A Formal Model for Reasoning About the Ideal Fitness in Evolutionary Processes

Thomas Gabor(✉) and Claudia Linnhoff-Popien

LMU Munich, Munich, Germany
thomas.gabor@ifi.lmu.de

Abstract. We introduce and discuss a formal model of evolutionary processes that summarizes various kinds of evolutionary algorithms and other optimization techniques. Based on that framework, we present assumptions called "random agnosticism" and "based optimism" that allow for new kinds of proofs about evolution. We apply them by providing all a proof design that the recently introduced notion of final productive fitness is the ideal target fitness function for any evolutionary process, opening up a new perspective on the fitness in evolution.

Keywords: Evolution · Evolutionary algorithms · Fitness

1 Introduction

Evolution in its broadest sense describes a process that finds solutions to complex problems via the application of comparatively simple local operators. Mostly, this process can be described as a search that starts quite uninformed and uses the knowledge gained through trial and error to guide the further search process. Note that usually this happens without central control and mostly without even any central viewpoint that would allow to overlook all parts of the evolution. However, evolution is often implemented deliberately (using evolutionary algorithms in software, e.g.) in order to search or optimize for a specific result according to an externally given target.

While this target is often provided directly to the evolutionary process so that intermediate results may be evaluated, many studies empirically show better results when using slightly different goal than going directly for the external target metric. Our recent study [8] has brought up empirical evidence that one such "indirect" metric (called *final productive fitness*) might be theoretically optimal (even when or perhaps because it is extremely costly to compute). However, little formal framework exists to reason about evolutionary processes (specifically goals in evolutionary processes) at such a broad level in order to formally prove a claim of optimality.

The aim of this paper is to show what kind of formal framework *would be sufficient* to produce a formal proof of final productive's fitness optimality. To this end, we first introduce two bold but crucial assumptions hat allow us to strip

T. Margaria and B. Steffen (Eds.): ISoLA 2020, LNCS 12477, pp. 473–490, 2020.
https://doi.org/10.1007/978-3-030-61470-6_28

away much of the complexity of reasoning about evolution. Then we construct the desired proof from them to show how they work. We hope that the novel tools (i.e., mainly Assumptions 1 and 2) designed here can be used for other high-level arguments about evolution.

All necessary definitions involving evolutionary processes are given in a consistent way in Sect. 2. Section 3 then discusses the issue of the ideal fitness function and introduces the tools to reason about it. We give a short glance at related work in Sect. 4 and conclude with Sect. 5.

2 Definitions

We follow the formal framework sketched in [8] to a vast extent but substantially expand it in generality. We provide an example in Sect. 2.3.

2.1 Evolutionary Processes

For all definitions, we aim to give them in such a basic form that they can span over various disciplines, from biology to formal methods. We use $\mathfrak{P}$ to denote the power set.

Definition 1 (Evolution)**.** *Let $\mathcal{X}$ be an arbitrary set called* search space*. Let $g \in \mathbb{N}$ be called the* generation count*. Let $X_i \subseteq \mathcal{X}$ for any $i \in \mathbb{N}, 0 \leq i \leq g$ be a subset of $\mathcal{X}$ called* population*. Let $E : \mathfrak{P}(\mathcal{X}) \to \mathfrak{P}(\mathfrak{P}(\mathcal{X}))$ be a function called* evolutionary function*.*

A tuple $(\langle X_i \rangle_{0 \leq i \leq g}, E)$ is called an evolution *over $\mathcal{X}$ iff $X_i \in E(X_{i-1})$ for all $i \in \mathbb{N}, 1 \leq i \leq g$.*

Any element of the search space $x \in \mathcal{X}$ is called solution candidate (or sometimes just solution for short). Members of a given population $x \in X$ are obviously always solution candidates, but are often also called individuals. Every i within the generation count $1 \leq i \leq g$ is called a generation number with X_i being the respective generation's population. If no confusion is possible, both i and X_i will also be called a generation. X_0 is called the initial population.

Note that an evolution can be generated given a configuration consisting of a search space $\mathcal{X}$, an initial population X_0 and an evolution function E. However, many possible evolutions can follow from the same configuration. Often, the initial population X_0 is not given as a set but instead generated (semi-)randomly. We write that as an initialization function $I : \mathfrak{R} \to \mathfrak{P}(\mathcal{X})$ where $\mathfrak{R}$ stands for random inputs.[1] Notation-wise, we omit random inputs and write $X_0 \sim I()$ (or simply $X_0 = I()$ if no confusion is possible) for the initial population generated by such a function.

Definition 2 (Target)**.** *Let $\mathcal{X}$ be a search space. A function $t : \mathcal{X} \to [0; 1]$ that assigns all elements in the search space a scalar value is called a* target function*.*

[1] In computers, these are often provided by a seed value and a hash function.

A target function assigns a value to each point in the search space, i.e., to any given solution candidate.[2] We assume that target values are bounded, so w.l.o.g. we can assume the target value space to be restricted to $[0;1]$ in Definition 2. Again this can be generalized but is rarely useful in praxis. Also note that target functions themselves are unconstrained: They can always be applied to the whole search space. Hard constraints must be implemented by altering the search space or "softening" them by representing them with different target values.

Furthermore, w.l.o.g. we assign every goal function a minimization semantic: For two solution candidates $x_1, x_2 \in \mathcal{X}$ we say that x_1 fulfills a goal t better iff $t(x_1) < t(x_2)$. Any solution candidate $x \in \mathcal{X}$ so that $t(x) \leq t(x') \quad \forall x' \in \mathcal{X}$ is called a global optimum. An algorithm searching for increasingly better solutions candidates is called an optimization algorithm. A configuration, a target function and an evolution form an evolutionary process:

Definition 3 (Evolutionary Process)**.** *Let $\mathcal{X}$ be a search space. Let $E : \mathfrak{P}(\mathcal{X}) \to \mathfrak{P}(\mathfrak{P}(\mathcal{X}))$ be an evolutionary function. Let $t : \mathcal{X} \to [0;1]$ be a target function. Let X_i be a population for any $i \in \mathbb{N}, 0 \leq i \leq g$.*

A tuple $\mathcal{E} = (\mathcal{X}, E, t, \langle X_i \rangle_{i \leq g})$ is an evolutionary process *iff $(\langle X_i \rangle_{i \leq g}, E)$ is an evolution.*

Effectively, an evolutionary process consists of a history of past populations (X_i) and the means to generate new population (E). We often implement the evolutionary function by giving an *evolutionary step function* $e : \mathfrak{P}(\mathcal{X}) \times \mathfrak{R} \to \mathfrak{P}(\mathcal{X})$ and write $X_{i+1} \sim e(X_i)$ (or simply $X_{i+1} = e(X_i)$ if no confusion is possible) for any population X_{i+1} that evolved from X_i by applying the evolutionary step function alongside with some (omitted) random input.

