First Steps Towards a Runtime Analysis of Neuroevolution

Fischer, Paul; Larsen, Emil Lundt; Witt, Carsten

Published in:
Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms

Link to article, DOI:
10.1145/3594805.3607125

Publication date:
2023

Document Version
Peer reviewed version

First Steps Towards a Runtime Analysis of Neuroevolution

Paul Fischer
DTU Compute
Technical University of Denmark
Kongens Lyngby, Denmark

Emil Lundt Larsen*
DTU Compute
Technical University of Denmark
Kongens Lyngby, Denmark

Carsten Witt
DTU Compute
Technical University of Denmark
Kongens Lyngby, Denmark

ABSTRACT

We consider a simple setting in neuroevolution where an evolutionary algorithm optimizes the weights and activation functions of a simple artificial neural network. We then define simple example functions to be learned by the network and conduct rigorous runtime analyses for networks with a single neuron and for a more advanced structure with several neurons and two layers. Our results show that the proposed algorithm is generally efficient on two example problems designed for one neuron and efficient with at least constant probability on the example problem for a two-layer network. In particular, the so-called harmonic mutation operator choosing steps of size j with probability proportional to $1/j$ turns out as a good choice for the underlying search space. However, for the case of one neuron, we also identify situations with hard-to-overcome local optima. Experimental investigations of our neuroevolutionary algorithm and a state-of-the-art CMA-ES support the theoretical findings.

CCS CONCEPTS

• **Theory of computation → Theory of randomized search heuristics**.

KEYWORDS

neuroevolution, theory, runtime analysis

ACM Reference Format:
Paul Fischer, Emil Lundt Larsen, and Carsten Witt. 2023. First Steps Towards a Runtime Analysis of Neuroevolution. In *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA '23), August 30–September 1, 2023, Potsdam, Germany*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3594805.3607125

1 INTRODUCTION

The term *neuroevolution* describes the generation and iterative improvement of artificial neural networks (ANNs) by means of evolutionary computation. Neuroevolution is applied in scenarios where classical techniques like backpropagation for the optimization of network weights are not available or not satisfactory. Moreover, neuroevolution allows for the automated optimization of network topology, i. e., the number of neurons, layers, and their interconnecting structure in the network, which may be a time-consuming manual task otherwise. Neuroevolution dates back to the late 1980s [19, 30] but has become increasingly popular in recent years along with several breakthroughs in the field of artificial intelligence, most notably, so-called deep neural networks. For a much broader overview, we refer the reader to the recent surveys [11, 23, 26] on neuroevolution and the strongly related field of evolutionary neural architecture search (which focuses on the optimization of neural network topology rather than the weights belonging to neurons).

Evolutionary algorithms (EAs) are nature-inspired, heuristic optimization techniques applied in virtually all engineering disciplines. There is huge empirical knowledge on their application, but also an increasingly solid theory that guides the design and application of EAs. In particular, theoretical runtime analysis has become an established branch in the theory of evolutionary computation that enables such results; see the works [6, 7, 16, 20] for an overview of classical and recent results. The first results from the early 1990s considered extremely simplified EAs like the famous (1+1) EA on the simple OneMax benchmark function. Such initial analyses have paved the way toward the analysis of more realistic, population-based EAs on advanced benchmarks and classical combinatorial optimization problems. Moreover, runtime analysis has led to theoretically grounded advice on parameter choices in EAs and the development of new, high-performing variants of EAs.

Despite these advancements in the theory of EAs and the huge empirical success of neuroevolution, we are not aware of any theoretical runtime analyses of neuroevolution. The aim of this paper is to be a starting point for such an analysis. We will suggest a simple optimization environment in neuroevolution inspired by the simple evolutionary algorithms mentioned above (e. g., the (1+1) EA) that evolves the parameters of neurons and suggest optimization problems dealing with the classification of certain points on the unit hypersphere. By giving the two halves of the hypersphere opposite labels and discretizing the setting, we arrive at a simple problem that could take the role of a kind of *OneMax of Neuroevolution*.

The first environment we investigate is restricted to the simplest possible network of one neuron with binary activation function only, where the evolutionary algorithm evolves the bias of the activation function and the weights of the inputs in a representation as a polar angle. We find that the algorithm is generally efficient on problems in two dimensions where an arc of constant size of the unit circle has to be classified positively. Afterwards, we will extend the environment to an arbitrary number of neurons and two layers and present modifications of the classification problems on the unit hypersphere that require more than one neuron to be solved exactly. Moreover, we will present problems with local optima which are hard to overcome. While analyzing the runtimes, we compare different mutation operators, more precisely a local

*The author now works at Abzu ApS, Copenhagen.

FOGA '23, August 30-September 1, 2023, Potsdam, Germany

ACM ISBN 979-8-4007-0202-0/23/08…$15.00
https://doi.org/10.1145/3594805.3607125

one and the harmonic mutation operator introduced in [2], and prove exponentially (in the desired resolution of the discretized search space) smaller bounds for the harmonic mutation in several cases. Our proposed classification problems may serve as examples of typical optimization (sub)scenarios in neuroevolution and as a starting point for the runtime analysis of more advanced scenarios.

This paper is structured as follows. In Section 2, we introduce the formal background on neural networks and their parametrization as well as the proposed neuroevolutionary algorithm and benchmark problems. Section 3 proves the concrete runtime results. Section 4 is devoted to experimental supplements, before we finish with some conclusions. Due to space restrictions, details of the experimental data were omitted. They can be found in a technical report [10].

2 PRELIMINARIES

In this section, we present the foundations of ANNs for classification problems that are relevant for our study, define a simple evolutionary algorithm for neuroevolution, and example problems that will be used in our theoretical and empirical studies. For a broader introduction to ANNs and machine learning, see, e. g., [25].

