In Proceedings of the 12th International Workshop on Foundations of Genetic Algorithms (FOGA-2013) 2013. ACM Press.

A Measure-Theoretic Analysis of Stochastic Optimization

Alan J. Lockett
Department of Computer Science
Univers ty of Texas
Austin, TX, USA
alockett@cs.utexas.edu

Risto Miikkulainen
Department of Computer Science
University of Texas
Austin, TX, USA
risto@cs.utexas.edu

ABSTRACT

This paper proposes a measure-theoretic framework to study iterative stochastic optimizers that provides theoretical tools to explore how the optimization methods may be improved. Within this framework, optimizers form a closed, convex subset of a normed vector space, implying the existence of a distance metric between any two optimizers and a meaningful and computable spectrum of new optimizers between them. It is shown how the formalism applies to evolutionary algorithms in general. The analytic property of continuity is studied in the context of genetic algorithms, revealing the conditions under which approximations such as meta-modeling or surrogate methods may be effective. These results demonstrate the power of the proposed analytic framework, which can be used to propose and analyze new techniques such as controlled convex combinations of optimizers, meta-optimization of algorithm parameters, and more.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.m [**Artificial Intelligence**]: Miscellaneous—*evolutionary computation, theory of optimization*

Keywords

Evolutionary computation, genetic algorithms, stochastic optimization, theory of genetic algorithms, optimizer space, population-based optimizers, functional analysis

1. INTRODUCTION AND MOTIVATION

Although particular evolutionary algorithms have been studied formally in great detail, there have been relatively few attempts to provide a unified framework that would allow meaningful formal comparisons between different evolutionary techniques, leading to improved methods. The work of Vose on Random Heuristic Search (RHS) is an important exception [17, 18], but it is limited to finite, discrete spaces, is primarily algebraic in nature, and is most commonly applied to genetic algorithms. This paper outlines a new formal approach to the study of optimization algorithms more generally as mathematical objects. It accommodates arbitrary Borel-measurable spaces and applies to all iterative optimization algorithms, although the particular focus in this paper is on evolutionary algorithms optimizing static (e.g. non-dynamic, deterministic) fitness functions.

This formalization leads to several interesting conclusions. First, in the proposed analysis, a variety of algebraic operations are defined that can be used to create new algorithms; these operators also characterize the space of optimization algorithms as an infinite-dimensional vector space, complete with formal definitions of distance and continuity. Second, the framework subsumes almost all evolutionary algorithms that are commonly used for optimization; it will be demonstrated how several evolutionary algorithms can be instantiated within the framework. Third, within this framework, it is possible to prove that techniques such as meta-modeling and fitness surrogates [4, 7] should perform well for most optimizers, since most of the common evolutionary algorithms are continuous almost surely, a fundamental result that means that approximations of fitness functions or optimizers behave like the approximated quantities. This paper addresses each of these three topics in sequence below.

2. THE OPTIMIZATION PROBLEM

An optimization problem will be denoted formally as a tuple (f, X), where f is a fitness function and X is the (potentially constrained) search domain over which f is to be optimized. Without loss of generality, optimization is assumed to be synonymous with minimization; a function f can be maximized by minimizing its additive inverse $-f$. This study makes the following assumptions about the nature of the search space and the fitness function:

The search space is a topological space (X, τ) where X is the collection of possible solutions and τ is a given topology [12]. Topologies are mainly used to reason about issues such as continuity, limits, and nearness without reference to distance metrics.

In addition, the search space is a measurable space $(X, \mathcal{B}_\tau)$, where $\mathcal{B}_\tau$ is the Borel σ-algebra on the topology τ. A σ-algebra on a space X is a set of subsets of X that can be measured [2, 3, 6]. That is, they preserve certain intuitive notions about volume or area. For instance, if any two subsets of X can be measured, then so can their union, intersection, and complements. The Borel σ-algebra is the

FOGA'13, January 16-20, 2013, Adelaide, Australia.

smallest σ-algebra under which the open and closed sets of the topology are measurable.

These requirements are quite broad and accommodate all familiar spaces on which optimization is performed, including binary strings, real vectors, neural networks, graphs, state machines, and programs.

The objective function is drawn from the space of real functions on X, denoted $\mathbb{R}^X$. The topology of pointwise convergence is assumed for this function space. Under this topology, a set of functions $\{f_n\}$ converges to a function f if and only if $f_n(x) \to f(x)$ for all $x \in X$. When a σ-algebra on $\mathbb{R}$ is required, the standard Borel σ-algebra for the Euclidean topology is assumed [2, 6].

The formalization below is based on measure theory. A *measure* is a set function (usually nonnegative) that assigns a volume to each set in a σ-algebra. A *probability measure* is a measure μ that additionally has $\mu(X) = 1$. Lebesgue integration over a real function with respect to a measure sums up the measure of the level sets of the function. A function f is measurable if the sets $\{x : f(x) \leq y\}$ are contained in $\mathcal{B}_\tau$ for all $y \in \mathbb{R}$. It is integrable on a measure μ if $\int_X |f| \, d\mu < \infty$ [3, 6]. For a given measure μ, the search space is a measure space, written as $(X, \mathcal{B}_\tau, \mu)$. In Euclidean space, μ is assumed to be the Lebesgue measure, the familiar measure of volume.

With this background, the space of iterative stochastic optimizers can now be defined.

3. OPTIMIZER SPACE

In order to state theorems that can encompass the vast majority of optimization methods, these methods must belong to a common mathematical space. Such a space can be found by examining the defining characteristic of iterative optimizers, namely, the iterative sequence of points they produce.

3.1 Stochastic Iterative Optimizers

At its core, an iterative optimizer follows a trajectory through the search domain. At each point in time, it observes the fitness value of each point along the trajectory so far and uses this information to propose one or more points to add to the trajectory, with the goal that later points in the trajectory should have better fitness. The internal structure of the fitness function is known only to a limited extent. Thus the iterative optimizer must propose new points to evaluate based only upon the current trajectory and its fitness evaluations. The distinguishing feature of a particular method is the mechanism that it uses in order to propose new points (e.g. how it creates new populations). Most non-analytic optimization methods follow an iterative process matching this general description.

Optimization has been studied from this perspective before in the context of No Free Lunch (NFL) theorems. Radcliffe and Surry presented such a formalization in 1995 [13], followed by Wolpert and Macready in 1997 [19]. Similar formalizations have been used broadly since that time [16, 1, 14], but always with the goal of studying NFL. With some modifications and generalizations, however, the same techniques can be used on its own in order to study theoretical questions about optimization methods.

Formally, recall that the fitness functions are drawn from the space $\mathbb{R}^X$, where X is the search domain. Let $\mathcal{T}[X]$ be the space of finite sequences with arbitrary length on X. Elements of $\mathcal{T}[X]$ will be referred to as *trajectories* or *evaluation histories*, and the entries in a trajectory will be indexed with superscripts, i.e. t^i is the i^{th} entry in the sequence t.[1] Negative superscripts will be used to index the trajectory from right to left, e.g. t^{-1} is the last element of the trajectory t. The notation $|t|$ will be used to refer to the length of the trajectory t, and $\emptyset$ will refer to the trajectory of length zero. Let $\mathcal{P}[X] = \mathcal{P}[X, \mathcal{B}_\tau]$ be the space of probability measures on the search domain X that are measurable with respect to the Borel σ-algebra $\mathcal{B}_\tau$.

Definition 1. A *stochastic iterative optimizer* is a function $\mathcal{G} : \mathcal{T}[X] \times \mathbb{R}^X \to \mathcal{P}[X]$. Let the space of all such optimizers be denoted by $\mathcal{PF} = \mathcal{PF}[X]$ (an abbreviation for *probability-valued functions*).

To make the meaning of these formal objects clear, consider how the optimizer $\mathcal{G}$ is applied to an example optimization problem (f, X). For concreteness, suppose that $\mathcal{G}$ implements a $(1+1)$–ES, and let the search domain be the real line, $X = \mathbb{R}$. Let the $(1+1)$–ES use Gaussian mutation with a fixed variance σ^2. The first step is to sample an initial point z_1 by sampling $z_1 \sim \mathcal{G}[\emptyset, f]$. In this case $\mathcal{G}[\emptyset, f]$ is an initial distribution, for example, a standard Gaussian. Then $f(z_1)$ is observed, and a new point z_2 is sampled from $\mathcal{G}[(z_1), f]$. In particular, $\mathcal{G}[(z_1), f]$ is a Gaussian centered at z_1 with variance σ^2. The process continues iteratively, and new points $z_3, z_4, z_5, \ldots$ are sampled one at a time, with $z_n \sim \mathcal{G}[(z_1, \ldots, z_{n-1}), f]$. For the $(1+1)$–ES, $\mathcal{G}[(z_1, \ldots, z_{n-1}), f]$ is a Gaussian distribution with variance σ^2 centered at

$$\text{argmax}_{x \in \{z_1, \ldots, z_{n-1}\}} f(x). \tag{1}$$

The discussion so far raises two questions about the selected formalism. First, why is the optimizer passed a full copy of the fitness function f, when in fact the $(1+1)$–ES depends only on the fitness evaluation of the points $z_1, z_2, \ldots$? Other similar formulations (e.g. [13, 19]) define an optimization algorithm to depend only on the value of the fitness function at the evaluation points thus far, e.g. $f(z_1), f(z_2), \ldots$. The more general definition here accounts for optimization methods that might look at features of the fitness evaluation other than just the fitness value, such as the derivatives of the fitness function, or statistics gathered during a simulation. Some evolutionary algorithms, such as Novelty Search [9], do indeed use such features to guide their exploration of the search domain.

The second question is how this formalization can account for evolutionary methods that use population sizes larger than one; this question is addressed in the next section.

3.2 Population-Based Optimizers

Evolutionary methods generate entire populations, not individual points, so the choice in the previous section to use probability distribution over individual points may seem unusual. However, each population is nothing more than a collection of points. A population can be generated by sampling the new set of points one at a time with the appropriate dependency relationships. It is entirely correct to approach evolutionary algorithms from a probabilistic perspective, since the underlying probability distribution governing these methods is nothing other than the mechanism

[1]The subscript notation, e.g. t_n, is reserved for sequences of trajectories, used in Section 8 to obtain continuity results.

by which the next population is created from previous populations with random variations.

A population-based optimizer $\mathcal{G}$ with population size $K > 0$ can be represented as a sequence of K separate optimizers $\mathcal{G}_1, \cdots, \mathcal{G}_K$ (not necessarily distinct), each of which is used to generate one individual per population. A trajectory can then be broken up into populations, with one optimizer assigned to each slot in the population. A population-based optimizer is given by $\mathcal{G}[t, f] = \mathcal{G}_{k(t)}[t, f]$, where $k(t) \equiv 1 + (|t| \mod K)$ is the index of the individual in the population currently being constructed. The function $k(t)$ will be used repeatedly below.

A population may be treated as an element in the product space X^K consisting of K copies of the search domain X. A trajectory $t \in \mathcal{T}[X]$ can be broken up into a history of populations $H = h^1, h^2, h^3, \cdots$ with $h^i \in X^K$ using the mapping $h^{i,k} = t^{(i-1)K+k+1}$, recalling that trajectories are indexed with superscripts. Let $H(t)$ be the history of complete populations in the trajectory t, so that $H(t)$ ignores any elements in t with index greater than $\lfloor |t|/K \rfloor$. Then $H(t)$ is a trajectory over populations, i.e. $H(t) \in \mathcal{T}\left[X^K\right]$. To complete the setup, let $\mathrm{traj}(H)$ convert a population history $H \in \mathcal{T}\left[X^K\right]$ to a trajectory in $\mathcal{T}[X]$ via the mapping $\mathrm{traj}(H)^j = H^{\lfloor j/K \rfloor, 1+(j \mod K)}$. Then $\mathrm{traj}(H(t)) = t$ if and only if the length of t is a multiple of K, i.e. $|t| = K\lfloor |t|/K \rfloor$; otherwise, it truncates the end of t at the last population boundary. The notation $\mathcal{G}[H(t), f]$ may be used to mean $\mathcal{G}[\mathrm{traj}(H(t)), f]$ when this notation is clear from the context.