An evolutionary process also carries a target function t. An evolutionary process $\mathcal{E}$ is *optimizing* iff $\min_{x \in X_0} t(x) \geq \min_{x' \in X_g} t(x')$. For many mechanisms in stochastic search as well as for more natural phenomena like biological evolution or human software development processes, optimization is a rather strong property. However, if we have sufficient space within a population and access to the target function, we can turn all evolutionary processes into optimizing ones by just saving the currently best individual alongside the evolution, i.e., ensuring that $\arg\min_{x \in X_i} t(x) \in X_{i+1}$.

Definition 4 (Elitism)**.** *An evolutionary process $\mathcal{E} = (\mathcal{X}, E, t, \langle X_i \rangle_{i \leq g})$ is called* elitist *iff for all $i \in \mathbb{N}, 1 \leq i \leq g$, it holds that $\min_{x \in X_{i-1}} t(x) \geq \min_{x' \in X_i} t(x')$.*

All elitist processes are optimizing. If not noted differently, we from now on assume every evolutionary process to be elitist by default.

[2] Note that by giving a function only parametrized on the individual itself, we assume that the target function is static. Dynamic optimization is an entire field of research that we heavily use in this paper. However, we leave dynamic target functions in our formalism to future work.

2.2 Evolutionary Algorithms

An evolutionary algorithm is special case of evolutionary process that uses an evolutionary function made up of a number of standard components called *evolutionary operators*. We now introduce standard definitions for these components that most instances of evolutionary algorithms can be mapped to. However, the field of evolutionary algorithms is vast and there are variants that alter many smaller details of how they work. It is interesting to note how robust the general concept of evolution is to such variations.

Nearly all evolutionary algorithms that use set-based populations introduce a fixed population size $n \in \mathbb{N}$ for all generations. This allows to keep memory resources easily manageable as the overall memory consumption will not increase over time. We also use this opportunity to introduce the concept of fitness functions. Note that $\mathfrak{E}$ is the space of all evolutionary processes.

Definition 5 (Fitness)**.** *Let $\mathcal{X}$ be a search space. A function $f : \mathcal{X} \times \mathfrak{E} \times \mathfrak{R} \rightarrow [0; 1]$ is called a* fitness function*. This function takes an individual, its evolutionary process up until now, and random input and returns a scalar value.*

The fitness function can be regarded as generalization of the concept of a target function (cf. Definition 2). It represents the goal definition that the evolutionary process can call upon and actively follows, which may or may not coincide with the target function. In addition to the solution candidate itself, it is able to process additional information about the context. Various approaches may allow nearly arbitrary information here. For a rather general approach, we just pass on a snapshot of the evolutionary process that generated the individual until now. This includes:

- The current population that the evaluated individual is a part of allows to define the fitness of an individual relative to its peers.
- The history of all populations until now allows to observe relative changes over time as well as trace the ancestry of individuals throughout evolution.
- The number of the current generation allows the fitness function to change over time and implement, e.g., a cool-down schedule.

Note that the random input that is also passed along allows fitness functions to also vary fitness values stochastically. However, most fitness functions will not make use of all this information. In these cases we allow to trim down the fitness function's signature and simply write $f(x)$ for an individual $x \in \mathcal{X}$ if all other parameters are ignored.

In many practical instances, developers will choose the target function as a fitness function, i.e., $f(x) = t(x)$ for all $x \in \mathcal{X}$, and for most target functions, evolution will end up achieving passable target values this way. It is the main point of this paper, however, to prove that the optimal choice in general is a different function derived from the target function.

Alongside the fitness function f an evolutionary algorithm also uses various selection functions. In general, a selection function returns a subset of the population for a specific purpose.

Definition 6 (Selection)**.** *A function* $s : \mathfrak{P}(\mathcal{X}) \times \mathfrak{E} \times \mathfrak{R} \to \mathfrak{P}(\mathcal{X})$ *is called a* selection function *iff* $s(X, \mathcal{E}, r) \subseteq X$ *for all* $X, \mathcal{E}, r$*. This function takes a population, its evolutionary process, and random input and returns a subset of the given population.*

Again note that we allow for a multitude of information that will rarely be used directly in any selection function and that will be omitted if not necessary. Most importantly, however, any selection function is able to call any fitness function since all its inputs can be provided.

As seemingly limitless variations of selection functions exist we use this opportunity to provide a few examples and at the same time define all families of selection functions that we use for the remainder of this paper. (Note that the current population X is always provided with an evolutionary process $\mathcal{E}$.)

Random Selection. This function $\varrho^m(X, \mathcal{E}, r) = \{x \sim X\} \cup \varrho^{m-1}(X, \mathcal{E}, r)$ selects m individuals of the population at random. Note that $x \sim X$ is one element $x \in X$ sampled uniformly at random. We define $\varrho^0(X, \mathcal{E}, r) = \emptyset$.

Cutoff Selection. This function $\sigma^m(X, \mathcal{E}, r) = \{\arg\min_{x \in X} f(x, \mathcal{E}, r)\} \cup \sigma^{m-1}(X, \mathcal{E}, r)$ selects the m best individuals according to the fitness function f. We define $\sigma^0(X, \mathcal{E}, r) = \emptyset$.