2.1 Artificial Neuron

We are considering artificial neurons with D inputs and a binary threshold activation function, i. e., the output is 0 or 1. Such a neuron is sometimes called *perceptron*. It has $D + 1$ parameters, the input weights $w_1, \dots, w_D$ and the thresholding value t. Let $x = (x_1, \dots, x_D) \in \mathbb{R}^D$ be the inputs of the neuron. The neuron outputs 1 if

$$w_1 x_1 + w_2 x_2 + \cdots + w_D x_D \geq t \tag{1}$$

and 0 otherwise. See Figure 1 for an illustration. The equation can be normalized such that $t = 1$.

A single neuron can be considered as a minimal, one-layer neural "network".

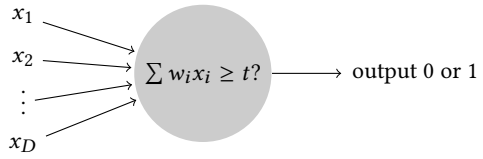

Figure 1: An artificial neuron

In a geometric interpretation, Equation (1) would mean that the point $(x_1, \dots, x_D) \in \mathbb{R}^D$ is classified 1 if it is *above or on* the hyperplane with normal vector $(w_1, \dots, w_D)$ and bias t, assuming an appropriate orientation of the coordinate system. If $D = 2$, this hyperplane becomes the line given by equation

$$y = \frac{t}{w_2} - \frac{w_1}{w_2} x \, .$$

Replacing $\geq$ by $\leq$ in Equation (1) will classify the points *below or on* the line as 1. Although the interpretation of classifying points as 0 or 1 does not depend on the dimension D, we will in this work mostly study the case $D = 2$ for simplicity.

Networks of artificial neurons (not restricted to perceptrons), called ANNs, are used to approximate (or even solve exactly) classification problems in high-dimensional spaces. Formally, a binary classification problem is a set of points $S \subseteq \mathbb{R}^D$ and the true classification of a point $x \in \mathbb{R}^D$ is simply the membership function. For example, points could be from the space of representations of pictures, S could be the set of pictures containing a cat, and the classification problem would be to determine whether a given point $x \in S' \subseteq \mathbb{R}^D$ is a picture containing a cat. Here S' is an appropriate subset of possible queries. We call the points in S positive (the others negative) and would like to know whether a given point $x \in S'$ is positive or not. In this paper, ANNs with a binary output/activation function are used to predict whether x is positive (output 1) or negative (output 0). The aim is to find a topology and parametrization of the ANN that gives the correct prediction on as many points from S' as possible. Usually, the degree to which it is achieved is measured by the so-called classification error. A classical iterative technique to set the weights of ANNs to minimize the classification error is called backpropagation, and the underlying iterative process is called "training" of the ANN. However, backpropagation does not straightforwardly work on non-differentiable output functions like the step function considered here.

In neuro-evolutionary algorithms, here again illustrated by the perceptron with two input dimensions, the search dynamics to minimize the classification error usually happens by modifying the parameters w_1, w_2 and t of the neuron, which results in moving the decision line associated with the neuron. In this paper, we will be dealing with classification problems whose point sets are subsets of the unit hypersphere. This motivates us to use a different representation of the decision line corresponding to Hesse normal form, i. e., by specifying angle φ of the unit normal vector for the hyperplane (in two dimensions, a line) and its bias b (which then is its distance from the origin measured in the opposing direction of that of the normal vector). Then the line and the halfspace into which the normal vector points are classified as 1. As an additional advantage, the parameter set consisting of angle and bias consists only of two values compared to three values w_1, w_2, t in the original representation. The representations (w_1, w_2, t) and (φ, b) are easily convertible into each other.

2.2 ANNs with Two Layers

After having considered the single perceptron, we will extend our analysis to ANNs with a larger number of neurons and layers. Here we study a simple structure of a so-called feed-forward network with two layers, a hidden one and an output layer.

The hidden layer comprises $N > 1$ neurons, still with binary output function, which are all connected to the inputs $x_1, \dots, x_D$. The output layer is assumed to compute the Boolean OR of the outputs of the hidden layer. This structure has been chosen as we will be dealing mostly with problems that can be described as the disjoint union of half-spaces of $\mathbb{R}^D$. See Figure 2 for an illustration.

While the hard-wired OR function in the output layer may seem rather problem-specific, it is actually not difficult to set weights $w_1, \dots, w_N$ and a threshold t that makes the output neuron compute a Boolean OR of the binary outputs $o_1, \dots, o_N$ from the hidden layer (e. g., choose $t > 0$ and all weights at least t). In experiments (see

Section 4), we placed a neuron in the output layer which mutated in the same way as the others. This neuron almost always settled to compute a Boolean function of the outputs of the hidden layer such that the overall fitness is optimal. The function was not necessarily an OR, but it would sometimes swap the classifications of the two sides of a line to make them correct. A full theoretical analysis, where also the parameters of the output layer are evolved, is subject for future research.

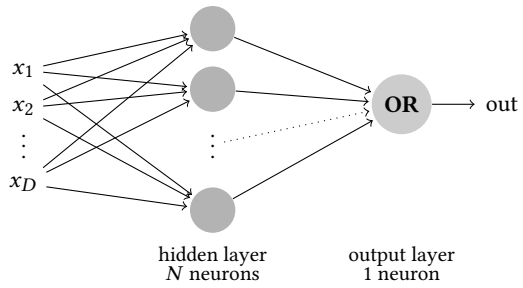

Figure 2: Structure of ANNs with two layers

We remark that from a theoretical perspective, even very small NNs are challenging and interesting to study. For example, training networks with binary activation functions is NP-hard already for 3 neurons [1]. In the case of the continuous sigmoidal activation function, even the training of a single neuron is NP-hard [27].

2.3 Algorithms

Classical frameworks for neuroevolution like NEAT [24] evolve both the topology and the weights (and, if applicable, the biases) of the network. This typically leads to a mixed discrete-continuous search space, which may be hard to analyze. As argued above, we assume a fixed topology for the network in this first runtime analysis of neuroevolution but let the evolutionary algorithm evolve the weights and biases of the neurons.