A population-based optimizer is distinguished by the fact that it respects the population boundary, and new populations can only be generated based on information available from prior populations.

Definition 2. An optimizer $\mathcal{G} \in \mathcal{PF}$ is a *population-based optimizer* with population size K if $\mathcal{G}[t_1, f] = \mathcal{G}[t_2, f]$ whenever $f(t_1^{i_1}) = f(t_2^{i_2})$ for all $1 \le i_j \le |t_j| - k(t_j)$, where $k(t)$ is computed for population size K.

That is, to be a population-based optimizer, an optimizer must be able to evaluate points in parallel. The space of all population-based optimizers of population size K will be denoted by $\mathcal{PBO}_K$.

The simplest way for an optimizer to be population-based is if the optimizer does not depend on the objective at all.

Definition 3. An optimizer $\mathcal{G} \in \mathcal{PF}$ is *objective-agnostic* if $\mathcal{G}[t, f] = \mathcal{G}[t, g]$ for all $f, g \in \mathbb{R}^X$.

Most algorithms in evolutionary and natural computation depend on the objective only through the fitness evaluations along the evaluation history, i.e. on $f(x)$ for $x \in t$. Such an algorithms will be termed *trajectory-restricted*.

Definition 4. An optimizer $\mathcal{G} \in \mathcal{PF}$ is *trajectory-restricted* if $\mathcal{G}[t, f] = \mathcal{G}[t, g]$ whenever $f(x) = g(x)$ for all $x \in t$.

The set of trajectory-restricted optimizers corresponds directly to the set of optimizers usually studied for the NFL theorems [19, 14]. Most evolutionary algorithms are trajectory-restricted population-based optimizers. The following example will illustrate these concepts.

3.3 Example: Simulated Annealing

Simulated annealing [8] is a widely used optimization method that can be formalized as a population-based optimizer with population size 2. At each non-initial time step for simulated annealing, there is an accepted solution x, and a new solution y is proposed. The objective value $f(y)$ is computed, and y replaces x as the accepted solution with probability

$$A(y, x, f, T) = \exp\left(\frac{1}{T}\left[f(x) - f(y)\right]\right) \wedge 1, \qquad (2)$$

where T is the temperature parameter of simulated annealing and the infix operator $\wedge$ indicates the minimum of its arguments, so that y is always accepted if $f(y) < f(x)$. Clearly, simulated annealing is trajectory-restricted. For this formalization, the population for simulated annealing consists of the accepted and proposed solutions.

Simulated annealing generates the proposed solution from a space-dependent *proposal distribution*. For this example, let the search domain be the real line, $X = \mathbb{R}$, with a multivariate Gaussian as the proposal distribution, $\mathcal{N}\langle \mu, \sigma^2 \rangle$, for some fixed positive number $\sigma \in \mathbb{R}$. Let the trajectory t store the accepted solution and the proposed solution in alternation, so that each point t^n in the trajectory is the accepted solution at the $\lfloor \frac{n}{2} \rfloor^{th}$ time step if n is odd, and the proposed solution if n is even. Then set $\mu = t^{-1}$, the last accepted solution in the trajectory. Then the proposal distribution is

$$\mathcal{P}[t, f] = \mathcal{N} < t^{-1}, I > . \qquad (3)$$

Given a proposed y and an accepted solution x, simulated annealing performs a Bernoulli trial to determine whether to accept y or keep x. Let $\mathcal{B}\langle p, y, x \rangle$ be a Bernoulli distribution that produces y with probability p and x with probability $1 - p$. The acceptance step for simulated annealing is then

$$\mathcal{A}[t, f] = \mathcal{B}\big\langle A(t^{-1}, t^{-2}, f, T(|t|/2)), t^{-1}, t^{-2} \big\rangle , \qquad (4)$$

recalling that t^{-1} contains the proposal and t^{-2} the accepted solution. The temperature $T(n)$ is assumed to be a function of the length of the trajectory, commonly $T(n) = 1/\log n$.

Simulated annealing can thus be viewed as a population-based optimizer $\mathcal{SA}$ of size 2 with $\mathcal{SA}_1 = \mathcal{A}$ and $\mathcal{SA}_2 = \mathcal{P}$. The starting population (t_1, t_2) is initialized randomly, and thenceforth $\mathcal{SA}_1$ and $\mathcal{SA}_2$ are used in alternation to accept and propose solutions.

The profusion of symbols in this example may seem unnecessary at first. However, the formalism makes it possible to compare simulated annealing directly with other optimization methods in a way that pseudocode does not allow. For instance, the classic $(1+1)$–ES is formally the norm-limit of $\mathcal{SA}$ as the temperature schedule goes to zero.

In addition, the simulated annealing example makes use of several components and techniques that will be used in defining evolutionary algorithms. The proposal distribution $\mathcal{P}$ plays the role of a mutation operator in evolutionary methods, randomly altering a previously evaluated point. The acceptance optimizer $\mathcal{A}$ mirrors the role of selection in evolutionary methods. The formal elements of evolutionary algorithms will be explored further in Section 6, after some required notation is introduced in Section 3.4.

3.4 Notation and Conventions

This subsection completes the notation and conventions that will be used throughout the subsequent text. Op-

timizers will typically be denoted by capital cursive letters, usually by $\mathcal{G}$. The expression $\mathcal{G}[t, f]$ will be used to refer to the probability measure corresponding to a trajectory $t \in \mathcal{T}[X]$ and a fitness function $f \in \mathbb{R}^X$. Accordingly, $\mathcal{G}[t, f](A)$ indicates the probability that the next point will lie inside of a set A contained in the σ-algebra $\mathcal{B}_\tau$. The notation $\mathcal{G}[t, f](dx)$ represents a quantity that can be integrated over A in the Lebesgue sense to obtain $\mathcal{G}[t, f](A) = \int_A \mathcal{G}[t, f](dx)$.

Fitness functions $f \in \mathbb{R}^X$ are assumed to have a finite minimum, denoted by $f^* = \inf_X f(x) > -\infty$. For the purposes of the present study, unbounded functions in $\mathbb{R}^X$ may be ignored without causing a problem.

Only static, single-objective fitness functions are considered in this theory. The formalism could be expanded to accommodate either dynamic or stochastic objective functions, but these adaptations would yield separate results and would complicate the discussion that follows. Unless mentioned or otherwise clear from context, the terms *optimization* and *optimum* should be interpreted as *minimization* and *minimum* in the remainder of the paper.

As mentioned above, trajectories will be indexed using superscripts, so that t^n indicates the n^{th} evaluation point in t, with indices starting at 1 for the initial point. Negative superscripts index the trajectory backwards, so that t^{-1} is the last point in t, t^{-2} the next to last, and so on. Subscripts on trajectories indicate a sequence of trajectories, so that e.g. t_n is not a point, but an entire sequence of points. Thus t_n^m represents a particular point within a sequence of evaluation histories. Two trajectories can be concatenated to form a longer trajectory, denoted by a union operator, e.g. $t = t_1 \cup t_2$. An element $x \in X$ can also be appended to a trajectory, denoted similarly by $t = t_1 \cup x$. The notation $t \cup \left(\bigcup_{i=1}^K x_i\right)$ indicates successive concatenation, i.e. $t \cup x_1 \cup x_2 \cup \cdots \cup x_K$. Given a sequence $(x_n)_{n=1}^\infty$, the expression $(x_n)_{n=1}^N$ represents a trajectory of length N, and $(x_n)_{n=1}^0 = \emptyset$ by convention. In addition to indexing, the notation $x \in t$ will be used to indicate that x is an arbitrary point occurring at some point in t, i.e., $x = t^n$ for some n.

As discussed in Section 3.2, the notation $H(t)$ is used to convert a trajectory in $\mathcal{T}[X]$ to a trajectory in $\mathcal{T}[X^K]$ for some fixed K that will be clear from the context. When this is done, $H(t)^n$ refers to the n^{th} entry of $H(t)$, an element of X^K, and $H(t)^{n,k}$ refers to the k^{th} component of the n^{th} entry, an element of X.

Consider the process of running an optimizer $\mathcal{G} \in \mathcal{PF}$ on an objective function f. First, the trajectory is initialized to $t_0 = \emptyset$. Then, a point $x_1 \in X$ is sampled from $\mathcal{G}[t_0, f]$. This population is appended to t_0 to create $t_1 = t_0 \cup x_1$. Next, a population x_2 is sampled from $\mathcal{G}[t_1, f]$ and appended to t_1 to form $t_2 = t_1 \cup x_2$. The process continues until a stopping criterion is reached. Thus in actual practice the trajectory t is sampled progressively from the optimizer $\mathcal{G}$, and the trajectory takes on random values. This random sequence of evaluation points is a stochastic process, termed the *optimization process*, denoted by $Z = (Z_n)_{n\in\mathbb{N}}$. The process Z is said to be *generated by* $\mathcal{G}$ on f.

With the basic formalism now presented, the Sections 4 and 5 will expand these fundamentals with definitions and constructive operators that make it possible to describe more complex optimizers in Section 6 and 7 and to prove theorems about them in Section 8.

4. ALGEBRAIC OPERATIONS

Optimizers can be combined or altered algebraically to form a new optimizer in several ways. In this section, mechanisms for algebraically combining operators are discussed. These operations will be used extensively to provide formal representations of evolutionary algorithms in Section 6.

4.1 Convolution

The first operator will be termed convolution due to its similarity to the convolution of two functions. In this case, two optimizers are convolved to produce a third optimizer, and the constituent parts may or may not be practical optimizers on their own. The convolution operator, denoted by $\star$, is defined by the equation

$$(\mathcal{G}_1 \star \mathcal{G}_2)[t, f](A) \equiv \int_X \mathcal{G}_2[t \cup x, f](A)\ \mathcal{G}_1[t, f](dx). \quad (5)$$

Convolution performs the intuitive function of applying two probability distributions in sequence. First, a point is sampled from $\mathcal{G}_1$, and then a point is sampled from $\mathcal{G}_2$ given the outcome of sampling $\mathcal{G}_1$. In fact, the entire process of stochastic optimization described in the previous sections boils down to the successive application of the convolution operator, so that if $(Z_n)_{n\in\mathbb{N}}$ is generated by $\mathcal{G}$, then $(Z_{2n})_{n\in\mathbb{N}}$ is generated by $\mathcal{G}\star\mathcal{G}$. In general, $Z_n \sim (\bigstar_{m=1}^n \mathcal{G})[\emptyset, f]$ when $Z \sim \mathcal{G}_f$, where $\bigstar_{m=1}^n \mathcal{G}$ represents n successive applications of convolution. When two or more convolution operators are used, convolution is assumed to be left associative, e.g. $\mathcal{G}_1 \star \mathcal{G}_2 \star \mathcal{G}_3 = (\mathcal{G}_1 \star \mathcal{G}_2) \star \mathcal{G}_3$. Because convolution is not necessarily commutative, right association is not equal to left association, and so the postfix notation $(\mathcal{G}\bigstar_{m=1}^n)$ will indicate chained right associations, e.g. $\mathcal{G}_1 \star (\mathcal{G}_2 \star \mathcal{G}_3)$. The description of evolutionary algorithms in particular can be substantially simplified by the use of the convolution operator. For example, a genetic algorithm may be described as the convolution of selection, crossover, and mutation distributions.