We can now move on to define the evolution function E. For all variants of evolutionary algorithms there exist certain building blocks, called *evolutionary operators*, that most evolutionary functions have in common. They take as arguments some individuals and return some (possibly new) individuals. During the execution of an evolutionary operator its input individuals are referred to as *parents* and its output individuals are referred to as *children*.

Mutation. This operator $mut : \mathcal{X} \times \mathfrak{R} \to \mathcal{X}$ generates a randomly slightly altered individual from a parent.

Recombination. This operator $rec : \mathcal{X} \times \mathcal{X} \times \mathfrak{R} \to \mathcal{X}$ takes two individuals to combine them into a new individual.

Migration. This operator $mig : \mathfrak{R} \to \mathcal{X}$ generates a random new individual.

Again, countless variants and implementations exist, most importantly among them there is non-random mutation and recombination with various amounts of parents and children. For brevity, we omit everything we do not use in this paper's study. Please note that all of these operators return entirely new individuals and leave their parents unchanged. In practical applications, it is equally common to apply (some of) these operators *in-place*, which means that the generated children replace their parents immediately. We, however, opt to just add the children to the population (and possibly eliminate the parents later) so that parents and their children can exist side by side within the same generation. Our main aim in doing this is that it makes elitism much easier to achieve.

As these operators work on single individuals, we define a shortcut to apply them to sets of individuals:

Definition 7 (Application of Operators)**.** *Let $X \subseteq \mathcal{X}$ be a set of individuals in search space $\mathcal{X}$. Let s be a selection function. We write $X \mid_{mut} s = \{mut(x) \mid x \in s(X)\}$ and $X \mid_{rec} s = \{rec(x_1, x_2) \mid x_1 \in s(X), x_2 \sim X\}$ for the sets of children when applying the respective operators. For consistency, we also write $X \mid_{mig} s = \{mig() \mid x \in s(X)\}$ to create $|s(X)|$ many new random individuals, even though their values do not depend on the individuals in X.*

We are now ready to define a scheme for the evolution function E in evolutionary algorithms. We do so by providing an evolutionary step function e as discussed above with parameters $A_1, A_2, A_3 \in \mathbb{N}$:

$$e(X) = \sigma^{|X|}(X \cup (X \mid_{rec} \sigma^{A_1}) \cup (X \mid_{mut} \varrho^{A_2}) \cup (X \mid_{mig} \varrho^{A_3})) \tag{1}$$

Note again that in this evolutionary step we place all generated children alongside their parents into one population and then cutoff-select the best from this population.[3] As it is common, we use random selection to select mutation parents. The selection function for the recombination parents is also called *parent selection*. We use cutoff selection on one parent with a randomly selected partner here. This gives some selective pressure (i.e., better individuals have a better chance of becoming recombination parents) without overcentralizing too much. Although many approaches to parent selection exist, we choose this one as it is both effective in practical implementations and mathematically very clean to define. The final selection function that is called upon the combined population of potential parents and new children is called *survivor selection*. We simply use cutoff selection here for ease of reasoning. Many evolutionary algorithms use more advanced survivor selection functions like roulette wheel selection where better individuals merely have a higher chance of being picked. We choose a hard cutoff for this kind of selection, mainly because it is simpler to define and understand, and its transparent to elitism. Since the cutoff point varies with the population's fitness structure that is subjected to random effects, the practical difference between both approaches for our examples is negligible. Note that we can emulate a lot of different selection schemes by choosing an appropriate fitness function: As the fitness function can vary at random, we can for example make the cutoff more fuzzy by simply adding noise to each fitness evaluation instead of changing the selection function. Also note that adding all the children non-destructively and using cutoff-selection makes the algorithm elitist if $f = t$.

We parametrize the evolutionary step function with the amount of recombination children A_1, amount of mutation children A_2 and amount of migration children A_3. These are also often given as rates relative to the population size.

Definition 8 (Evolutionary Algorithm)**.** *An evolutionary algorithm is an evolutionary process $\mathbb{E} = (\mathcal{X}, E, t, \langle X_i \rangle_{i \leq g})$ where the evolutionary function is given via an evolutionary step function of the form described in Eq. 1, where a fitness function f is used for all selection functions and evolutionary operators and the target function t is only accessible insofar it is part of f.*

[3] In the field of evolutionary computing, this is called a $\mu + \lambda$ selection scheme.

Note that for the ease of use in later notation, we will often denote two evolutionary processes that differ solely in their fitness function (ϕ vs. ψ, e.g.) by denoting that fitness function as a subscript ($\mathcal{E}_\phi$ vs. $\mathcal{E}_\psi$). Independently of that we denote the best individual of the final generation of $\mathcal{E}_\phi$ according to some fitness or target function ψ with

$$|\mathcal{E}_\phi|_\psi = \arg\min_{x \in X_g} \psi(x) \tag{2}$$

and the best of all generations with

$$||\mathcal{E}_\phi||_\psi = \arg\min_{\substack{x \in X_i \\ i \in \mathbb{N} \\ 0 \leq i \leq g}} \psi(x). \tag{3}$$

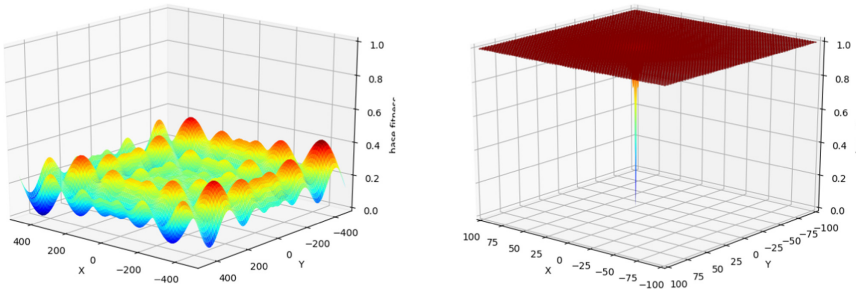
It is clear that if $\mathcal{E}_\phi$ is elitist with respect to ψ, then $|\mathcal{E}_\phi|_\psi = ||\mathcal{E}_\phi||_\psi$. Note that when we use a fitness function $f \neq t$ then we usually obtain the overall result of the evolutionary algorithm by computing $||\mathcal{E}_f||_t$ or $|\mathcal{E}_f|_t$ if we are confident about the elitism at least to the extent that we do not worry about substantial results getting lost along the way. In most cases we will assume that if f is close enough to t at least in the final generations, elitism with respect to f grants us quasi-elitism with respect t, i.e., if $f \approx t$ and $\mathcal{E}_f$ is elitist with respect to f, we assume that $||\mathcal{E}_f||_t \approx ||\mathcal{E}_f||_f$.