Setting the weights and biases of neurons in ANNs is usually a continuous optimization problem. However, rigorous runtime analysis is much less developed for continuous optimization problems than for discrete problems. With the idea of defining a *OneMax of Neuroevolution* in mind, i. e., the transfer of a discrete optimization problem, we would like to find a discretization of the setting that still represents key aspects of the original, continuous setting. Here we do not find a characterization as pseudo-boolean optimization problem $f\colon \{0,1\}^n \to \mathbb{R}$ appropriate. One could imagine n parallel neurons and for each of these a binary parameter that corresponds to activating/not activating the neuron and count the number of activated neurons; however, this would be essentially the same as OneMax. The search space $\{0,1\}^n$ could also be used to model non-binary parameters by dividing it into blocks, e. g., of $\lceil \log r \rceil$ bits to represent an integer in $\{0,\dots,r-1\}$. Again, we do not follow this choice since small changes like flipping a bit on the genotype (i. e., representation) level might lead to large changes in the phenotype. Alternative mappings like grey codes [22], that avoid these excessive changes, seem hard to analyze.

In this paper, we will use the state space $\{0,\dots,r\}^m$, where m is proportional to the number of neurons and r is the resolution of the discretization of a continuous parameter from a compact domain; e. g., if the parameter lives on $[0,1]$, then the discretization allows for the values $[0, 1/r, 2/r, \dots, 1]$. Search spaces of the type $\{0,\dots,r\}^m$, where usually r is small but m is growing, have been considered before in runtime analyses, see, e. g., [3, 8, 12, 18].

After having defined the search space, we must agree on the search operators used in the (neuro)evolutionary algorithm. A classical search operator in neuroevolution is mutation, which may add a Gaussian random variable to a weight, choose a weight uniformly from a compact interval etc. (both of which are valid choices in NEAT). We suggest mutation operators that change the network parameter (e. g., a weight or bias of a neuron) by adding ℓ/r to the parameter or subtracting ℓ/r with ℓ drawn from some distribution discussed below. Since we assume in advance that the parameters lives on the compact interval like $[-a,a]$ for some constant a (since bounding the domain is a typical assumption in the optimization of network parameters), we continue the interval cyclically, i. e., formally the result is taken modulo r (or modulo $r+1$). This makes sense especially for the angle of the hyperplane/line belonging to a the neuron because of its periodic structure. See more details below.

In principle, the discretized search space $\{-a, -a+2a/r, a+4a/r, \dots, a-2a/r, a\}$ arising from dividing the intervals $[-a,a]$ in r equally spaced segments allows the algorithm to find solutions up to an error of $2a/r$ in the search space – not necessarily in the objective space. To obtain arbitrary precision, real-valued EAs would typically introduce a form of self-adaptation (like a 1/5-rule, cf. [15]), which we ignore in this first study of the runtime of neuroevolution algorithms. As a possible alternative to our discretization, one could also try to work with heavy-tailed mutation operators for compact continuous search spaces, but these are not easy to analyze from a theoretical runtime perspective, so we only consider them in the experimental part (Section 4). Moreover, we do not use more advanced search operators like crossover here.

Finally, we define the *fitness function* f used in the following. Informally this can be understood as the fraction of correctly classified points on the unit hypersphere. Formally, we consider points $S_D := \{x \in \mathbb{R}^D \mid \|x\|_2 = 1\}$ as inputs to the ANN and and a binary classification problem with labels in $\{0,1\}$ on these points. We then compute

$$\frac{\mathrm{vol}(((C_D \cap L_D) \cup (\overline{C_D} \cap \overline{L_D})) \cap S_D)}{\mathrm{vol}(S_D)},$$

where C_D is the union of half-spaces above (or on) the hyperplanes spanned by the N neurons, $L_D \subset S_D$ the set of points classified 1, $\overline{A} = \mathbb{R}^D \setminus A$, and $\mathrm{vol}(\cdot)$ denotes the (hyper)volume. As a side note, this fitness function can also be understood as the quality of the network with respect to a training set uniformly distributed on S_D. Note that this training set would be of infinite size; in future work one might want to consider a finite-size sample of the set (e. g., uniformly), which we think would lead to similar results like in the present paper but would involve more corner cases and a more involved analysis.

Based on our discussion, we suggest the so-called *(1+1) Neuroevolution Algorithm* ((1+1) NA), given as Algorithm 1. It maintains N neurons with two inputs each as explained in Sections 2.1–2.2, for which biases and angles of normal vectors in polar coordinates are evolved. Recall that the representation with angles

Algorithm 1: (1+1) NA

$t \leftarrow 0$; select x_0 uniformly at random from $\{0,\dots,r\}^{2N}$.
while *termination criterion not met* **do**
  Let $y = (\varphi_1, b_1, \dots, \varphi_N, b_N) \leftarrow x_t$;
  For all $i \in \{1,\dots,N\}$, mutate φ_i and b_i with probability $\frac{1}{2N}$, independently of each other and other indices;
  Mutation chooses $\sigma \in \{-1,1\}$ u. a. r. and $\ell \sim \mathrm{Harm}(r)$ and adds $\sigma\ell$ to the selected component; the result is taken modulo r for angle and modulo $r+1$ for bias;
  For $i \in \{1,\dots,N\}$, set polar angle $2\pi\varphi_i/r$ and bias $2b_i/r - 1$ for neuron i to evaluate $f(y)$;
  if $f(y) \geq f(x_t)$ **then** $x_{t+1} \leftarrow y$;
  else $x_{t+1} = x_t$;
  $t \leftarrow t+1$;

Notation	Interpretation
N	number of neurons
D	input dimension for the ANN (usually $D = 2$)
r	resolution of angle and bias
$[-1,1]$	domain of neurons' bias
$[0,2\pi)$	domain of neurons' angle
$\{0,\dots,r\}^{2N}$	search space of algorithm
$(\varphi_1, b_1, \varphi_2, b_2, \dots, \varphi_N, b_N)$	search point: list of N angle/bias-pairs (φ_i, b_i), $i \in \{1,\dots,N\}$, for the N neurons; $\varphi_i, b_i \in \{0,\dots,r\}$
$f\colon \{0,\dots,r\}^{2N} \to [0,1]$	fitness function, returning the fraction of correctly classified points on the unit hypersphere

