D. Pintér. Global Optimization in Action. *Nonconvex Optimization and Its Applications*, 1996.

[42] Warren B Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 2019.

[43] W. L. Price. Global optimization by controlled random search. *Journal of Optimization Theory and Applications*, 1983.

[44] Konstantin Riedl. Leveraging memory effects and gradient information in consensus-based optimisation: On global convergence in mean-field law. *European Journal of Applied Mathematics*, 2024.

[45] Gaëtan Serré, Argyris Kalogeratos, and Nicolas Vayatis. Stein Boltzmann Sampling: A Variational Approach for Global Optimization. In *International*

Conference on Artificial Intelligence and Statistics, 2025.

[46] Bobak Shahriari, Alexandre Bouchard-Cote, and Nando Freitas. Unbounded Bayesian Optimization via Regularization. In *International Conference on Artificial Intelligence and Statistics*, 2016.

[47] Terence Tao. Analysis I. *Texts and Readings in Mathematics*, 2022.

[48] The Coq Development Team. The Coq Proof Assistant, 2024.

[49] CT Ionescu Tulcea. Mesures dans les espaces produits. *Atti Accad. Naz. Lincei Rend*, 1949.

[50] Floris van Doorn, Patrick Massot, and Oliver Nash. Formalising the h-principle and sphere eversion. In *ACM SIGPLAN International Conference on Certified Programs and Proofs*, 2023.

[51] Eric Wieser and Utensil Song. Formalizing geometric algebra in lean. *Advances in Applied Clifford Algebras*, 2022.

[52] Pan Xu, Jinghui Chen, Difan Zou, and Quanquan Gu. Global convergence of Langevin dynamics based algorithms for nonconvex optimization. *Advances in Neural Information Processing Systems*, 2018.