2.3 Example

We provide a running example accompanying these definitions.[4] For a target function, we choose two common benchmark functions from literature as they are implemented in the DEAP framework [2,12]. The first problem is based on the two-dimensional Schwefel function although we adjusted the target value space to fit comfortably within $[0; 1]$ (cf. Fig. 1a). We chose only two dimensions for ease of visualization. Higher-dimensional Schwefel is also covered in [8]. The Schwefel function is characterized by many valleys and hills of varying depth. The global optimum is at $X = Y \approx 420$. By contrast, our second example is the H1 function [14] that features one very distinct global optimum at $X = 8.6998, Y = 6.7665$. However, it feature very many little (hard to see) local optima throughout the whole surface. We took the classical H1 function, which is defined as a maximization problem and turned it upside down to produce a minimization problem (cf. Fig. 1b). For both target functions $t \in \{t_{Schwefel}, t_{H1}\}$ we construct the same evolutionary algorithm.

The search space is given as $\mathcal{X}_{Schwefel} = [-500; 500] \subseteq \mathbb{R}^2$ and $\mathcal{X}_{H1} = [-100; 100] \subseteq \mathbb{R}^2$ respectively. We initialize the search by generating X_0 from 25 random samples within the search space in both cases. The size of this population remains constant with application of the evolutionary step function e, which is constructed according to Eq. 1 with $A_1 = 0.3 \cdot |X|, A_2 = 0.1 \cdot |X|, A_3 = 0.1 \cdot |X|$.

[4] The code for all examples can be found at github.com/thomasgabor/isola-evolib.



(a) Normalized two-dimensional Schwefel (b) Inverse normalized H1

Fig. 1. Benchmark target functions used for the running example.

Let w be the range of a single dimensional value in the search space (i.e., $w_{Schwefel} = 1000, w_{H1} = 200$), then the mutation operator returns

$$mut((X,Y)) \in \{(X \oplus \delta, Y), (X, Y \oplus \delta) \mid \delta \in [-0.1w; 0.1w]\} \tag{4}$$

chosen random uniform where $\oplus$ only adds or subtracts as much of its second argument so that the resulting value remains within the search space. We further define the recombination operator so that its result is at random uniform picked from

$$rec((X,Y),(X',Y')) \in \{(X,Y),(X,Y'),(X',Y),(X',Y')\}. \tag{5}$$

Note that both operators include random cases where the operator does not do anything at, which does not harm the overall search and can be further counter-acted by increasing the respective amount of selected individuals for that operator. The migration operator just samples random uniform from the search space, returning $mig() \in \mathcal{X}$.

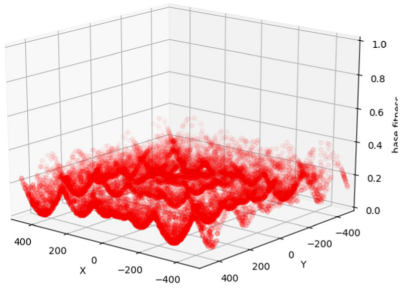
To illustrate the behavior of evolution, we ran independently initialized evolutionary processes for each problem 500 times each for 50 generations. Figure 2 shows all solution candidates found within a specific generation among *all* evolutionary processes. We can clearly trace how the search start random uniform and then focuses towards the global optima, sometimes getting stuck in local optima in the target value landscape (compare Fig. 1).

3 Approach

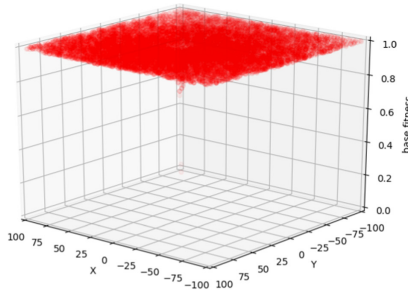
We apply the framework to give the definition of productive fitness. To present the full proof design we introduce and discuss Assumptions 1 and 2. We continue our example in Sect. 3.3.

3.1 The Ideal Fitness

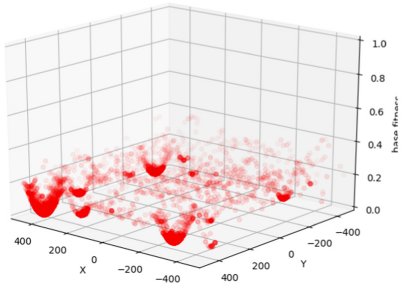
So far, we discussed some example definitions using the target function as fitness, $f = t$, and noted that it works (but not optimally). Obviously, having f correlate



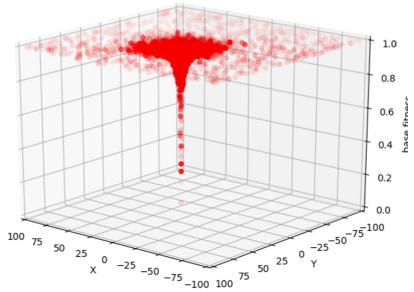
(a) Schwefel, generation 1



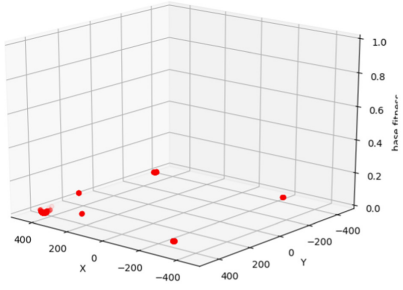
(b) H1, generation 1



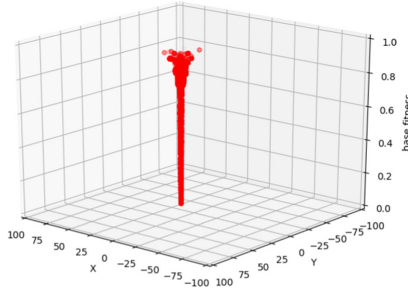
(c) Schwefel, generation 10



(d) H1, generation 10



(e) Schwefel, generation 50



(f) H1, generation 50

Fig. 2. Individuals from 500 independent runs of the evolutionary processes.