4.2 Trajectory Truncation

Sometimes an optimizer will ignore one or more elements of the evaluation history. This property is particularly important for population-based optimizers, which often ignore the previously generated members of the current population. Define trajectory truncation by the symbol $\triangleleft$ so that

$$(\triangleleft\mathcal{G})[t \cup x, f] = \mathcal{G}[t, f], \quad (6)$$

with the base case $(\triangleleft\mathcal{G})[\emptyset, f] = \mathcal{G}[\emptyset, f]$. This operator can be applied to the same optimizer more than once. The notation $\triangleleft_k\mathcal{G}$ will be used to represent the optimizer resulting from $k \geq 0$ applications of trajectory truncation, with $\triangleleft_0\mathcal{G} \equiv \mathcal{G}$.

4.3 Pointwise Scalar Multiplication

Define pointwise scalar multiplication so that

$$(\alpha\mathcal{G})[t, f](A) \equiv \alpha(\mathcal{G}[t, f](A)) \quad (7)$$

for $\alpha \in \mathbb{R}$. It is clear that $\alpha\mathcal{G}$ is not a member of $\mathcal{PF}$ for $\alpha \neq 1$, since $\alpha\mathcal{G}[t, f](X) = \alpha$ and $\alpha\mathcal{G}[t, f]$ is not a probability distribution. However, $\alpha\mathcal{G}$ is a member of a larger space that will be introduced in Section 5 below.

4.4 Pointwise Addition

Define pointwise addition so that

$$(\mathcal{G}_1 + \mathcal{G}_2)[t, f](A) \equiv \mathcal{G}_1[t, f](A) + \mathcal{G}_2[t, f](A). \quad (8)$$

As with scalar multiplication, it is clear that $\mathcal{G}_1 + \mathcal{G}_2$ is not contained within $\mathcal{PF}$, but the operation is well-defined nonetheless and is used below.

4.5 Convex Combination

Optimizers can be combined convexly to form new operators using the basic operations of pointwise addition and pointwise scalar multiplication. Used by themselves, these two operations are not closed on $\mathcal{PF}$, but their convex combinations are closed.

Let $\alpha \in [0,1]$ and consider $\mathcal{G} = \alpha\mathcal{G}_1 + (1-\alpha)\mathcal{G}_2$. Then $\mathcal{G}[t,f]$ is always a probability distribution, so $\mathcal{G} \in \mathcal{PF}$. More generally, choose $\alpha_1, \ldots \alpha_n$ in $[0,1]$ such that $\sum_i \alpha_i = 1$, and suppose that $\mathcal{G}_1, \ldots, \mathcal{G}_n$ are optimizers. Then $\mathcal{G} = \sum_i \alpha_i \mathcal{G}_i$ is a convex combination of $\mathcal{G}_1, \ldots, \mathcal{G}_n$, and $\mathcal{G} \in \mathcal{PF}$. So $\mathcal{PF}$ is closed under convex combination.

To emphasize, $\mathcal{PF}$ is a convex space, i.e. it is closed under convex combinations. For any $\mathcal{G}_1, \ldots, \mathcal{G}_n$ contained in any one of these spaces, all convex combinations also lie inside the same space.

The fact that $\mathcal{PF}$ is convex is the first part of a larger result. In fact, $\mathcal{PF}$ is a closed, convex subset of a normed vector space, which means that the space of optimizers is continuous and linear. This topic is explored next.

5. A NORMED VECTOR SPACE

Iterative stochastic optimizers are vectors in an appropriate vector space under the vector operations of pointwise scalar multiplication and pointwise addition. A vector space is continuous and linear, meaning that one can make arbitrarily small perturbations of one vector in the direction of any other vector. Such spaces have a regular structure that is easily manipulated in order to prove theorems. The vector space of optimizers consists of functionals that take a trajectory and a fitness function and return a signed measure.

5.1 Vector Space

A *vector space*, also termed a *linear space*, is a space in which each element (called a *vector*) can be scaled to form a line through scalar multiplication or added to another element to obtain a third element also contained in the space. The multiplication and addition operations must satisfy certain requirements, namely: addition needs to be commutative and invertible; an identity must exist for each operation; and multiplication must distribute over addition. A *normed vector space* additionally possesses a *norm*, which assigns an absolute magnitude to each element in the space and can be used to generate a distance metric. The space of optimizers $\mathcal{PF}$ can be used to generate a normed vector space by extending it to include the closure of $\mathcal{PF}$ under scalar multiplication and pointwise addition.

5.2 Measure-Valued Functionals

For $\mathcal{G} \in \mathcal{PF}$ and $\alpha \neq 1$, the pseudo-optimizer $\alpha\mathcal{G}$ fails to be in $\mathcal{PF}$ only because $\alpha\mathcal{G}[t,f](X) = \alpha \neq 1$. Although $\alpha\mathcal{G}[t,f]$ is not a probability measure, it is a measure. A *finite signed measure* is a set-valued function defined over a σ-algebra that is additive on disjoint sets. It may take on both positive and negative values, but must be finite on every set in the σ-algebra. Denote by $\mathcal{M}[X] = \mathcal{M}[X, \mathcal{B}_\tau]$ the space of all finite signed measures on $(X, \mathcal{B}_\tau)$. The space $\mathcal{M}[X]$ is a Banach space, a complete, normed vector space. The standard norm for $\mathcal{M}[X]$ is the *total variation norm*, which is the largest absolute measure assigned to any set in the σ-algebra, $||\mu||_{\mathcal{M}} \equiv \sup_{A \in \mathcal{B}_\tau} |\mu(A)|$.

The space of probability measures $\mathcal{P}[X]$ is a closed, convex subset of $\mathcal{M}[X]$. Although all probability measures have a total variation norm of 1 by definition, the difference of two probability measures is well defined and non-trivial. This difference defines a distance metric on probability measures,

$$d(\mathbb{P}, \mathbb{Q}) = ||\mathbb{P} - \mathbb{Q}||_{\mathcal{M}} = \sup_{A \in \mathcal{B}_\tau} |\mathbb{P}(A) - \mathbb{Q}(A)| \quad (9)$$

for probability measures $\mathbb{P}$ and $\mathbb{Q}$. Intuitively, the distance between two probability measures is determined by the set to which the two measures assign the largest difference in probability mass.

Now define the functional space

$$\mathcal{MF}_0 = \mathcal{MF}_0[X] = \left\{\mathcal{G} : \mathcal{T}[X] \times \mathbb{R}^X \to \mathcal{M}[X]\right\}, \quad (10)$$

where $\mathcal{MF}$ stands for *measure-valued functionals*. The space $\mathcal{MF}_0$ contains $\mathcal{PF}$, but it also contains many other objects as well. An element in $\mathcal{MF}_0$ is a function that produces a finite signed measure over the search space when given any finite trajectory and any fitness function. Define vector operations in $\mathcal{MF}_0$ pointwise as for $\mathcal{PF}$. These vector operations satisfy the required vector properties. The zero vector for $\mathcal{MF}_0$ is the function that returns the zero measure on all inputs. So $\mathcal{MF}_0$ is a vector space. In fact, $\mathcal{MF}_0$ is just the vector closure of $\mathcal{PF}$ under the operations of pointwise scalar multiplication and addition.

The next step is to find a norm for $\mathcal{MF}_0$ to compare the distance between optimizers. A norm can be created from

$$||\mathcal{G}||_{\mathcal{MF}} = \sup_{t \in \mathcal{T}, f \in \mathbb{R}^X} ||\mathcal{G}[t,f]||_{\mathcal{M}}. \quad (11)$$

The function $||\cdot||_{\mathcal{MF}}$ satisfies all the properties of a norm with the exception that it is not bounded on $\mathcal{MF}_0$. However, the subset of $\mathcal{MF}_0$ on which it is finite forms a vector subspace $\mathcal{MF}_0$ that contains $\mathcal{PF}$. To this end, define

$$\mathcal{MF} \equiv \{\mathcal{G} \in \mathcal{MF}_0 : ||\mathcal{G}||_{\mathcal{MF}} < \infty\}.$$

THEOREM 5.1. *$\mathcal{MF}$ is a normed vector subspace of $\mathcal{MF}_0$ under $||\cdot||_{\mathcal{MF}}$.*

PROOF. The vector space structure of $\mathcal{MF}_0$ has already been discussed. To see that $||\cdot||_{\mathcal{MF}}$ is a norm, note that for $\mathcal{G} \in \mathcal{MF}$, $\alpha \in \mathbb{R}$,

$$||\alpha\mathcal{G}||_{\mathcal{MF}} = \sup_{t,f} ||\alpha\mathcal{G}||_{\mathcal{M}} = |\alpha|\,||\mathcal{G}||_{\mathcal{MF}}.$$

Additionally, if $\mathcal{G} \neq 0$, there exist t, f s.t. $||\mathcal{G}[t,f]||_{\mathcal{M}} > 0$, so $||\mathcal{G}||_{\mathcal{MF}} > 0$ as well. For the triangle inequality,

$$\begin{aligned} ||\mathcal{G}_1 + \mathcal{G}_2||_{\mathcal{MF}} &= \sup_{t \in \mathcal{T}, f \in \mathbb{R}^X} ||\mathcal{G}_1[t,f] + \mathcal{G}_2[t,f]||_{\mathcal{M}} && (12)\\ &\leq \sup_{t \in \mathcal{T}, f \in \mathbb{R}^X} ||\mathcal{G}_1[t,f]||_{\mathcal{M}} + ||\mathcal{G}_2[t,f]||_{\mathcal{M}} && (13)\\ &\leq \sup_{t \in \mathcal{T}, f \in \mathbb{R}^X} ||\mathcal{G}_1[t,f]||_{\mathcal{M}} + \sup_{t \in \mathcal{T}, f \in \mathbb{R}^X} ||\mathcal{G}_2[t,f]||_{\mathcal{M}} && (14)\\ &= ||\mathcal{G}_1||_{\mathcal{MF}} + ||\mathcal{G}_2||_{\mathcal{MF}} < \infty. && (15) \end{aligned}$$

So $||\cdot||_{\mathcal{MF}}$ is indeed a norm.

$\mathcal{MF}$ is a vector subspace because it contains the zero vector and is closed under vector addition and scalar multiplication. □

It may be asked whether $\mathcal{MF}$ is complete and therefore Banach. The answer is no; it is easy to create sequences in $\mathcal{MF}$ with an unbounded norm in the limit. However, this fact will not be particularly restrictive for the purpose of analysis, since the iterative stochastic optimizers form a closed subset of $\mathcal{MF}$.

PROPOSITION 5.2. *$\mathcal{PF}$ is a closed, convex subset of $\mathcal{MF}$.*

PROOF. First of all, if $\mathcal{G} \in \mathcal{PF}$ then $||\mathcal{G}||_{\mathcal{MF}} = 1 < \infty$, so $\mathcal{PF} \subseteq \mathcal{MF}$. To show that $\mathcal{PF}$ is closed, let $||\mathcal{G}_n - \mathcal{G}||_{\mathcal{MF}} \to 0$ for $(\mathcal{G}_n) \subseteq \mathcal{PF}$. Then for all t, f, $||\mathcal{G}_n[t,f] - \mathcal{G}[t,f]||_{\mathcal{M}} \to 0$, and for all $A \in \mathcal{B}_\tau$,

$$\mathcal{G}[t,f](X) = \lim_n \mathcal{G}_n[t,f](X) = 1, \quad (16)$$

$$\mathcal{G}[t,f](A) = \lim_n \mathcal{G}_n[t,f](A) \geq 0. \quad (17)$$

That is, $\mathcal{G}[t,f]$ is a probability measure, so $\mathcal{G} \in \mathcal{PF}$.