Table 1: Overview of notation in (1+1) NA (and its analyses)

is preferred over Cartesian coordinates because of the spherical structure of our forthcoming benchmark problems. We also recall that the bias of a neuron coincides with the distance of its line from the origin, assuming a unit length for the normal vector. The algorithm has a global search operator using the harmonic distribution $Harm(r)$ on $\{1,\dots,r\}$ for the magnitude of change ℓ; more precisely, $\mathrm{Prob}(\ell = i) = 1/(iH_r)$ for $i \in \{1,\dots,r\}$, where $H_r = \sum_{i=1}^{r} 1/i$. This operator was used before in [2, 3, 13] for similar search spaces. Table 1 summarizes the parameters and settings of the (1+1) NA.

We note that the independent choices for the mutated components allow void steps where nothing is mutated. We ignore this here for the sake of simplicity; from an algorithm-engineering perspective, one would simply redraw the mutation if it does not change anything [4, 21].

Variants of (1+1) NA. We shall also define and analyze the following natural simplifications of the above (1+1) NA:

- The *local (1+1) NA* only changes its components by ±1, i. e., $\ell = 1$ is fixed instead of being drawn from a Harmonic distribution. This operator is called *unit mutation* in [3].
- The *(1+1) NA without bias* fixes $b_i = 0$ for $\{1,\dots,N\}$ and maintains search points $(\varphi_1,\dots,\varphi_N)$ consisting of N angles only, which are subject to the same type of mutation as the original (1+1) NA.

Further variants of the algorithm may be investigated in the future, e. g., different choices for the mutation and other probabilistic elements. In particular, the self-adjusting mutation from [3] is a rather relevant alternative to the simple mutations considered here.

Optimization time. A common convention in runtime analysis in discrete search spaces is to define the optimization time (synonymously, *runtime*) as the number of fitness function evaluations until a solution having optimal fitness value has been sampled. We adapt this to our discretized search space by still counting fitness evaluations, but saying that the function has been optimized if the current search point $x = (\varphi_1, b_1, \dots, \varphi_N, b_N)$ deviates by an absolute value of less than 1 in the representation of angles and biases, i. e.,

$$\max_{i=1}^{N}\{|\varphi_i - \varphi_i^*|, |b_i - b_i^*|\} < 1$$

for an optimal (fractional) solution $(\varphi_1^*, b_1^*, \dots, \varphi_N^*, b_N^*)$, where the absolute values are with wrap-around in the respective intervals. This corresponds to an $O(1/r)$-error in terms of the actual value of bias or angle. Typically, the expected value of the stochastic optimization time is bounded. Since each element of the search space has a probability of at least $(1/(rH_r))^{2N}$ of being hit by mutation, we obtain the following bound, similar to the worst-case bound for the (1+1) EA on pseudo-boolean problems [9].

Lemma 2.1. *The expected optimization time of the (1+1) NA on an arbitrary problem is at most* $O((r\log r + r)^{2N})$.

Even if $N = c$ for a constant c, we do not consider the general runtime bound $O((r\log r + r)^{2c})$ as particularly efficient. In fact, for simple problems bounds being polylogarithmic in r like $(\log r)^{O(1)}$ can be obtained, as shown and discussed below.

Finally, we remark that the set of optimal solutions for a given optimization problem may depend on the number of allowed neurons and whether bias is allowed or not. We will consider examples where with only one neuron, not all points of the underlying classification problem can be classified correctly, while this is possible with at least two neurons.

2.4 Problems

In this section, we define several benchmark problems that shall illustrate how the (1+1) NA makes progress towards a correct classification with one or several neurons. Also, the section serves to point out typical situations in the optimization that can make the algorithm stuck in a local optimum. As argued above, we identify problems with the points in $S_D \subseteq \mathbb{R}^D$ classified positively, i. e., as 1. All problems are defined for arbitrary $D \geq 2$; however, for the sake of simplicity most analyses will be restricted to $D = 2$.

The following problem can be thought of as a kind of OneMax for the (1+1) NA without bias. However, there are limits to this analogy since the fitness landscape for the (1+1) NA (see Section 3) is more uniform than the for the classical (1+1) EA on OneMax.

Definition 2.2. The problem Half consists of all points with non-negative x_D-dimension on the unit hypersphere, i. e.,

$$\begin{aligned}\textsc{Half} &= \{x \in \mathbb{R}^D \mid \|x\|_2 = 1 \text{ and } x_D \geq 0\}\\ &= \{x \in \mathbb{R}^D \mid \|x\|_2 = 1 \text{ and } \psi_{D-1} \in [0, \pi]\},\end{aligned}$$

where ψ_{D-1} is the polar spherical angle between x and the unit hypersphere on the first $D-1$ dimensions.

Obviously, setting the angle of a single neuron to $\pi/2$ and its bias to 0 is optimal here. See Figure 3 for sketch of Half and the following two problems with $D = 2$.

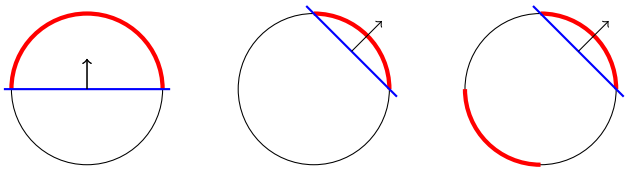

Figure 3: Illustration of Half, Quarter and TwoQuarters (from left to right). The thick red areas constitute the target points that should be classified positively. Hyperplanes at optimal positions are shown in blue. The arrow points to the positively classified halfspace. The fitnesses are 1, 1, and $3/4$, respectively.