to some extend with t is a good thing if in the end we value our results with respect to t. However, it has long been known that augmenting the fitness with additional (meta-)information can greatly aid the optimization process in some cases. This fact is extensively discussed in literature [1,15] including previous works by the authors [7,8]. We sum the results up in the following observation:

Observation 1. *There exist evolutionary processes* $\mathcal{E}_\phi = (\mathcal{X}, E_\phi, t, \langle X_i \rangle_{i \leq g})$ *and* $\mathcal{E}_t = (\mathcal{X}, E_t, t, \langle X'_i \rangle_{i \leq g})$ *whose configurations only differ in the fitness function and there exist fitness functions* $\phi \neq t$ *so that* $||\mathcal{E}_\phi||_t < ||\mathcal{E}_t||_t$.

Observation 1 states that an evolutionary process can yield better results *with respect to* t by *not using* t directly but a somewhat approximate version of t given via ϕ, which includes additional information but likewise "waters down" the pure information of our original target. It is somewhat surprising that a deviation from the original target can yield an improvement. Commonly this phenomenon is explained by the *exploration/exploitation trade-off*: In an unknown solution landscape made up by t, we gain knowledge through evaluating solution candidates. When we have evaluated all solution candidates $x \in \mathcal{X}$, we simply need to compute $\arg\min_{x \in \mathcal{X}} t(x)$, which of course is infeasible for most practical search spaces. Giving limited time resources, we need to decide if we put additional effort into exploring more and new parts of the search space in hope of finding valuable solution candidates there or if we exploit the knowledge we have already gathered to further improve the solution candidates we already evaluated. This can be seen of a trade-off between large-scale search for exploration and small-scale search for exploitation.

Dealing with the exploration/exploitation trade-off certainly is one of the central tasks when implementing metaheuristic search and has been covered extensively in literature. Many of these approaches have been discovered bottom-up, often by analogy to biological or physical processes. Even though many similarities between approaches have been discovered, there does not exist a general framework for how to construct the right fitness function for a specific target function and evolutionary process.

Problem 1. *Given a target function* t*, what is the theoretically best fitness function* ϕ^* *for an evolutionary process* $\mathcal{E}_{\phi^*}$ *to optimize for* t*, i.e., optimize* $||\mathcal{E}_{\phi^*}||_t$*?*

We gave an answer to that question for the special case of standard evolutionary algorithms in [8]: We defined a measurement called *final productive fitness* and have sketched a proof that it represents the ideal fitness function for evolutionary algorithms. However, it is important to note that computing it a priori is infeasible. We approximated final productive fitness for an evolution a posteriori and provided empirical evidence that evolutionary algorithms are working better the better their fitness approximates final productive fitness.

In this paper, we formally introduce the necessary tools to provide the full proof of the ideal fitness for evolutionary algorithms. First, we need to re-iterate a few definitions of [8] in order to formally define final productive fitness.

Definition 9 (Descendants [8])**.** *Given an individual* x *in the population of generation* i*,* $x \in X_i$*, of an evolutionary process* $\mathcal{E}$*. All individuals* $x' \in X_{i+1}$ *so that* x' *resulted from* x *via a mutation operator, i.e.,* $x' = mut(x, r)$ *for some* $r \in \mathfrak{R}$*, or a recombination operator with any other parent, i.e., there exists* $y \in X_i$ *so that* $x' = rec(x, y, r)$ *for some* $r \in \mathfrak{R}$*, are called* direct descendants *of* x*. Further given a series of populations* $(X_i)_{0<i<g}$ *we define the set of all descendants* D_x *as the transitive hull on all direct descendants of* x*.*

The main idea behind productive fitness is to measure an individual's effect on the optimization process. If the optimization process is stopping right now, i.e., if we are in the final generation g, then we can equate any individual's effect with its target function value. However, for any previous generations an individual's effect on the optimization corresponds to the best target function values that its descendants have achieved within the evolution.

Definition 10 (Productive Fitness [8])**.** *Given an individual x in the population of generation i, $x \in X_i$, of an evolutionary process $\mathcal{E}$. Let $D_x \subseteq \mathcal{X}$ be the set of all descendants from x. The* productive fitness after n generations *or* optimistic n-productive fitness ϕ_n^+ *is the average achieved target value of x's descendants n generations later, written*

$$\phi_n^+(x) = \begin{cases} \text{avg}_{x' \in D_x \cap X_{i+n}}\, t(x') & \text{if } D_x \cap X_{i+n} \neq \emptyset \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

Note that in case the individual x has no descendants in n generations, we set its productive fitness $\phi_n^+(x)$ to a worst case value of 1.

From [8] we repeat two major arguments against this definition:

- The use of avg as an aggregator over target values might be a bit pessimistic. By doing so, we penalize an individual's fitness if that individual bloats up the optimization with many low-value individuals. However, if it thereby also delivers at least one superior descendant, we should actually be fine with that when we only care about the end result. If such effects actually occur in practical scenarios is up to future work to discover. Empirical evidence discovered in [8] strongly argues in favor of using the average, which is why we repeat it in this definition. In the proof we will later derive a min-version from one of our assumptions.
- Assigning the value 1 in case the given individual has no further descendants in generation $i + n$ is a design choice. We might leave the productive fitness in this case undefined or at least assign a value outside the common range of target function values. We suggest that even without living descendants there might still be inherent value to having explored certain solution candidates (and having them clearly discarded for the ongoing process). Still, determining this incentive is up to future research.