To establish convexity, let $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{PF}$, and let $\alpha \in [0,1]$. Set $\mathcal{G} = \alpha\mathcal{G}_1 + (1-\alpha)\mathcal{G}_2$. Then for all t, f, and A,

$$\mathcal{G}[t,f](X) = \alpha\mathcal{G}_1[t,f](X) + (1-\alpha)\mathcal{G}_2[t,f](X) = 1, \quad (18)$$

$$\mathcal{G}[t,f](A) = \alpha\mathcal{G}_1[t,f](A) + (1-\alpha)\mathcal{G}_2[t,f](A) \geq 0, \quad (19)$$

and therefore $\mathcal{G} \in \mathcal{PF}$. □

Thus iterative stochastic optimizers are vectors. There is a fixed numeric distance between any two computable optimizers, and there is a an entire spectrum of computable optimizers that progressively blend between them, no matter how different the optimizers in question may be. For example, there exists a unique optimizer that is exactly halfway between conjugate gradient descent and differential evolution. This insight is surprising and opens a fundamentally new way of looking at optimization methods. A sampling of how this theory of optimizers may be used to prove theorems about optimizers is given in Section 8, after specific discussion of how evolutionary algorithms fit into the formalism.

6. EVOLUTIONARY ALGORITHMS

In this section, the basic components of evolutionary algorithms are represented in the formal framework of the previous sections. This process demonstrates the formalization and lays the ground work to prove complex theorems, as is done in Section 8.

6.1 Evolutionary Algorithms Characteristics

The core characteristics of an evolutionary algorithm include competition within a population, preferential selection of competitive individuals, reproduction among selected individuals, and random variation of selected individuals. These four processes can be realized into evaluation, selection, recombination, and mutation phases. In the case of asexual reproduction, recombination may be vacuous. In formal terms, an evolutionary algorithm can be formalized as a convolution of three components, one each for selection, recombination, and mutation processes.

Each phase of an evolutionary algorithm can be described as an optimizer, just as the acceptance phase and proposal phase of simulated annealing were separated into two different components in Section 3.3. Thus evolutionary algorithms can be formalized by defining what principles make an optimizer work as a selection rule, a recombination operator, or a mutation operator. The optimizers representing each phase are not effective optimizers by themselves. The constituent components of an evolutionary algorithm are only optimizers in a formal sense, but these components represent genetic operators that can be mixed and matched mathematically to form new optimization methods.

6.2 Selection, Recombination, and Mutation

These three phases of an evolutionary algorithm may be thought of as intermediate steps, each of which creates a full population and hands it off to the next phase. Selection chooses K points from among the previously observed points. Recombination invokes one or more additional selection rules to tack on extra parents and then merges these parents with a crossover rule. This merged output of K individuals is then handed off to the mutation operator, which alters each individual independently. These three stages will now be discussed rigorously one at a time.

6.2.1 Selection Rules

Selection in evolutionary algorithms is a filtering task, characterized as follows: Given a set of previously observed individuals, select a group of K individuals to form the basis of the next population. The selection process must place zero weight on unobserved individuals; only members of the population history can be selected. Given a trajectory $t \in \mathcal{T}$, define the previously observed individuals in t as $P(t) = \{x \in X : \exists n \text{ s.t. } x = t^n\}$. Accounting for populations, a *selection rule* is an optimizer that places zero probability on any proposed population that would expand $P(t)$.

Definition 5. An optimizer $\mathcal{S} \in \mathcal{PBO}_K$ is a *selection rule* if $\mathcal{S}[t,f](A) = 0$ whenever $\exists x \in A$ s.t. $P(t \cup x) \neq P(t)$.

It may seem strange that the selection rule is allowed to select any member of $P(t)$ and not just the members of the last population ($P(H(t)^{-1})$). But there are a number of evolutionary methods that select members of populations prior to the last population, such as those using elitist selection. Methods that store the locally best individual (such as differential evolution) also need the flexibility to select from previous generations. Furthermore, several recently proposed techniques such as Novelty Search [9], Curiosity Search [15], and Evolutionary Annealing [10] store members from each population in an archive, making them available for selection.

6.2.2 Recombination and Crossover

Recombination combines some number of selected individuals as parents to form a hybrid child. Although traditional recombination methods in genetic algorithms utilize only two parents, other methods use an arbitrary number of parents. In evolution strategies, for example, intermediate crossover averages components across several solutions. A recombination operator first selects the parents for each member of the population and then invokes a crossover rule to combine the parents. The number of selected parents (usually just two) is said to be the *order* of the crossover rule and the recombination operator. Parent selection for an n^{th} order operator stacks n populations on top of the current trajectory. A crossover rule consumes these n populations and leaves a single merged population in their place.

The key feature of a crossover rule is that it should combine only the selected parents. It should therefore be independent of all other components of the input trajectory. It should also ignore the fitness of the selected parents, deferring such judgments to the selection operators. For the k^{th}

member of the population the selected parents in a crossover rule of order n are just the k^{th} members of the previous n populations in the trajectory. Define the trajectory

$$\text{parents}(t, n, k, K) \equiv \bigcup_{i=1}^{n} H(t)^{-i,k}, \tag{20}$$

recalling that $H(t)$ is the population history of t, negative indices count backwards from the end of the history, and the double index chooses the k^{th} member of the $-i^{th}$ population. Then $\text{parents}(t, n, k, K)$ is the reverse ordered list of the parents available to the crossover rule.

Definition 6. An objective-agnostic optimizer $\mathcal{C} \in \mathcal{PBO}_K$ is a crossover rule of order n if there exist $\mathcal{C}_1, \ldots, \mathcal{C}_K \in \mathcal{PF}$ such that $\mathcal{C}[t, f] = \mathcal{C}_{k(t)}[t, f]$ and for all $k = 1, \ldots, K$ and all $t_1, t_2 \in \mathcal{T}$, $\mathcal{C}_k[t_1, f] = \mathcal{C}_k[t_2, f]$ whenever

$$\text{parents}(t_1, n, k, K) = \text{parents}(t_2, n, k, K).$$

That is, a crossover rule is independent of all but the selected parents.

This definition of crossover accepts a wide range of instantiations that do not necessarily match the concept of crossover in a traditional genetic algorithm. This intuition will be restored with the introduction of crossover masks in Section 7. With crossover rules defined, the definition of a recombination operator can now be given.

Definition 7. An optimizer $\mathcal{R} \in \mathcal{PBO}_K$ is a recombination operator of order n if there exists a sequence of $n-1$ selection rules $\mathcal{S}_1, \ldots, \mathcal{S}_{n-1} \in \mathcal{PBO}_K$ and a crossover rule $\mathcal{C} \in \mathcal{PBO}_K$ of order n such that

$$\mathcal{R} = \triangleleft\mathcal{S}_1 \star (\triangleleft_2\mathcal{S}_2 \star (\cdots \star (\triangleleft_{n-1}\mathcal{S}_{n-1} \star \mathcal{C}))).$$

Operationally, each of the selection rules $\mathcal{S}_i$ are applied in order, with the previous selection hidden by the trajectory-truncation operator. Finally, the crossover rule is invoked to combine the selected points, including the first point selected by an initial selection rule outside of the recombination operator. [2] The convolution is performed with right association so that the results of selection are stacked together and not consumed until the crossover rule is reached. Note that there is only one possible recombination operator of order 1, and it vacuously reproduces the selected population, representing asexual reproduction.

6.2.3 Mutation Operators

Mutation in evolutionary algorithms alters a single member of a proposed population. Mutation must be objective-agnostic; it cannot be aware of the fitness of the point it is mutating.In addition, a mutation operator can only vary the individual member of the population that has been proposed to it. That is, a mutation operator must ignore every member of the trajectory except the one that is being mutated. Conversely, a mutation operator cannot simply ignore the individual it is mutating. Therefore, a condition is included stating that the mutation operator must depend on the object being mutated for at least some trajectories. These restrictions are added to make the definition of evolutionary algorithms in Definition 9 below meaningful; without these conditions, any optimizer would be an evolutionary algorithm.

Definition 8. An optimizer $\mathcal{V} \in \mathcal{PBO}_K$ is a *mutation operator* if $\mathcal{V}$ is factorial and objective-agnostic and for all $1 \le i \le K$, the following two conditions hold:

- $\forall t_1, t_2 \in \mathcal{T}$, $\mathcal{V}_i[t_1, f] = \mathcal{V}_i[t_2, f]$ whenever $H(t_1)^{-1,i} = H(t_2)^{-1,i}$, and
- $\exists t_1, t_2 \in \mathcal{T}$ s.t. $H(t_1)^{-1,i} \neq H(t_2)^{-1,i}$ and $\mathcal{V}_i[t_1, f] \neq \mathcal{V}_i[t_2, f]$.

These definitions are constructed to be as restrictive as possible while still accounting for the full range of genetic operators commonly used in evolutionary methods. They can be assembled to form a complete algorithm using the convolution operator.

6.3 Quasi-Evolutionary Algorithms

Selection, recombination, and mutation operators can be combined to form a general schema for evolutionary algorithms. The result is slightly more general than a typical evolutionary algorithm and is termed a *quasi-evolutionary algorithm.*

Definition 9. An optimizer $\mathcal{E} \in \mathcal{PBO}_K$ is called a *quasi-evolutionary algorithm* if it is not objective-agnostic and if there exist a selection rule $\mathcal{S}$, a recombination operator $\mathcal{R}$ of order 1 or greater, and a mutation operator $\mathcal{V}$ such that $\mathcal{E} = \mathcal{S} \star \mathcal{R} \star \mathcal{V}$.

PROPOSITION 6.1. *By implication, $\mathcal{E} \in \mathcal{PBO}_K$ is also a quasi-evolutionary algorithm if it is not objective-agnostic and there is a selection rule $\mathcal{S}$ and a mutation operator $\mathcal{V}$ such that $\mathcal{E} = \mathcal{S} \star \mathcal{V}$, in which case $\mathcal{E}$ has a recombination operator of order* 1.

Intuitively, a quasi-evolutionary algorithm samples one or more selection rules to propose a new parent population from the selected individuals, then recombines the parent population to form a new child population, and finally samples a mutation operator to alter the selected individuals.

The term "quasi-evolutionary algorithm" is used because this formalism does not exclude certain non-evolutionary algorithms such as Nelder-Mead or greedy hill-climbing. In fact, many optimization methods can be formally described as "quasi-evolutionary" through the use of innovative crossover rules, despite the substantial restrictions on the definitions above. When evolutionary algorithms are described mathematically in this manner, it is not clear that there exists a single elegant description that separates traditional evolutionary algorithms from other optimization methods [11].

7. GENETIC ALGORITHMS

Modern genetic algorithms mix and match a variety of selection, crossover, and mutation components to form an optimization routine. This section will examine some of the most common among these components.

7.1 Selection in GAs

Selection in genetic algorithms is typically restricted to the members of the last population, so that a genetic algorithm unfolds as a sequence of populations each constructed solely from the previous one. An optimizer $\mathcal{G} \in \mathcal{PBO}_K$ is termed

[2] The initial selection rule could have been pushed inside the recombination operator, but keeping it outside makes the formal definition of an evolutionary algorithm more natural.

population-Markov if it depends only on the last population, that is, if $\mathcal{G}[t_1, f] = \mathcal{G}[t_2, f]$ whenever $H(t_1)^{-1} = H(t_2)^{-1}$. Genetic algorithms are population-Markov in general.