Similarly, we define Quarter. It can still be solved optimally (according to our definition above) with the special case of 1 neuron if the bias is allowed to vary. The global optimum is at angle $\pi/4$ and bias $\sqrt{2}/2$.

Definition 2.3. The problem Quarter consists of all points with non-negative x_{D-1} and x_D-dimension on the unit hypersphere, i. e.,

$$\begin{aligned}\textsc{Quarter} &= \{x \in \mathbb{R}^D \mid \|x\|_2 = 1 \text{ and } (x_{D-1}, x_D) \geq (0, 0)\}\\ &= \{x \in \mathbb{R}^D \mid \|x\|_2 = 1 \text{ and } \psi_{D-1} \in [0, \pi/2]\},\end{aligned}$$

The next problem requires at least two linear classifiers, i. e., two neurons, and a neuron (possibly hard-wired) joining the results of those two to be solved exactly. With only one neuron, at least $1/4$ of the circle will be classified incorrectly.

Definition 2.4. The problem TwoQuarters consists of all points with either both non-negative or both positive x_{D-1} and x_D-dimension on the unit hypersphere, i. e.,

$$\begin{aligned}\textsc{TwoQuarters} &= \{x \in \mathbb{R}^D \mid \|x\|_2 = 1 \text{ and } x_{D-1}x_D \geq 0\}\\ &= \{x \in \mathbb{R}^D \mid \|x\|_2 = 1 \text{ and } \psi_{D-1} \in [0, \pi/2] \cup [\pi, 3\pi/2]\},\end{aligned}$$

Finally, we define the problem that has hard-to-overcome local optima in the fitness landscape given by the local (1+1) NA. See more details in the following section. It is most convenient to define the problem by means of polar coordinates.

Definition 2.5. The problem LocalOpt consists of all points on the unit hypersphere with polar angle ψ_{D-1} between 0 and 60, 120 and 180, or 240 and 330 degrees, i. e.,

$$\begin{aligned}\textsc{LocalOpt} = \{&x \in \mathbb{R}^D \mid \|x\|_2 = 1\\ &\text{and } \psi_{D-1} \in [0, \pi/3] \cup [2\pi/3, \pi] \cup [4\pi/3, 11\pi/6]\}\end{aligned}$$

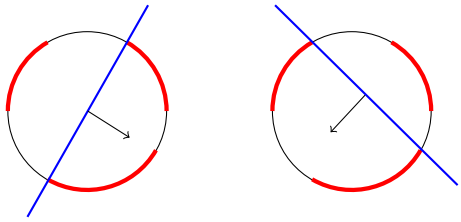

Figure 4: Examples for LocalOpt. The colors are as in figure 3. One optimal solution is shown at the left having a fitness of $3/4$ (there are two more). The solution at right is locally optimal with a fitness of $2/3$.

Figure 4 shows examples of a globally optimal solution and one in a local optimum, assuming $D = 2$.

To conclude this section and to prepare the forthcoming analyses, we define a problem class that all above problems belong to.

Definition 2.6. A classification problem $S \subseteq \mathbb{R}^D$ is called *union of (generalized) arcs* if there are two constants $n^* \in \mathbb{N}, a^* \in \mathbb{R}^{\geq 0}$ such that S is the disjoint union of n^* hyperspherical caps $C_1, \ldots, C_{n^*}$ of the unit hypersphere, where each cap C_i, $i = 1, \ldots, n^*$, is given by

$$C_i = \{x \in \mathbb{R}^D \mid \|x\|_2 = 1 \text{ and } \psi_{D-1} \in [\alpha_i, \beta_i]\}$$

for $\alpha_i, \beta_i \in \mathbb{R}$ with $\beta_i - \alpha_i \geq a^*$, i. e., is defined on a interval of constant size with respect to the polar angle ψ_{D-1}.

Since we mostly will work with $D = 2$, we prefer the term *union of arcs* instead of the generalized "union of hyperspherical caps" or similar in the following. Obviously, from the problem definitions it immediately follows that the problems defined above are all union of arc problems.

3 RUNTIME ANALYSIS

In this section, we conduct rigorous runtime analyses of the (1+1) NA with harmonic and local mutation on the example problems defined in Section 2.4. We exclusively consider the case $D = 2$ here, which justifies the use of the terms for 2 dimensions like "unit circle", "line" etc. instead of the general "unit hypersphere (surface)", "hyperplane" etc. Extensions of the analyses to larger dimensions seem promising, but would require inputs of higher dimensionality for the neurons and would introduce additional complexity.

We start our analysis with the simplest of the problems, which can be solved optimally (in the sense defined above) with 1 neuron even if the bias is fixed at 0.

Theorem 3.1. *The expected optimization time of the (1+1) NA with harmonic mutation, $N = 1$ and without bias on Half for $D = 2$ is $O(\log^2 r)$. For the local (1+1) NA it is $O(r)$.*

To prove this and the following theorems, it is crucial to understand how the (1+1) NA can make progress towards solutions of higher fitness. To this end, we give the following definition and characterization of local optima.

Definition 3.2. Let $x_t = (\varphi_1, b_1, \ldots, \varphi_N, b_N)$ be a search point of the (1+1) NA. We call x_t a *local optimum* if there is no x' of strictly larger fitness value that differs from x_t in exactly one component c

and furthermore by either +1 or −1 in that component (modulo r if c denotes an angle and modulo $r+1$ if c denotes a bias).

Lemma 3.3. *Consider the (1+1) NA with $N = 1$ on a unit-of-arcs problem according to Definition 2.6. Assume for the current search point x_t that not both intersecting points of the unit circle and the neuron's line are at a boundary of a positive or negative arc. If at least one of the two following conditions hold, then x_t is not locally optimal: (1) none of the intersecting points is at a boundary and the smallest distance between the endpoints of a positive arc and the intersecting points is at least $2/r$; (2) one intersecting point is at a boundary and there is a negative arc of length at least $2\pi/r$ incident on the other intersecting point.*

The lemma still applies to the case $N > 1$ if all lines belonging to the N neurons classify disjoint areas of $[0,1]^2$ positively, have constant distance from each other, and r is at least a sufficiently large constant.