Of course, productive fitness ϕ_n^+ only measures the effect locally after a fixed amount of generations. For the effect for the whole evolution we can now easily define the notion of final productive fitness.

Definition 11 (Final Productive Fitness)**.** *Given an individual x in the population of generation i, $x \in X_i$, of an evolutionary process $\mathcal{E}$ with g generations in total. The final productive fitness of x is the fitness of its descendants in the final generation, i.e.,*

$$\phi^\dagger(x) = \phi_{g-i}^+(x). \tag{7}$$

Described shortly, the final productive fitness of an individual x can be seen as an answer to the question: "How much did x contribute to the fitness of the individuals of the final population?" We claim that optimizing for that measurement results in the optimal evolutionary process (as considered in Problem 1).

Practically, of course, optimizing for that measurement is rather difficult (which in fact may be the entire reason it is the optimal fitness function): To make the completely right decision in generation $i = 1$, we would have to evaluate all possible future generations for each single individual being involved in any selection or altered in any way by evolutionary operators. Within a single generation, these are exponentially many possibilities, which of course grow exponentially with each generation. Still, in [8] we designed some approximation of final productive fitness that can at least be computed a posteriori for an already run evolution, giving some insight into the algorithm's workings. In this paper, we now provide the full design for a proof of final productive fitness's optimality, although there are still many risky new tools involved.

3.2 Proof Design

We hope that the notion of final productive fitness is intuitive enough so that it seems plausible how $\phi^\dagger$ might be the ideal fitness function ϕ^* for any evolutionary algorithm. However, evolutionary processes are highly stochastic entities and little framework exists to reason about their performance. We now first provide such a framework, although we resort to making some strong assumptions along the way.

Assumption 1 (Random Agnosticism)**.** *Random effects residing in selection functions and evolutionary operators exert the same general effects on the evolutionary function E regardless of the used fitness function.*

The main intention behind Assumption 1 is, of course, to exclude any concept of randomness from the proof design. We effectively assume that the distribution of outcomes (i.e., selected individuals or generated children) depending on random inputs does not depend on the fitness function. At first, this is a certainly outlandish and strong assumption, especially as it allows us to deduce quite strong properties. We just give a few reasons why it might be viable:

- Wherever random effects are used, they are usually designed to break clear fitness borders (for example when using a fuzzy cutoff vs. a discrete cutoff). In these cases, random effects overpower the effect of the fitness function so that (in the extreme case, consider random selection) the used fitness function has little impact on the outcome. If we flip the perspective around, different fitness functions then also have little difference for the outcome.
- Within the evolutionary function E, typically lots of random effects come together. Even if some of their distributions are altered by using a different fitness function, as long as they are not altered towards a specific result, the effect may still cancel out on the larger scale. Basically, we expect the outcome distribution of the whole evolutionary function E to approach the normal distribution irregardless of (un-systematic) mix-ups.

– From practical perspective, the shape of outcome distributions is rarely considered directly when constructing an evolutionary algorithm. That should usually indicate that not much effect can be observed there in most cases.

Eventually, all these reasons are flawed, of course. Otherwise, we would not have kept Assumption 1. Still, we feel that proofs built on Assumption 1 might have practical relevance for the time being.

It would be natural to follow up the effective elimination of randomness (coming from Assumption 1) by replacing all possibly random outcomes with the expected value of the distribution and treating all function as non-stochastic. However, the expected value is still computed from the distribution, so this would not make things much easier. Instead, we opt of the ideal outcome, which can be derived much easier, but might shift effects drastically: A recombination operator that performs so bad on average that it brings down the whole evolutionary process might now look like it gives rise to a very effective evolutionary process just because it has a very small chance of getting a really good result.

Assumption 2 (Based Optimism)**.** *For an evolutionary function E with a limited amount of possible outcomes, the best possible outcome is representative for its expected average result.*

We recognize that "limited" is not fully defined here. We suggest that future work looks into enumerability or local boundedness. For practical purposes, however, it is clear which of the classic operators are affected: Random initialization and migration can generate individuals across the whole search space. If we minimize over their possible outcomes, the whole algorithm reaches the global optimum in a single step. Mutation and recombination (with any kind of selection) on the other side are limited operators: Given certain individuals as input parameters, they will only navigate a limited range of options related to those individuals. Again the main argument for the plausibility of Assumption 2 is that the results usually approach normal distribution anyway and there is no real reason why they should act any differently given exactly the two fitness functions we are about to compare. However, given that we completely alter the rules of evolutionary algorithms with this one, it is definitely a bold assumption. Further note how we interpret the qualification "best" in Assumption 2: For a given evolutionary function E, its notion of "best" corresponds to its fitness function. So if we choose the best of two evolutionary processes with different fitness functions, we might actually choose two different points in the outcome distribution (depending on the fitness), which again is an immensely powerful tool based on a big assumption.

What Assumption 2 then provides is means to simplify Definition 10: Productive fitness is defined as the average fitness of all descendants. We can now use the *best* fitness of the descendants to compute the fitness measurement which

we will call *optimistic productive fitness*.[5] Note that implementing these assumptions has a great effect on the behavior of the evolutionary process. However, we do not claim that they leave the evolutionary process intact, we just claim that a clearly better fitness function remains the better fitness function even in the altered setting.

The tools provided by Assumptions 1 and 2 are rather novel and very powerful, so we are aware that any results based on them should be taken with a great amount of caution. However, in order to present these tools at work, we can use them to provide a proof that final productive fitness $\phi^\dagger$ is one answer for Problem 1.