PROPOSITION 7.1. *A quasi-evolutionary algorithm is population-Markov if and only if its recombination operator and selection rule are, and a recombination operator is population-Markov if and only if all of its selection rules are.*

Two of the most common selection rules historically are proportional selection and tournament selection. In *proportional selection*, members of the prior population are selected independently proportional to their fitness in the previous population. Ordinarily, the fitness function is assumed to be positive, and the genetic algorithm is maximizing the fitness and so prefers larger fitness values. To use proportional selection for minimization, a *modulating function* $g > 0$ is introduced so that $g(t, y)$ is intended to be positive and increasing as $y = f(x)$ is minimized. If it is desired to maximize f and $f > 0$, then $g(t, y) = |y|$ will prefer the minimal values of $-f$. Proportional selection with this choice of modulating function will be termed *standard proportional selection* or *roulette wheel selection*. A more neutral choice is $g(t, x) = \exp(-x)$; note that this choice is similar to the acceptance probability for simulated annealing. Given a modulating function g, proportional selection is given by

$$\mathcal{PS}\left\langle g\right\rangle [t, f]\left(\{y\}\right) \propto N_{H(t)^{-1}}\left(y\right)\left[g(t, f\left(y\right))\right], \tag{21}$$

where $N_P\left(y\right)$ is the number of times the individual y appears in the population P. Then N_P is nonzero for at most K points, so the normalization can be computed by summing over the prior population $H(t)^{-1}$.

Proportional selection is highly sensitive to the magnitude of variation in the fitness function and so can become trapped in steep local minima. *Tournament selection* chooses members of the prior population according to their rank in the population in order to maintain diversity within the population. In this section only tournaments over the full population are considered. Full tournament selection chooses the best member of the last population with probability q. If the best member is not selected, the second best member is chosen with probability q, and then the third, and the fourth, and so on. If the population is exhausted, the selection wraps back around to the best individual. The parameter q is referred to as the *selection pressure* since high values of q favor the best individuals in the population. Tournament selection is given explicitly by

$$\mathcal{TS}\left\langle q\right\rangle [t, f]\left(\{y\}\right) \propto (1-q)^{R\left(y, f, H(t)^{-1}\right)}, \tag{22}$$

where $R\left(y, f, P\right) \in \mathbb{N} \cup \{\infty\}$ is the rank of the individual y in the population P under the fitness function f, with 0 being the best rank, and $R\left(y, f, P\right) = \infty$ if y does not appear in P. In case of ties among non-equal individuals, assume later members of the population are ranked higher. Again, $\mathcal{TS}$ is nonzero for at most K points so that the normalization is easily computed. Notice that the right-hand side of Equation 22 can be treated as a modulating function.

7.2 Crossover in GAs

The distinguishing characteristic of a genetic algorithm is undoubtedly recombination with two parents (sexual reproduction). Standard crossover rules of order 2 include one point crossover, multipoint crossover, and uniform crossover. The parents are selected using one or more selection rules, and then a "child" is created using the crossover rule to combine the properties of the parents.

Because crossover rules are specific to the search space, examples will only be given for the case in which the search space X is a d-dimensional vector space, $X = Y^d$, such as $X = \mathbb{R}^d$ (Euclidean space) or $X = \{0, 1\}^d$ (binary space). In this case, many second-order crossover rules can be determined by a random binary vector $M \in \{0, 1\}^d$ which will be termed the *crossover mask*. If $M_i = 1$, then the child copies the i^{th} attribute of the father. If $M_i = 0$, then the child copies the i^{th} attribute of the mother. Denote by $\mathbf{1}$ the vector in $\{0, 1\}^d$ whose entries are all one, and let $x \otimes y$ be the vector that is the componentwise product of vectors x and y. For a trajectory t, let $p(t)$ be the selected father and $m(t)$ the selected mother, so that $p(t) = \text{parents}(t, 2, k(t), K)^{-1}$ and $m(t) = \text{parents}(t, 2, k(t), K)^{-2}$. Define a random variable C_t by

$$C_t = M \otimes p(t) + (\mathbf{1} - M) \otimes m(t). \tag{23}$$

Then given a distribution $\mathbb{P}_M$ over M, a *masked crossover rule* is just the distribution of C_t and can be written as

$$\mathcal{C}\left\langle \mathbb{P}_M\right\rangle [t, f](A) = \sum_{z \in \{0,1\}^d} \mathbb{P}\left(C_t \in A \mid M = z\right)\ \mathbb{P}_M\left(z\right), \tag{24}$$

Single point, multipoint, and uniform crossover can be defined by specifying $\mathbb{P}_M$. For uniform crossover, the choice of mask is uniformly random,

$$\mathcal{UC}[t, f] = \mathcal{C}\left\langle \text{Uniform}\left(\{0, 1\}^d\right)\right\rangle. \tag{25}$$

For single point crossover, a random index $i \in \{1, \ldots, d\}$ is chosen, and the mask is set so that $M_j = 1$ for $j \leq i$ and $M_j = 0$ for $j > i$. In multipoint crossover, a fixed number of random indices $i_1, \ldots, i_n$ are chosen and then sorted. M then alternates between series of zeros and a series of ones, starting with ones and with switches occurring at each of the i_j. Let $\mathcal{SC}$ denote single-point crossover and let $\mathcal{MC}$ represent multipoint crossover.

7.3 Mutation Operators

Mutation operators depend on the search space and can be almost any objective-agnostic distribution. The most common mutators, however, are Bernoulli mutation in binary spaces and Gaussian mutation in Euclidean space. In discrete or combinatorial spaces, mutation distributions typically involve random structural operators.

First, consider Gaussian mutation in $X = \mathbb{R}^d$. The mean of the Gaussian is simply the point being mutated (t^{-1}) and the covariance is a function of the prior points evaluated, often a constant. Then *Gaussian mutation* with a covariance-generating function Σ is given by

$$\mathcal{N}\left\langle \Sigma\right\rangle [t, f] \quad = \quad \mathcal{N}\left(t^{-1}, \Sigma(H(t))\right), \tag{26}$$

where $\mathcal{N}\left(\mu, \Sigma\right)$ is the normal distribution and the symbol $\mathcal{N}$ is overloaded to represent Gaussian mutation as well.

When the search space is binary, $X = \{0, 1\}^d$, *Bernoulli mutation* at rate p is given by

$$\mathcal{B}\left\langle p\right\rangle [t \cup z, f]\left(\{y\}\right) = \prod_j p^{|y_j - z_{i,j}|}\left(1-p\right)^{\left(1-|y_j - z_{i,j}|\right)} \tag{27}$$

7.4 Formal Genetic Algorithms

An evolutionary algorithm can be defined as a quasi-evolutionary algorithm with a masked crossover rule. A genetic algorithm can then be identified as an evolutionary algorithm that is also population-Markov. In contrast, evolution strategies depend on adaptive parameters and are not usually population-Markov.

Definition 10. An optimizer $\mathcal{G} \in \mathcal{PBO}_K$ is an *evolutionary algorithm* if it is a quasi-evolutionary algorithm with a masked crossover rule. Additionally, $\mathcal{G}$ is a *genetic algorithm* if it is also population-Markov.

This definition encompasses most traditional evolutionary algorithms and excludes more recent developments that still conform to the definition of a quasi-evolutionary algorithm as defined above. Again, a crossover rule of order one may be used, so that every quasi-evolutionary algorithm with a vacuous crossover rule is trivially an evolutionary algorithm.

Putting all of these pieces together, a basic genetic algorithm with single-point crossover, proportional selection, and a binary encoding can be written as

$$\mathcal{SGA}\langle p\rangle = (\mathcal{PS} \star ((\triangleleft \mathcal{PS}) \star \mathcal{SC})) \star \mathcal{B}\langle p\rangle, \qquad (28)$$

which is Goldberg's simple genetic algorithm with a mutation rate of p [5]. The resulting algorithm is formally a genetic algorithm, since it is composed of a population-Markov selection rule, a recombination operator with masked crossover, and a mutation operator. Most standard genetic algorithms can be written down similarly by mixing and matching the components described in this section.

8. CONTINUITY OF OPTIMIZERS

The adopted formalism for stochastic optimizers is useful because it leads to general theoretical results. The characterization of optimizers as a closed, convex subset of a normed vector space in Section 5 is one such result. But many more results are also possible. For example, the theory allows one to state the conditions under which optimizers may be expected to have similar behavior on similar problems, i.e., the conditions under which they are continuous. Continuity is important because it determines when approximations of the fitness function or the optimizer may be used without a loss of performance. Such approximations are already used to optimize complex fitness functions more efficiently. The theory may also suggest that certain uncomputable optimizers may perform well, and continuity means that computable approximations will perform similarly. The section states several theorems about the continuity of genetic algorithms in particular. A few proofs are omitted; these proofs can be found in Chapter 5 of [11].

8.1 What is a Continuous Optimizer?

Optimizers are functions from a trajectory and a fitness function to a signed measure over the search space. Continuity answers the following two questions:

- Will an optimizer choose similar points when given similar evaluation histories?
- Will an optimizer choose similar points when given similar fitness functions?

The first question pertains to *continuity in trajectories*, and the second question to *continuity in objectives*.

Continuity is a topological concept. The most familiar type of topology is the metric topology, which induces the epsilon-delta definition of continuity. A function f that maps one metric space (X, d_X) to another metric space (Y, d_Y) is continuous if for every $\epsilon > 0$ and every point x there exists a $\delta = \delta(x) > 0$ such that for all y with $d_X(x, y) < \delta$, it holds that $d_Y(f(x), f(y)) < \epsilon$.

The search domain X is assumed to be metric from this point forward for simplicity, since most practical search domains are metric. A metric on the space of evaluation histories $\mathcal{T}[X]$ is then given by

$$d_\rho(t_1, t_2) = |\,|t_1| - |t_2|\,| + \sum_{i=1}^{|t_1|\wedge|t_2|} \rho\left(t_1^i, t_2^i\right), \qquad (29)$$

where ρ is a metric on X, t_1^i is the i^{th} element of t_1, and $|t_1|$ is the length of the trajectory t_1.

Definition 11. An optimizer $\mathcal{G} \in \mathcal{MF}[X]$ is *continuous in objectives* at f if for any sequence of fitness functions $\{f_n\}$, $f_n \to f$ implies $||\mathcal{G}[t, f] - \mathcal{G}[t, f_n]||_{\mathcal{M}} \to 0$.

Definition 12. An optimizer $\mathcal{G} \in \mathcal{MF}$ is *continuous in trajectories* at t if for every $\epsilon > 0$ there exists a $\delta > 0$ such that whenever $d_\rho(t, u) < \delta$ then $||\mathcal{G}[t, f] - \mathcal{G}[u, f]||_{\mathcal{M}} < \epsilon$.

If an optimizer is continuous in objectives, then it can be expected to perform similarly on similar problems. If an optimizer is continuous in trajectories, then it can be expected to make similar decisions on similar trajectories.

8.2 Continuity of Evolutionary Algorithms

In Section 6, a quasi-evolutionary algorithm was defined as a convolution $\mathcal{E} = \mathcal{S} \star \mathcal{R} \star \mathcal{V}$. Evolutionary algorithms can be continuous or discontinuous, depending on the details of the genetic operators. Mutation operators are independent of the fitness function and therefore trivially continuous in objectives. Typically, mutation operators are continuous in trajectories as well, as with Bernoulli or Gaussian mutation. Crossover rules are likewise independent of objectives and therefore continuous over objectives. Cases where evolutionary algorithms as a whole are continuous or discontinuous will be addressed with two general theorems in this subsection. These theorems demonstrate two distinct cases in which a convolution can be continuous. First, a convolution $\mathcal{A} \star \mathcal{B}$ is continuous if both optimizers are continuous at certain points. Second, a convolution may be continuous if the right side is continuous and the left side generates convergent samples.