Proof. We recall that the unit circle is composed of a constant number of arcs of constant size, where each arc is either classified completely positively or completely negatively.

We consider the two cases stated in the lemma. If none of the intersecting points is at a boundary, then it is sufficient to change the bias component of the current search point by ±1, moving the line either closer to a positive arc (if it is above the line) or further away (if below). This reduces the length of the wrongly classified region and does not move the line into a different arc by the assumptions of distance at least $2/r$.

If only one intersecting point is at a boundary, then, by the assumption on the negative arc length of at least $2\pi/r$, changing the angle of the line is possible without decreasing the fitness. More precisely, we change the angle such that the intersecting point, which previously was at a boundary, now lies within a negative arc. If the angle changes by δ, this increases the negative length above the line by δ (since we are dealing with a unit circle). However, the other intersecting point moves by the same amount closer to a positive arc. See Figure 5 for an illustration.

The conclusions above are still valid for larger N if moving or rotating a line preserves disjointness of the N arcs classified positively by the neurons. If r is bounded from below by a sufficiently large constant, this holds since we assume at least constant distance between the N lines (measured within the unit circle). □

With the above tools at hand, we can prove the first theorem.

Proof of Theorem 3.1. By definition, the optimal angle for the problem is $\pi/2$, resulting in the halfspace $\{x \in \mathbb{R}^2 \mid x_2 \geq 0\}$ being classified positively. (Here we use that the bias is fixed at 0.) We consider the local (1+1) NA first and use a classical fitness-level argument [28] for the underlying unimodal fitness landscape to bound the time until the current angle of the (1+1) NA has reached $\pi/2$, corresponding to $\varphi_t = r/4$ in the search space (note that with our definition of optimization time, any angle $\varphi_t \in (r/4-1, r/4+1)$ would be considered as optimal). Let $\xi_t = \min\{|r/4-\varphi_t|, 5r/4-1-\varphi_t\}$, i. e., the smallest distance of φ_t from its optimum $r/4$ in the representation with wrap-around, where 0 is a neighbor of $r-1$.

If $\xi_t = i > 0$ (corresponding, e. g., to an angle $2\pi i/r + \pi/2$), then incrementing or decrementing φ_t by 1 (modulo r) will improve fitness since such a step increases the length of the intersection of the arc above the neuron's line (with normal vector of angle φ_t) and the points in Half. Whether increasing or decreasing (or both) improves fitness depends on whether $\varphi_t < 3r/4$. The local (1+1) NA chooses the improving direction for the angle with probability at least $1/2$ and reduces ξ_t by 1 with probability at least $1/2$. Altogether, the probability of improving is at least $1/4$. Since at most $r/2$ improvements are sufficient, the total expected optimization time is at most $(r/2) \cdot 4 = O(r)$.

For the standard (1+1) NA with Harmonic mutation, we use multiplicative drift analysis [5], inspired by the analysis of the Harmonic mutation on a generalized OneMax function from [3]. Let $\xi_t = i$. Then all decreasing steps of sizes $1, \dots, i$ are accepted. The probability of a decreasing step of size $j \leq i$ is $1/(2jH_r)$, where the factor 2 accounts for the choice of direction. Hence, the expected distance decrease is at least $\sum_{j=1}^{i} \frac{1}{2jH_r} \cdot j = \frac{i}{2H_r}$; in other words, the drift is bounded from below by $i\delta$ with $\delta = 1/(2H_r)$. By the multiplicative drift theorem [5], the expected hitting time of 0 is $O((\ln \varphi_0 + 1)/\delta) = O(\ln^2 r)$. □

We remark that the optimization problem underlying Theorem 3.1 corresponds to the generalized OneMax on $\{0, \dots, r\}^n$ for $n = 1$ as considered in [3]. Hence, it seems straightforward to transfer their results for an advanced self-adjusting mutation to the scenario of Theorem 3.1. However, this is not obvious for the following problems, so we stick to the more simple local and harmonic mutation operators for the rest of this paper.

We proceed now to the problem Quarter, which cannot be solved optimally with one neuron if the bias stays fixed at 0. We show that allowing varying bias leads to polynomial in r expected optimization time for the local mutation and even to polylogarithmic times in special cases for the Harmonic mutation. We remark that the positive arc length of $\pi/2$ has been chosen for simplicity. The analyses also hold for a larger class of problems where the length of the positive arc is some constant value in the interval $(0, \pi)$.

Theorem 3.4. *Let $r = 8k$ for an integer k, let $N = 1$ and allow variable bias. Consider the (1+1) NA with local and harmonic mutation on the problem Quarter. Then:*

1. *The expected optimization time of the local (1+1) NA is $O(r^2)$.*
2. *With at least constant probability, the optimization time of the local (1+1) NA is $O(r)$.*
3. *With harmonic mutation, the expected optimization time is $O(\log^3 r)$.*

We will need the following characterization of fitness, assuming that there is piece of positive length above the line.

Lemma 3.5. *Let $x_t = (b_t, \varphi_t)$ be the current search point of the (1+1) NA with $N = 1$ on Quarter and assume that there is piece of positive length on the unit circle above the neuron's line. Let $d_b^{(t)} = (2b_t/r - 1) - \sqrt{2}/2$ and $d_\varphi^{(t)} = |2\pi\varphi_t/r - \pi/4|$ be the difference of current bias and absolute difference of angle, respectively, from their optimal values and let $\eta_t = 2\arccos(2b_t/r - 1) - \pi/2$ be the difference of the length of the arc above the line from its optimum*

value. If $d_b^{(t)} \geq -\sqrt{2}/2$, then it holds for the current fitness value that