Proof 1 (Problem 1)**.** *Let $\mathcal{E}^\dagger = \langle \mathcal{X}, E^\dagger, t, (X_i^\dagger)_{i<g} \rangle$ be an evolutionary process using optimistic final productive fitness $\phi^\dagger$. Let $\mathcal{E}^* = \langle \mathcal{X}, E^*, t, (X_i^*)_{i<g} \rangle$ be an evolutionary process using a different (possibly more ideal) fitness ϕ^*. According to the transformation discussed in Sect. 2.1, let both $\mathcal{E}^\dagger$ and $\mathcal{E}^*$ be elitist. Let $X_0^\dagger = X_0^*$. We assume that $t(||\mathcal{E}^\dagger||_t) > t(||\mathcal{E}^*||_t)$, i.e., because of elitism*

$$\min_{x \in X_g^\dagger} t(x) > \min_{x \in X_g^*} t(x). \tag{8}$$

From Eq. 8 it follows that there exists an individual $x \in X_g^*$ so that $x \notin X_g^\dagger$ and $t(x) < \min_{y \in X_g^\dagger} t(y)$. The better individual x could not have been introduced into the population of $\mathcal{E}^*$ by migration (or random initialization for that matter) as we could use Assumption 1 to just introduce x into $\mathcal{E}^\dagger$ then.

Then x needs to stem from an individual x' that is an ancestor of x, i.e., $x \in D_{x'}$, so that x' was selected for survival in $\mathcal{E}^*$ and not in $\mathcal{E}^\dagger$, which implies that $\phi^\dagger(x') > \phi^*(x')$. However, since x is a possible descendant for x', the computation of $\phi^\dagger(x')$ should have taken $t(x)$ into account,[6] meaning that x' should have survived in $\mathcal{E}^\dagger$ because of elitism after all, which contradicts the previous assumption (Eq. 8). □

3.3 Example

We now illustrate the notion of productive fitness for our running example. For each of the individuals generated in Sect. 2.3 we computed an a posteriori approximation for final productive fitness: Basically, we took the descendants that *have in fact been generated* during evolution as a representative subset of

[5] Note that the best possible choice for the average fitness of descendants is the minimum of the possible descendants' fitness values. When we are allowed to adjust the random choice for the best possible outcomes, worse-than-optimal children will not be born. This changes the game: Our ideal choice from the vast space of random possibilities now yields at most one (i.e. the best possible) descendant per individual per generation.

[6] Note that $t(x)$ cannot be compensated by other descendants of x' with possibly bad objective fitness since we assumed optimistic final productive fitness following Assumption 2.

the descendants that *could have been generated*. This allows us to compute a value for $\phi^\dagger$ for an already finished evolution.[7]

Note that the notion of final productive fitness is most powerful in the beginning of the evolutionary process, when it carries information about the whole evolution to come. Figures 3a and 3b provide a clear situation how final productive fitness is a better fitness function than the target function:

- The final productive fitness landscape has fewer valleys as its local optima correspond to individuals that remained in the final generation of some evolutionary process. This makes the landscape less deceptive and individuals are more clearly guided towards at least somewhat good results.
- The basins around the optima are wider, again making the local optimization towards the final result more clear.
- The differences between the global optimum and other local optima are more pronounced, giving an edge to the global optimum.

As discussed, if we could use final productive fitness during evolution, it would allow for better results. However, approximations of various quality may exist for specific problems or problem instances [8].

In Figs. 3c and 3d we can see how the final productive fitness landscapes deteriorates with the progressing evolution. As we can see from the red dots, evolution has focused on certain areas of the solution landscape, leaving wide areas without a meaningful final productive fitness to be computed. This effect is even more prominent in Figs. 3e and 3f, where individuals that are still randomly generated in certain areas die out rather quickly, leaving them with a productive fitness of 1. Note that productive fitness cannot meaningfully be computed for the last generation so we deliberately choose to show generation 49 last here.

Note how Fig. 3 also illustrates the usage of different evolutionary operators: For the Schwefel function, many individuals have a good final productive fitness when the evolution starts. That means they have direct descendants who manage to achieve nearly optimal target values. By contrast, H1 shows no individuals with good productive fitness in the beginning, meaning that the final results were mostly discovered via the migration operator *mig* as that is not traced by productive fitness.[8]

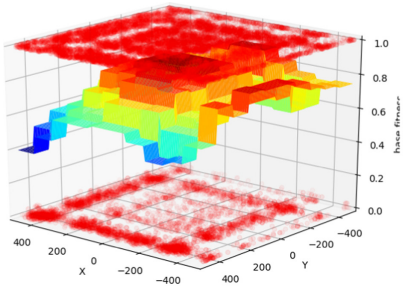
4 Related Work

We first introduced the gist of the formal framework for evolutionary processes as well as the notion of productive fitness in [8]. In this paper, we provide and discuss the full, substantially extended framework and introduce the assumptions and tools a proof design for productive fitness's validity can be built with.

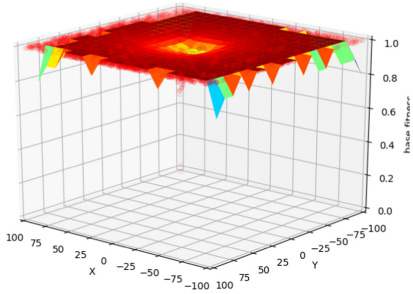
Theoretical work on evolutionary algorithms has been traditionally focused on the complexity of the search process (on rather simple search problems) or

[7] For details on how this is done, please refer to [8].

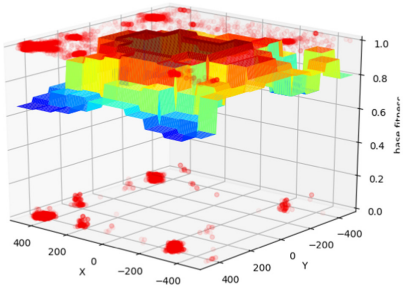
[8] Migrants are generated randomly and are thus not ascribed to be any individual's descendant. How to include migrants in productive fitness is left for future work.