In order to support the following theorem, the property of bounded magnitude is introduced. An optimizer $\mathcal{G} \in \mathcal{MF}$ is of *bounded magnitude* if there exists a number $M < \infty$ such that $||\mathcal{G}[t, f]||_{\mathcal{M}} \leq M$ for all t, f. Otherwise, an unbounded sequence could cause a discontinuity. This condition is satisfied trivially for any optimizer in $\mathcal{PF}$.

THEOREM 8.1. *Let $\mathcal{S}, \mathcal{V} \in \mathcal{MF}$. Then $\mathcal{S} \star \mathcal{V}$ is continuous in objectives (or trajectories) at t, f if both $\mathcal{S}$ and $\mathcal{V}$ are of bounded magnitude, $\mathcal{S}$ is continuous in objectives (or trajectories) at t, f, and for some $C \in \mathcal{B}_\tau$ with $|\mathcal{S}[t, f]|(C) = |\mathcal{S}[t, f]|(X)$, $\mathcal{V}$ is continuous in objectives (or trajectories) at $t \cup x, f$ for all $x \in C$.*

PROOF. Assume that $\mathcal{S}$ and $\mathcal{V}$ are continuous in both objectives and trajectories at t, f. Fix $\epsilon > 0$. Suppose

$||\mathcal{S}[u,g]||_{\mathcal{M}} \leq M < \infty$ and $||\mathcal{V}[u,g]||_{\mathcal{M}} \leq M$ for all u, g. Let $f_n \to f$, $t_n \to t$. Let $A \in \mathcal{B}_\tau$. Then

$$
\begin{aligned}
&|\mathcal{S} \star \mathcal{V}[t_n, f_n](A) - \mathcal{S} \star \mathcal{V}[t, f](A)| \\
&= \left| \int_X \mathcal{V}[t_n \cup x, f_n](A) \mathcal{S}[t_n, f_n](dx) \right. \\
&\qquad \left. - \int_X \mathcal{V}[t \cup x, f](A) \mathcal{S}[t, f](dx) \right| \\
&\leq \left| \int_X \mathcal{V}[t_n \cup x, f_n](A) \mathcal{S}[t_n, f_n](dx) \right. \\
&\qquad \left. - \int_X \mathcal{V}[t \cup x, f](A) \mathcal{S}[t_n, f_n](dx) \right| \\
&\qquad + \left| \int_X \mathcal{V}[t \cup x, f](A) \mathcal{S}[t_n, f_n](dx) \right. \\
&\qquad \left. - \int_X \mathcal{V}[t \cup x, f](A) \mathcal{S}[t, f](dx) \right| \\
&\leq \int_C |\mathcal{V}[t_n \cup x, f_n](A) - \mathcal{V}[t \cup x, f](A)| |\mathcal{S}[t_n, f_n](dx)| \\
&\qquad + \int_X |\mathcal{V}[t \cup x, f](A)| |\mathcal{S}[t_n, f_n](dx) - \mathcal{S}[t, f](dx)| \quad (30) \\
&< \frac{\epsilon}{2} \frac{1}{M} |\mathcal{S}[t_n, f_n]|(C) + M |\mathcal{S}[t_n, f_n] - \mathcal{S}[t, f]|(X) \quad (31) \\
&< \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.
\end{aligned}
$$

To obtain Equation 31, use the fact that $\mathcal{V}$ is continuous on the left side and the fact that $\mathcal{V}$ is of bounded magnitude on the right. To obtain Equation 32, note that $\mathcal{S}$ is bounded in magnitude by M and that $\mathcal{S}$ is continuous. For Equation 31, continuity is sufficient to imply that there exists an N independent of x such that

$$|\mathcal{V}[t_n \cup x, f_n](A) - \mathcal{V}[t \cup x, f](A)| < \frac{\epsilon}{2} \frac{1}{M}$$

for all $n > N$ because $d_\rho(t_n \cup x, t \cup x) = d_\rho(t_n, t)$ for all $x \in X$, using d_ρ from Equation 29. This justification can be extended to general topological spaces; the details are not included here.

The proof above holds for jointly continuous $\mathcal{S}$ and $\mathcal{V}$; continuity in either objectives or trajectories separately can be proven by repeating the equations above with $t_n = t$ or $f_n = f$ as needed. □

Theorem 8.1 can be applied to evolutionary algorithms to deduce continuity based on the continuity of the selection rules, the crossover rule, and the mutation operator. There is a problem with this approach, however, since most selection and crossover rules are discontinuous in some sense. This problem can be circumvented by using the concept of *sample convergence* instead.

Definition 13. An optimizer $\mathcal{G} \in \mathcal{MF}$ is *sample-convergent* in trajectories at t, f if

1. there is a trajectory $u_{t,f} \in \mathcal{T}[X]$ s.t. $\{y \in u_{t,f}\}$ has full measure on $\mathcal{G}[t, f]$ and $|\mathcal{G}[t, f]|(\{y\}) > 0$ for each $y \in u_{t,f}$,
2. $t_n \to t$ implies $\exists u_{t_n,f}$ as in the prior statement and $u_{t_n,f} \to u_{t,f}$, and
3. $t_n \to t$ implies $\mathcal{G}[t_n, f](\{u^i_{t_n,f}\}) \to \mathcal{G}[t, f](\{u^i_{t,f}\})$ for all $1 \leq i \leq |u_{t,f}|$.

If the above statements hold when $t_n \to t$ is replaced with $f_n \to f$, then $\mathcal{G}$ is sample-convergent in objectives at t, f.

A sample drawn from a sample-convergent optimizer converges along a sequence of trajectories or objectives. That is, if $Y_{t,f} \sim \mathcal{G}[t, f]$ for all t, f, then Y_{t_n,f_n} converges in distribution to $Y_{t,f}$ when $t_n, f_n \to t, f$. Sample convergence can make a convolution be continuous.

THEOREM 8.2. *Suppose $\mathcal{G} \in \mathcal{MF}$. If $\mathcal{G}$ can be written as $\mathcal{A} \star \mathcal{B}$ where $\mathcal{A}$ and $\mathcal{B}$ are both of bounded magnitude, $\mathcal{A}$ is sample convergent in objectives (or trajectories) at t, f with trajectory $u_{t,f}$ as above, and $\mathcal{B}$ is continuous in objectives (or trajectories) at $t \cup x, f$ for all $x \in u_{t,f}$, then $\mathcal{G}$ is continuous in objectives (or trajectories) at t, f.*

PROOF. Without loss of generality, suppose $\mathcal{A}$ is sample convergent in both objectives and trajectories at t, f and that $\mathcal{B}$ is continuous in both objectives and trajectories at t, f. Fix $\epsilon > 0$ and suppose $\mathcal{A} \leq M < \infty$ and $\mathcal{B} \leq M$. The optimizer $\mathcal{G}$ can be written as

$$\mathcal{G}[t, f](A) = \sum_{i=1}^{|u_{t,f}|} \mathcal{A}[t, f](\{u^i_{t,f}\}) \; \mathcal{B}[t \cup u^i_{t,f}, f](A). \quad (32)$$

To reduce notation, let $N = |u_{t,f}|$, $p(i, t, f) = \mathcal{A}[t, f](\{u^i_{t,f}\})$, and $\nu_{i,t,f}(A) = \mathcal{B}[t \cup u^i_{t,f}, f](A)$. Then the above can be restated as

$$\mathcal{G}[t, f](A) = \sum_{i=1}^{N} p(i, t, f) \; \nu_{i,t,f}(A). \quad (33)$$

Suppose now that $t_n \to t$ and $f_n \to f$. Because $\mathcal{A}$ is sample convergent, it follows that $p(i, t_n, f_n) \to p(i, t, f)$. Also, $\nu_{i,t_n,f_n}(A) \to \nu_{i,t,f}$ since $\mathcal{B}$ is continuous. But then

$$
\begin{aligned}
&|\mathcal{G}[t, f](A) - \mathcal{G}[t_n, f_n](A)| \\
&\quad \leq \textstyle\sum_{i=1}^{N} |p(i, t, f)\nu_{i,t,f}(A) - p(i, t_n, f_n)\nu_{i,t_n,f_n}(A)| \\
&\quad \leq \textstyle\sum_{i=1}^{N} |p(i, t, f)\nu_{i,t,f}(A) - p(i, t, f)\nu_{i,t_n,f_n}(A)| \\
&\qquad + \sum_{i=1}^{N} |p(i, t, f)\nu_{i,t_n,f_n}(A) - p(i, t_n, f_n)\nu_{i,t_n,f_n}(A)| \\
&\quad = \textstyle\sum_{i=1}^{N} |p(i, t, f)| \, |\nu_{i,t,f}(A) - \nu_{i,t_n,f_n}(A)| \\
&\qquad + \sum_{i=1}^{N} |\nu_{i,t_n,f_n}(A)| \, |p(i, t, f) - p(i, t_n, f_n)| \\
&\quad \leq M \sum_{i=1}^{N} \left[\frac{\epsilon}{2NM} + \frac{\epsilon}{2NM} \right] = \epsilon \quad (34)
\end{aligned}
$$

where the next to last line follows from the convergence of p and ν mentioned above and from the bounded magnitude of $\mathcal{A}$ and $\mathcal{B}$. Thus $\mathcal{G}$ is continuous in both objectives and trajectories at t, f. To show $\mathcal{G}$ is only continuous in objectives or trajectories separately, repeat the above steps with $t_n = t$ or $f_n = f$. □

COROLLARY 8.3. *A quasi-evolutionary algorithm is continuous in objectives at t, f if its selection rules are sample-convergent in objectives at t, f.*

The next theorem states that masked crossover rules are sample-convergent if they have sample-convergent selection rules. Since most crossover rules are masked crossover rules, the continuity of most evolutionary algorithms depends only on the sample convergence of the selection rule.

THEOREM 8.4. *The convolution of a selection rule and a recombination operator with a masked crossover rule is sample-convergent in objectives (or trajectories) at t, f if its selection rules are also sample-convergent in objectives (or trajectories) at t, f.*

PROOF. Suppose X is a d-dimensional vector space, so that a masked crossover rule can be applied. Let $\mathcal{S}$ be a selection rule that is sample convergent in both trajectories and objectives. Let $\mathcal{R}$ be a recombination operator with a masked crossover rule. Then

$$\mathcal{S} \star \mathcal{R} = \mathcal{S} \star (\triangleleft \mathcal{S}_1 \star (\cdots \star (\triangleleft_{n-1} \mathcal{S}_{n-1} \star \mathcal{C} < \mathbb{P}_M >)))$$

for sample convergent selection rules $\mathcal{S}_1, \ldots, \mathcal{S}_{n-1}$ and a masked crossover rule $\mathcal{C} < \mathbb{P}_M >$ of order n. Assume for now that each selection rule is sample convergent in both objectives and trajectories at t, f. Let $\mathcal{S}_0 = \mathcal{S}$ to simplify the notation.

For all t, f there is a trajectory $u_{i,t,f}$ for each selection rule $\mathcal{S}_i$ with $i = 0, \ldots, n-1$ such that $\mathcal{S}_i[t, f](\{y \in u_{i,t,f}\}) = 1$. There are n^d possible crossover masks, and each selection rule can only select one of $|u_{i,t,f}|$ points. Thus there are exactly $n^d \prod_i |u_{i,t,f}| < \infty$ points that can result from recombination, and these points may be enumerated within a trajectory $\tilde{u}_{t,f}$, where the order of enumeration is independent of t and f. To be specific, for each position k in $\tilde{u}_{t,f}$ there is a crossover mask m^k and an index to a selected parent $p_{i,k}$ for each selection rule i such that m^k and $(p_{i,k})_{i=1}^n$ depend solely on the position k and not on t, f. Recalling Equation 23, $\tilde{u}^k_{t,f} = \sum_{i=1}^n m^k \otimes_i u^{p_{i,k}}_{i,t,f}$, and $\mathcal{S} \star \mathcal{R}[t, f](\{y \in \tilde{u}_{t,f}\}) = 1$.