$$f(x_t) = 1 - |\eta_t| - \max\{0, d_\varphi^{(t)} - |\eta_t|/2\}.$$

Proof. By simple trigonometry, the length of the arc above the line is $2\arccos(2b_t/r - 1)$. We distinguish two cases according to $d_b^{(t)}$. If $d_b^{(t)} \leq 0$, which means that the whole positive arc of Quarter has room to be lie completely above the line, a total length of least $\eta_t = 2\arccos(2b_t/r - 1) - \pi/2 \geq 0$ of that arc is negative and therefore wrongly classified. However, if $d_\varphi^{(t)} > \eta_t/2$, then the positive arc of Quarter intersects the line and an additional arc of length $d_\varphi^{(t)} - \eta_t/2$ lies below the line and is wrongly classified. This gives the formula

$$1 - f(x_t) = \eta_t + \max\{0, d_\varphi^{(t)} - \eta_t/2\}.$$

If $d_b^{(t)} > 0$, then $\eta_t < 0$ and at least one endpoint of the positive arc of Quarter is below the line. Moreover, positive arcs of total length at least $-\eta_t = \pi/2 - 2\arccos(2b_t/r - 1)$ are below the line and wrongly classified, where we use that the bias is non-negative due to the assumption $d_b^{(t)} \geq -\sqrt{2}/2$. Together with the additional negative arc above the line in the case $d_\varphi^{(t)} > |\eta_t|/2$, we obtain

$$1 - f(x_t) = -\eta_t + \max\{0, d_\varphi^{(t)} + \eta_t/2\},$$

and the lemma follows. □

We shall show the statements of the Theorem 3.4 separately and start with the simpler case of local mutation analyzed in the first two statements. The second statement considers an initialization where there is a clear gradient towards on optimal solution. The first statement considers general initialization, which may result in a longer random-walk behavior with the line of the neuron being tangent on the unit circle.

For all parts of Theorem 3.4, we will frequently use the following helper lemma. Often the line ℓ considered in the lemma corresponds to an optimal placement of the neuron's line.

Lemma 3.6. *Let ℓ be a line passing through the unit circle and p be a point on the unit circle. Let d_ℓ be the distance of ℓ from the origin and let d_p be the distance between the origin and the line that is parallel to ℓ and passes through p. Assume that both d_ℓ and d_p less than* 1.

Suppose that p is rotated by an angle of ρ on the circle such that it moves either closer to or further away from ℓ during the whole rotation and does not change side w. r. t. ℓ. Then there are constants $0 < c_1 < c_2$ such that d_p reduces by at least $c_1\rho$ and at most $c_2\rho$. Similarly, if ℓ is moved closer to p by an amount of $\delta > 0$, then the arc between ℓ and p decreases by at least $c_3\delta$ and at most $c_4\delta$ for constants $0 < c_3 < c_4$.

Proof. The distance d_p is given by the sine of the angle α between ℓ and the line passing through the center point and p. Now, since moving the point decreases the angle by a constant, the first claim follows by noting that sine is monotone increasing in the considered ranges and that its derivative is constant as soon as the angle has moved away from $\pm\pi/2$. The second claim follows analogously with a linear approximation of the arcsine. □

We now give the proofs of the second statement and afterwards of the first statement of Theorem 3.4.

Proof of 2nd statement of Theorem 3.4. This statement considers beneficial initializations of angle and biases. If the set Quarter (an arc of length $\pi/2$) intersects or lies completely above the line of the neuron in the initial solution, then fitness is improved by rotating the line to increase the arc length above the line or increasing the bias (or both). We consider the event of an initial bias in $(0, 0.5]$ (i. e., $b_0 \in [r/2 + 1, 3r/4]$ in the initialization) and an initial angle strictly in between $a := \arcsin(0.5)$ and $\pi/2 - a$ (roughly corresponding to $\varphi_0 \in (0.0833r, 0.1667r)$), which has constant probability. Simple geometry then shows that the Quarter arc is completely above the line of the neuron. This implies that the points in Quarter and the region below the line, which comprise more than half of the circle because of the positive bias, are correctly classified. Hence, the fitness of such a search point is strictly larger than 3/4. It is not possible to achieve such a fitness without having the Quarter arc partially or fully above the neuron's line, so this property will be maintained during the run.

To analyze fitness improvements, we consider two types of steps:

(1) increasing the bias, i. e., moving the line closer to the positive arc of Quarter without moving any of its points below the line,
(2) changing the angle such that the arc of Quarter appears more centered above the line of the neuron; formally, the absolute difference between angle and $\pi/4$, i. e., the quantity $d_\varphi^{(t)}$ from Lemma 3.5 decreases. This may be necessary to allow a further increase in bias and fitness. Figure 5 depicts a situation where a type-2 step has to be applied before further improvements.

The characterization of Lemma 3.3 shows that type-1 steps or type-2 steps are available before the line has found its optimal position (up to the allowed tolerance $\pm O(1/r)$). Moreover, type-1 steps strictly improve fitness (unless the bias component of the search point has reached the optimum value ± 1). As long as type-1 steps are available (i. e., can improve fitness), there is a probability of at least $\Omega(1)$ of not changing the angle and mutating the bias in the desired direction (increasing it by $2/r$). By Lemma 3.5, this increases the fitness since η_t decreases. This fitness improvement is at least $\Omega(1/r)$ since the length of wrongly classified region above the line decreases linearly with the increase of bias by Lemma 3.6. Here we exploit the bias considered here is at least 0 and at most $\sqrt{2}/2 + 1/r$, which is by a constant away from 1, i. e., the radius of the unit circle.

When a type-1 step is not possible, a type-2 step may decrease the quantity $d_\varphi^{(t)}$. Such steps do never decrease fitness and are accepted, however, they do not necessarily increase fitness if Quarter is already completely above the line. If neither type-1 nor type-2 steps are available, both angle and bias are within an additive distance of less than 1 from the optimum in the representation. This holds since we assume that r is a multiple of 8, so the optimum angle of $\pi/4$ can be represented as $r/8$ in the search point. Hence, when the angle takes precisely its optimum value, the bias may decrease to its optimum $\sqrt{2}/2$ within the error $O(1/r)$ introduced by the discretization. This is the desired state that we analyze the algorithm to reach.