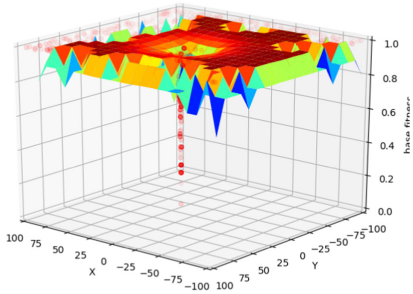
(a) Schwefel, generation 1



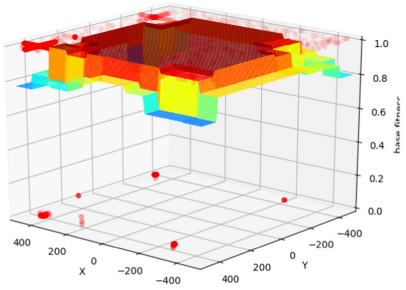
(b) H1, generation 1



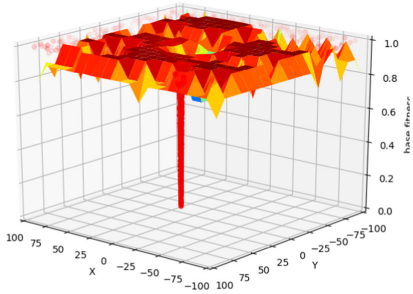
(c) Schwefel, generation 10



(d) H1, generation 10



(e) Schwefel, generation 49



(f) H1, generation 49

Fig. 3. Individuals from 500 independent runs of the evolutionary processes plotted with their a posteriori approximated final productive fitness. The surface represents the same data set as the scatter points, where each tile has the Z value equal to the average Z value of all the points within it.

the performance of various types and variants of algorithms in general. We point to [4–6] for a few selective examples without any attempt at giving a full overview over this old and comprehensive field of research. By contrast, we fully work out the difference between a target function that is given from the outside world and a fitness function that (potentially) emerges implicitly throughout the process of evolution. As this concept in itself is rather novel, the constructs supporting it have been freshly developed as well (and are still in their infancy).

It should be pointed out that there probably exists a connection from the assumptions and approximations we make to a complexity-based analysis of evolution, as these tools allow us to rule out exponentially many options and thus bring the respective computation to a feasible level.[9]

Various meta-measurements of fitness in evolutionary algorithms have been designed. We would like to point out *effective fitness* [13], which describes the fitness threshold under which individuals can manage to increase their dominance in the population. This usually is a harsher border than *reproductive fitness* [11], which is the probability of an individual to successfully produce offspring. Both follow a similar line of thought of measuring what fitness an individual needs to have for certain effects to occur, but none suggest using the meta-measurement as a fitness value itself.

5 Conclusion

We have introduced and discussed a formal description of evolutionary processes that summarizes various kinds of evolutionary algorithms and other optimization techniques. Based on that framework, we defined the notion of productive fitness as it is defined in [8], where an argument was sketched why it might be the ideal fitness function. In this paper, we introduced the tools necessary to implement the proof, discussed their validity and thus gave the full proof design. We argue that while the approach is somewhat bold, the assumptions made could be useful for similar arguments about evolutionary processes and hope the perspective on fitness functions given here will open up new ways to reason about highly dynamic and uncertain processes, especially evolution.

We pointed out future work where we encountered open questions. We consider the connection suggested to traditional runtime analysis of evolutionary algorithms and subsequently to the No Free Lunch theorem [10] and how it related to the cases of having and using as well as finding and approximating the ideal fitness function to be especially promising. In addition, we suggest that it might be of particular relevance to also expand the scope of the framework beyond evolutionary algorithms; even the proof design might be adapted to not only work for fitness used by evolutionary operators but for example to deliver the ideal reward function for reinforcement learning [3,9].

[9] As no computational limit on biological evolution, e.g., has been recognized it could be an interesting endeavor to use the framework presented in this paper to translate arguments from runtime analysis of evolutionary algorithms back to a more general concept of evolution.

References

1. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: a survey and categorisation. Inf. Fusion **6**(1), 5–20 (2005)
2. DEAP Project: Benchmarks (2020). https://deap.readthedocs.io/en/master/api/benchmarks.html. Accessed June 1 2020
3. Dewey, D.: Reinforcement learning and the reward engineering principle. In: 2014 AAAI Spring Symposium Series (2014)
4. Doerr, B., Happ, E., Klein, C.: Crossover can provably be useful in evolutionary computation. Theoret. Comput. Sci. **425**, 17–33 (2012)
5. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoret. Comput. Sci. **276**(1–2), 51–81 (2002)
6. Friedrich, T., Oliveto, P.S., Sudholt, D., Witt, C.: Analysis of diversity-preserving mechanisms for global exploration. Evol. Comp. **17**(4), 455–476 (2009)
7. Gabor, T., Belzner, L., Linnhoff-Popien, C.: Inheritance-based diversity measures for explicit convergence control in evolutionary algorithms. In: Genetic and Evolutionary Computation Conference, pp. 841–848 (2018)
8. Gabor, T., Phan, T., Linnhoff-Popien, C.: Productive fitness in diversity-aware evolutionary algorithms (2021). (submitted)
9. Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S.J., Dragan, A.: Inverse reward design. In: Advances in Neural Information Processing Systems, pp. 6765–6774 (2017)
10. Ho, Y.C., Pepyne, D.L.: Simple explanation of the no free lunch theorem of optimization. In: 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228), vol. 5, pp. 4409–4414. IEEE (2001)
11. Hu, T., Banzhaf, W.: Evolvability and speed of evolutionary algorithms in light of recent developments in biology. J. Artif. Evol. Appl. **2010**, 1 (2010)
12. Rainville, D., Fortin, F.A., Gardner, M.A., Parizeau, M., Gagné, C., et al.: Deap: a python framework for evolutionary algorithms. In: Conference Companion on Genetic and Evolutionary Computation, pp. 85–92. ACM (2012)
13. Stephens, C.R.: "Effective" fitness landscapes for evolutionary systems. In: 1999 Congress on Evolutionary Computation (CEC 1999), vol. 1, pp. 703–714. IEEE (1999)
14. Van Soest, A.K., Casius, L.R.: The merits of a parallel genetic algorithm in solving hard optimization problems. J. Biomech. Eng. **125**(1), 141–146 (2003)
15. Wineberg, M., Oppacher, F.: The underlying similarity of diversity measures used in evolutionary computation. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1493–1504. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45110-2_21