Suppose $t_n \to t$ and $f_n \to f$. Then $u_{i,t_n,f_n} \to u_{i,t,f}$ for each selection rule $\mathcal{S}_i$. Let $x = \tilde{u}^k_{t,f}$, the k^{th} element of the trajectory $\tilde{u}_{t,f}$. Then x is generated from a particular crossover mask m determined by the position k. Suppose m has the value j in the ℓ^{th} component, i.e. $m_\ell = j$. Then $x_\ell = \left(u^k_{j,t,f}\right)_\ell$. Let $x^n = \tilde{u}^k_{t_n,f_n}$. Then because the enumeration order was fixed, $x^n_\ell = \left(u^k_{j,t_n,f_n}\right)_\ell$. Since $u_{j,t_n,f_n} \to u_{j,t,f}$, it follows that $x^n_\ell \to x_\ell$. But k, j, and ℓ were arbitrary, so it follows that $\tilde{u}_{t_n,f_n} \to \tilde{u}_{t,f}$.

Again, suppose $t_n \to t$ and $f_n \to f$. Let m be the crossover mask for $u^k_{t,f}$, and let $y_i = u^{p_{i,k}}_{i,t,f}$ be the point selected on t, f by the i^{th} selection rule at the k^{th} position in the enumeration. Observe that

$$\mathcal{S} \star \mathcal{R}[t, f](\{\tilde{u}^k_{t,f}\}) = \mathbb{P}_M(m) \prod_i \mathcal{S}_i[t, f](\{y_i\}). \tag{35}$$

Let $y^n_i = u^{p_{i,k}}_{i,t_n,f_n}$ be the point selected on t_n, f_n by the i^{th} selection rule at the k^{th} position in the enumeration and note that (1) $y^n_i \to y_i$, (2) $\tilde{u}^k_{t_n,f_n} \to \tilde{u}^k_{t,f}$, and (3) the particular mask m is a function of the position k independent of t, f. Since $\mathbb{P}_M(m)$ is independent of t, f and $\mathcal{S}_i[t_n, f_n](\{y^n_i\}) \to \mathcal{S}_i[t, f](\{y_i\})$ for all i, it follows that $\mathcal{S} \star \mathcal{R}[t_n, f_n](\{\tilde{u}^k_{t_n,f_n}\}) \to \mathcal{S} \star \mathcal{R}[t, f](\{\tilde{u}^k_{t,f}\})$. Therefore $\mathcal{S} \star \mathcal{R}$ is sample convergent at t, f. To show that $\mathcal{S} \star \mathcal{R}$ is only convergent in either trajectories or objectives, repeat the above with $f_n = f$ or $t_n = t$. □

COROLLARY 8.5. *An evolutionary algorithm with a masked crossover rule is continuous in trajectories (or objectives) at t, f if its selection rules are sample convergent in trajectories (or objectives) at t, f and its mutation operator is continuous in trajectories (or objectives) at $t \cup x, f$ for all x generated by masked crossover of elements in t.*

As a final piece of the puzzle, proportional selection is sample-convergent on $\overline{C[X]}$ under certain conditions. [3] Recall that $\mathcal{PS}\langle g \rangle$ from Equation 21 is generalized proportional selection with a modulating function g. The theorem below implies that the simple genetic algorithm is continuous.

THEOREM 8.6. *Proportional selection with modulating function g is sample convergent on all trajectories and all objectives in $\overline{C[X]}$ if g is continuous in both arguments and for every pair t, f with $f \in \overline{C[X]}$, $h_k(u, \tilde{f}) = g(u, \tilde{f}(H(u)^{-1,k}))$ is bounded on some neighborhood of t, f for all $1 \le k \le K$.*

PROOF. To make the proof simpler, use unnormalized proportional selection,

$$\mathcal{UPS} < g > [t, f](B) = \sum_{k=1}^{K} g(t, f\left(H(t)^{-1,k}\right)) 1_B(H(t)^{-1,k}), \tag{36}$$

noting that $H(t)^{-1}$ is a sequence that may repeat points.

Suppose $t_n \to t$ and $f_n \to f$. Without loss of generality, suppose f_n is continuous, as we may, since continuous functions are dense in $\overline{C[X]}$. Clearly, the set $P_{t,f} = \left\{y \in H(t)^{-1}\right\}$ has full measure on $\mathcal{UPS}[t, f]$ for all t, f, and $H(t_n)^{-1} \to H(t)^{-1}$ in X^K (or in $\mathcal{T}[X]$). It remains to show that

$$\mathcal{UPS}[t_n, f_n](\left\{H(t_n)^{-1,k}\right\}) \to \mathcal{UPS}[t, f](\left\{H(t)^{-1,k}\right\})$$

for all k. The definitions imply

$$\begin{aligned} &\left|\mathcal{UPS}[t_n, f_n](\left\{H(t_n)^{-1,k}\right\}) - \mathcal{UPS}[t, f](\left\{H(t)^{-1,k}\right\})\right| \\ &\quad = \left|g(t_n, f_n\left(H(t_n)^{-1,k}\right)) - g(t, f\left(H(t)^{-1,k}\right))\right|. \end{aligned} \tag{37}$$

Now f_n is continuous and $f_n \to f$, so for any $\epsilon > 0$,

$$\begin{aligned} &\left|f_n\left(H(t_n)^{-1,k}\right) - f\left(H(t)^{-1,k}\right)\right| \\ &\quad \le \left|f_n\left(H(t_n)^{-1,k}\right) - f_n\left(H(t)^{-1,k}\right)\right| \\ &\qquad + \left|f_n\left(H(t)^{-1,k}\right) - f\left(H(t)^{-1,k}\right)\right| \\ &\quad < \tfrac{\epsilon}{2} + \tfrac{\epsilon}{2} = \epsilon. \end{aligned} \tag{38}$$

Since g is continuous in both arguments, it follows that $\mathcal{UPS}$ is sample convergent at t, f. Because g is bounded near t, f, the desired conclusion follows by normalizing $\mathcal{UPS}$. □

COROLLARY 8.7. *The simple genetic algorithm of Equation 28 is jointly continuous in trajectories and objectives.*

PROOF. Recall that

$$\mathcal{SGA} < p > = (\mathcal{PS} < |x| > \star ((\triangleleft \mathcal{PS} < |x| >) \star \mathcal{SC})) \star \mathcal{B} < p >,$$

where the objective is assumed to be negative (for minimization). The search space is $\{0, 1\}^d$ with the discrete topology (i.e. all sets are open), and therefore $\overline{C[\{0, 1\}^d]} = \mathbb{R}^X$. The function $g(t, x) = |x|$ is continuous. The fitness function f is bounded, since f can only take finitely many values on X. Fix $\epsilon > 0$. Then for any f, the set $N^f_\epsilon = \{u, \tilde{f} \mid \sup_{x \in X} |\tilde{f}(x) - f(x)| < \epsilon\}$ forms a neighborhood of t, f on which $\tilde{f}$ and therefore g is bounded, and hence the neighborhood requirement of Theorem 8.6 is satisfied. Thus $\mathcal{PS} < |x| >$ is sample convergent everywhere by Theorem 8.6. The Bernoulli mutation operator $\mathcal{B} < p >$ is jointly

[3] $\overline{C[X]} \subseteq \mathbb{R}^X$ consists of all continuous real functions and their pointwise limits, including functions with jump discontinuities or point discontinuities.

continuous. Single-point crossover $\mathcal{SC}$ is a masked crossover rule, so Corollary 8.5 implies that $\mathcal{SGA}$ is jointly continuous everywhere. □

Genetic algorithms in any space are jointly continuous on all trajectories and objectives in $\overline{C[X]}$ when they use masked crossover and proportional selection with a continuous modulating function. For example, a real-coded genetic algorithm with proportional selection, uniform crossover, and Gaussian mutation is continuous in this way.

Proportional selection is no longer commonly used as a selection rule; it has been replaced by rank-based methods such as tournament selection. Whereas proportional selection makes a genetic algorithm continuous, rank-based selection is discontinuous at some points. The following subsection identifies these discontinuities, leading up to a full characterization of when these selection rules are continuous.

8.3 Discontinuity of Some Methods

Sample convergence was used to show that many genetic algorithms are continuous on a large set of objectives. These results also have a converse; selection rules whose samples diverge are a source of discontinuities in the optimizer.

Definition 14. An optimizer $\mathcal{G} \in \mathcal{MF}$ is *sample-divergent* in trajectories at t, f if

1. there is a trajectory $u_{t,f} \in \mathcal{T}[X]$ s.t. $\{y \in u_{t,f}\}$ has full measure on $\mathcal{G}[t,f]$ and $\mathcal{G}[t,f](\{y\}) > 0$ for each $y \in u_{t,f}$,
2. $t_n \to t$ implies $\exists u_{t_n,f}$ as in the prior statement, and $u_{t_n,f} \to u_{t,f}$,
3. $t_n \to t$ implies $\mathcal{G}[t_n,f](\{u^i_{t_n,f}\}) \nrightarrow \mathcal{G}[t,f](\{u^i_{t,f}\})$ for some $1 \le i \le |u_{t,f}|$.

If the above statements hold when $t_n \to t$ is replaced with $f_n \to f$, then $\mathcal{G}$ is sample-divergent in objectives at t, f.

The first two properties of sample divergence are identical to those for sample-convergence, but the final properties are opposites. Any optimizer $\mathcal{G}$ together with any pair t, f that meets the first two requirements must be either sample-convergent or sample-divergent and cannot be both, since the trajectory with full measure must be unique.

THEOREM 8.8. *Suppose $\mathcal{G} \in \mathcal{MF}$. If $\mathcal{G}$ can be written as $\mathcal{A} \star \mathcal{B}$ where $\mathcal{A}$ and $\mathcal{B}$ are both of bounded magnitude, $\mathcal{A}$ is sample divergent in objectives (or trajectories) at t, f, and $\mathcal{B}$ is continuous in objectives (or trajectories) at t, f, then $\mathcal{G}$ is discontinuous in objectives (or trajectories) at t, f provided that for all $x \in u_t, f$ for $\mathcal{A}$, $\mathcal{B}[t \cup x, f] \neq 0$.*

PROOF. Without loss of generality, assume that $\mathcal{A}$ is sample divergent in both objectives and trajectories at t, f and that $\mathcal{B}$ is continuous in both objectives and trajectories at t, f. Adopt notation for $\mathcal{G}$ as in Equation 33. Suppose $\mathcal{A} \le M < \infty$ and $\mathcal{B} \le M$. Then there exist sequences $t_n \to t$ and $f_n \to f$ and some i such that $p(i,t_n,f_n) \nrightarrow p(i,t,f)$, i.e. there is some constant $c_1 > 0$ such that for any N, there exists $m > N$ with $|p(i,t_m,f_m) - p(i,t,f)| \ge c_1$. There is also a set $A \in \mathcal{B}_\tau$ such that $|\nu_{i,t,f}|(A) = c_2 > 0$, since $\mathcal{B}[t \cup u^i_{t,f}, f] \neq 0$. The following inequalities hold:

$$\begin{aligned} ||\mathcal{G}[t,f] - \mathcal{G}[t_n,f_n]||_{\mathcal{M}} &\ge |\mathcal{G}[t,f](A) - \mathcal{G}[t_n,f_n](A)| \\ &\ge |p(i,t,f)\nu_{i,t,f}(A) - p(i,t_n,f_n)\nu_{i,t_n,f_n}(A)| \\ &\ge |[p(i,t,f) - p(i,t_n,f_n)]\,\nu_{i,t,f}(A) \\ &\quad + p(i,t_n,f_n)\,[\nu_{i,t,f}(A) - \nu_{i,t_n,f_n}(A)]|\,. \end{aligned} \tag{39}$$

Because $\mathcal{A}$ is bounded by M, $|p(i,t,f)| \le M$, and because $\mathcal{B}$ is continuous at t, f, there is some N_0 such that for $m > N_0$, $|\nu_{i,t,f}(A) - \nu_{i,t_m,f_m}(A)| < \frac{c_1 c_2}{2M}$. Therefore

$$|p(i,t,f)\,[\nu_{i,t,f}(A) - \nu_{i,t_m,f_m}(A)]| < \frac{c_1 c_2}{2}.$$

On the other hand, regardless of the value of N_0, m can be chosen so that $|p(i,t,f) - p(i,t_m,f_m)| \ge c_1$ and hence

$$|[p(i,t,f) - p(i,t_n,f_n)]\,\nu_{i,t,f}(A)| \ge c_1 c_2.$$

Consequently,

$$||\mathcal{G}[t,f] - \mathcal{G}[t_m,f_m]||_{\mathcal{M}} > c_1 c_2 - \frac{c_1 c_2}{2} = \frac{c_1 c_2}{2}, \tag{40}$$

and this inequality holds for any value of $N \ge N_0$. Thus $||\mathcal{G}[t,f] - \mathcal{G}[t_n,f_n]||_{\mathcal{M}}$ does not converge, and $\mathcal{G}$ is discontinuous at t, f. □

Theorem 8.4 stated that a masked crossover rule preserves sample convergence from its selection rules. As an analogue, masked crossover also preserves sample divergence. The following Theorem and Corollary can be proven in a similar way to Theorem 8.4 and its corollaries, and so the proofs are omitted.

THEOREM 8.9. *A recombination operator with a masked crossover rule is sample-divergent in objectives (or trajectories) at t, f if all of its selection rules are sample-divergent in objectives (or trajectories) at t, f.*

COROLLARY 8.10. *An evolutionary algorithm with a masked crossover rule is discontinuous in objectives (or trajectories) at t, f if all of its selection rules are sample-divergent in objectives (or trajectories) at t, f and its mutation operator is continuous in objectives (or trajectories) at $t \cup x, f$ for all x generated by masked crossover of elements in t.*

Theorem 8.6 showed that generalized proportional selection is sample-convergent where the modulating function is continuous on the image of the fitness. Conversely, proportional selection is sample-divergent when the composition of the modulating function and the objective is discontinuous.

THEOREM 8.11. *Proportional selection with modulating function g is sample divergent in objectives (or trajectories) at t, f whenever $h_k(u,\tilde{f}) = g(u, \tilde{f}(H(u)^{-1,k}))$ is discontinuous in objectives (or trajectories) at t, f and bounded away from zero on some neighborhood of t, f for all $1 \le k \le K$, i.e., $|h_k(u,\tilde{f})| > c > 0$.*

PROOF. Suppose $t_n \to t$ and $f_n \to f$ but $h_k(t_n,f_n) \nrightarrow h_k(t,f)$. As in the proof of Theorem 8.6, use unnormalized proportional selection, $\mathcal{UPS}$. Also as in that proof, $\mathcal{UPS}$ meets the basic requirements of sample divergence (or convergence), i.e. $u_{t,f} = H(t)^{-1}$ and $P_{t,f} = \{y \in H(t)^{-1}\}$ has full measure. Let $x^n_k = H(t_n)^{-1,k}$. Then

$$\begin{aligned} |\mathcal{UPS}[t_n,f_n](\{x^n_k\}) &- \mathcal{UPS}[t,f](\{x_k\})| \\ &= |h_k(t_n,f_n) - h_k(t,f)|\,. \end{aligned} \tag{41}$$

That is, $\mathcal{UPS}[t_n,f_n](\{x^n_k\}) \nrightarrow \mathcal{UPS}[t,f](\{x_k\})$. Because h_k is bounded away from zero, normalization yields that $\mathcal{PS}$ is sample divergent at t, f. □

Theorem 8.11 can be leveraged to conclude that tournament selection (and, by extension, any selection rule based

on rank) is sample-divergent on the majority of objectives for trajectories that have distinct points with the same fitness.

Definition 15. A trajectory $t \in \mathcal{T}[X]$ is of *ambivalent fitness* at degree K on a fitness function f if there exist points $x, y \in H(t)^{-1}$ for population size K with $x \neq y$ but $f(x) = f(y)$. Otherwise, t is of *unambivalent fitness* at degree K on f. The trajectory t is ambivalent at full degree if $K = |t|$; the degree may be omitted if clear from the context.

THEOREM 8.12. *Tournament selection (Equation 22) with selection pressure $q \in [0, 1)$ is sample divergent in objectives at every objective on trajectories of ambivalent fitness at the degree of the selection rule.*

PROOF. Let $R(y, f, P)$ be the ranking function of Section 7. Tournament selection over the whole population is proportional selection with $h_k(u, \tilde{f})$ in the statement of Theorem 8.11 given by

$$h_k(u, \tilde{f}) = (1 - q)^{R(H(u)^{-1,k}, \tilde{f}, H(u)^{-1})}. \tag{42}$$

This h_k is bounded away from zero everywhere, so Theorem 8.11 implies that tournament selection is sample divergent at the discontinuities of h_k. Now h_k is a continuous function of $R(H(u)^{-1,k}, \tilde{f}, H(u)^{-1})$, and thus its discontinuities are exactly the discontinuities of R.

Let f be any non-monotonic objective and let t be a trajectory of ambivalent fitness on f at the degree of the selection rule, so that there are two points y and z in $H(t)^{-1}$ with $y \neq z$ and $f(y) = f(z)$, where z occurs later in the population. Next, construct f_n so that $f_n(z) = f(z) + \frac{1}{n}$ and $f_n(x) = f(x)$ for all $x \neq z$. Then $f_n \to f$, and

$$R(z, f_n, H(t)^{-1}) - R(y, f_n, H(t)^{-1}) > 0$$

is a positive constant independent of n, i.e. y is ranked higher than z, and thus has a lower index in the ranked population. But according to the disambiguation rule in Section 7,

$$R(z, f, H(t)^{-1}) - R(y, f, H(t)^{-1}) < 0,$$

that is, y is ranked lower than z at the limit and has a higher index in the population. Therefore R is discontinuous in objectives at t, f, and by consequence tournament selection is discontinuous in objectives at t, f as well. If the tie-breaking procedure is reversed, the proof still holds by using $f_n(z) = f(z) - \frac{1}{n}$ instead. □

Trajectories of ambivalent fitness have measure zero in the optimization process generated by tournament selection, unless the fitness function has a plateau. Even on functions with many small plateaus, trajectories of ambivalent fitness will rarely be encountered. If trajectories of ambivalent fitness are avoided, then tournament selection is sample-convergent.

THEOREM 8.13. *Tournament selection is sample-convergent in objectives in $\overline{C[X]}$ at trajectories of unambivalent fitness.*

PROOF. As in the proof of Theorem 8.12, tournament and ranking selection are sample divergent at exactly the points where R is discontinuous. Let $f \in \overline{C[X]}$, and let t be a trajectory that is of unambivalent fitness on f at the degree of the selection rule. Assume $f_n \to f$. Then there is an N such that $R(x, f_n, H(t)^{-1}) = R(x, f, H(t)^{-1})$ for $n > N$, since the population size K is finite, and any finite set of points in $\mathbb{R}$ can be separated by disjoint open sets. But then R is continuous on f at t, and therefore tournament and ranking selection are sample convergent by Theorem 8.6 using h_k from the proof of Theorem 8.12. □

Thus a genetic algorithm with tournament selection is continuous almost surely on fitness functions without plateaus; the same fact holds for any rank-based selection mechanism, e.g. ranking selection or truncation selection. These facts provide a complete picture of exactly when common genetic algorithms can be expected to behave similarly on similar problems. This result is particularly useful, since it provides theoretical justification for methods that approximate fitness functions such as metamodeling or surrogates in evolutionary computation.

9. DISCUSSION AND FUTURE WORK

This paper presented a formal measure-theoretic approach that identifies iterative stochastic optimizers as a closed, convex subset of a normed vector space. This approach provides a rich setting within which powerful theorems about optimization methods can be proven. It has been shown in detail how genetic algorithms fit in to this framework. To demonstrate a first taste of what is possible using these theoretical tools, a full characterization of the continuity of genetic algorithms was provided.

These results are a promising beginning of what can potentially be proven with these tools. This formal framework can be used for many innovative tasks: to define and reason about explicit performance criteria; to prove generalized NFL theorems, including necessary and sufficient conditions for arbitrary fitness priors; and to establish rigorously a mathematical duality between problems and the optimizers. See [11] for a more detailed discussion of these matters.

The convexity of optimizer space as proven in this paper suggests that better performance on a particular problem may be achieved by blending two known optimizers convexly; this claim is supported by preliminary experiments [11]. The optimizer-to-objective duality that can be defined based on these results can be used as a starting point for meta-optimization or to examine the question of optimal optimization for a given problem class. These topics and others will be explored in future work.

10. CONCLUSION

Overall, a rigorous formal approach to optimization may lead to new optimization methods and new ways of configuring and improving existing methods. Such an approach can help to organize the profusion of new and diverse optimization methods as well as guide new research. The theory in this paper constitutes a first step towards these goals.

11. REFERENCES

[1] AUGER, A., AND TEYTAUD, O. Continuous lunches are free! In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO-2007)* (New York, 2007), ACM Press.

[2] BILLINGSLEY, P. *Probability and Measure*. John Wiley, 1986.

[3] Cohn, D. *Measure Theory*. Birkhauser, Boston, MA, 1980.

[4] El-Beltagy, M., Nair, P. B., and Keane, A. J. Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations. In *GECCO'99* (1999), pp. 196–203.

[5] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[6] Halmos, P. *Measure Theory*. Springer-Verlag, New York, NY, 1974.

[7] Jin, Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation 1*, 2 (2011), 61 – 70.

[8] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science 220*, 4598 (1983).

[9] Lehman, J., and Stanley, K. O. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation 19*, 2 (2011).

[10] Lockett, A., and Miikkulainen, R. Real-space evolutionary annealing. In *Proceedings of the 2011 Genetic and Evolutionary Computation Conference (GECCO-2011)* (2011).

[11] Lockett, A. J. *General-Purpose Optimization Through Information Maximization*. PhD thesis, University of Texas at Austin, 2012.

[12] Munkres, J. R. *Topology*. Prentice Hall, Upper Saddle River, NJ, 2000.

[13] Radcliffe, N., and Surry, P. D. Fundamental limitations on search algorithms: Evolutionary computing in perspective. In *LECTURE NOTES IN COMPUTER SCIENCE 1000* (1995), Springer-Verlag, pp. 275–291.

[14] Rowe, J. E., Vose, M. D., and Wright, A. H. Reinterpreting no free lunch. *Evolutionary Computation 17*, 1 (2009).

[15] Schaul, T., Sun, Y., Wierstra, D., Gomez, F., and Schmidhuber, J. Curiosity-Driven Optimization. In *IEEE Congress on Evolutionary Computation (CEC)* (2011).

[16] Schumacher, C., Vose, M. D., and Whitley, L. D. The no free lunch and problem description length. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001* (2001), Morgan Kaufmann, pp. 565–570.

[17] Vose, M. *The Simple Genetic Algorithm*. MIT Press, Cambridge, Massachusetts, 1999.

[18] Vose, M. D. Random heuristic search. *Theoretical Computer Science 229* (1999), 103–142.

[19] Wolpert, D. H., and Macready, W. G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation 1*, 1 (1997).