If a type-2 step is available that increases fitness, then we are in the situation $d_\varphi^{(t)} \geq \eta_t/2$ of Lemma 3.5 and the fitness improvement is at least $1/r$ (using again that r is a multiple of 8, so $d_\varphi^{(t)}$ cannot take values strictly in between 0 and $1/r$). Finally, we have to analyze fitness progress in the situation that only type-2 steps are available that do not decrease fitness since $d_\varphi^{(t)} \leq \eta_t/2$. A type-2 step may decrease $d_\varphi^{(t)}$ and thereby increase the distance of closest endpoint of the positive Quarter arc from the neuron's line. If this distance increases by at least $1/r$, a type-1 step becomes available. Hence, using again Lemma 3.6 and noting that old and new bias are bounded by a constant less than 1 and at least 0, rotating the angle by at least c/r for a sufficiently large constant c increases the distance of the closest endpoint of the Quarter arc from the line of the neuron by at least $2/r$. Thus, c consecutive steps rotating in one direction are sufficient, which happens with constant probability at least $(1/4)^c$. Altogether, there is a constant probability of changing the angle by at least c/r in the desired direction and afterwards increasing the bias by $2/r$, i. e., a sequence of type-2 steps and a type-1 step. Hence, as long the algorithm is not yet within the allowed distance of the optimal bias, there is a constant probability of improving fitness by $\Omega(1/r)$. Altogether, in expected $O(r)$ steps the (1+1) NA has reached the desired state. □

Proof of 1st statement of Theorem 3.4. Again, we use the ideas from Lemma 3.3. Hence, there is always a constant probability of increasing the fitness unless either the global optimum has been reached (in the sense of an error of $O(1/r)$ as described above) or the bias has reached its maximum value 1. If a part of Quarter is above the line, then fitness increases by moving a larger part above the line without moving too many of the negative points below it. If, however, Quarter is completely below the line, then fitness increases by decreasing the length of the arc above the line (consisting of negative points only). Hence, an increase of bias increases fitness in this situation, up to the point where it reaches 1 and, up to an intersection of volume 0, the whole unit circle is below the line. Such a state corresponds to a fitness of 3/4. Decreasing the bias is not accepted in this situation unless it moves parts of the positive Quarter arc above the neuron's line. As soon as this happens, fitness increases above 3/4 and an optimum is bound in expected time $O(r)$ by decreasing bias and rotating the line until reaching bias $\sqrt{2}/2 \pm O(1/r)$ and angle $\pi/4$, using the same arguments as in the analysis of the first statement. Also, again using the same arguments, in expected time $O(r)$ the bias reaches 1 if the optimum is not found before. Hence, we only have to analyze the time, starting from a bias of 1, until the angle enters the interval $(0, \pi/4)$. In this situation there is a constant probability of decreasing the bias (without changing the angle).

To complete the proof, we consider the random walk of the angle while the bias is 1. Formally, let X_t be the representation of the angle at time $t \geq 0$ (i. e., as an integer in $\{0, \dots, r-1\}$), where time 0 corresponds to the first point in time with bias 1. We consider the first hitting time $T := \min\{t \geq 0 \mid X_t \in [0, r/4]$, assuming $X_0 \in (r/4, r-1]$. If the bias does not change, the random walk takes independently in each step a uniform decision to increase or decrease the angle, and the absolute change is independently drawn from the same distribution. Hence, we have $\mathrm{E}[X_{t+1} \mid X_t] \leq X_t$, i. e., a supermartingale, where the inequality stems that the mutation is taken modulo r. We pessimistically assume the case of a martingale. Clearly, since the change has constant variance and uniform random sign, the variance satisfies $\mathrm{Var}[X_{t+1} \mid X_t] \geq c$ for a constant $c > 0$. Hence, by the upper bound for martingale drift (Corollary 26 in [17]), we have $\mathrm{E}[T] = O(r^2)$. By Markov's inequality, $T = O(r^2)$ with constant probability, and with altogether constant probability after $O(r^2)$ steps the bias decreases to less than 1.

Finally, the total expected runtime is $O(r^2)$ by a standard restart argument. Formally, we can consider independent phases of length $O(r^2)$ and constant success probability. The expected number of such phases is $O(1)$. □

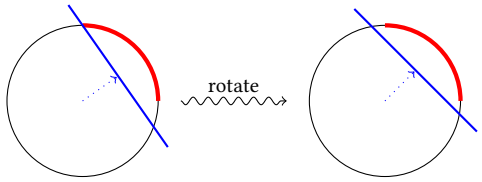

Figure 5: The line can be rotated without changing fitness.

To complete the analysis, we still have to analyze the standard (1+1) NA with the global, harmonic mutation. While several ideas from the case with local mutation will reappear, the analysis is more complex and requires a careful study of the decomposition of the fitness value from Lemma 3.5, along with more advanced drift arguments. We will need the following helper statement.

Lemma 3.7. *Let X denote the random outcome of the harmonic mutation operator with parameter r. Let a, b, where $a < b$, be two positive integers. Then* $\mathrm{Prob}(a < X \leq b) \geq (\ln(b/a) - 1/a)/H_r$

Proof of Lemma 3.7. We compute

$$\begin{aligned}\mathrm{Prob}(a < X \leq b) &= \sum_{j=a+1}^{b} \frac{1}{jH_r} = \frac{1}{H_r}(H_b - H_a) \\ &\geq \frac{\ln(b) - \ln(a) - \frac{1}{a}}{H_r} = \frac{\ln(b/a) - \frac{1}{a}}{H_r},\end{aligned}$$

where we bounded the Harmonic sums by integrals using upper and lower sums. □

Lemma 3.7 is often used for the case that $b - a = \Omega(r)$, which gives a probability of $\Omega(1/\ln r)$ of hitting the interval $(a, b]$. We can now give the analysis of the harmonic mutation on Quarter.

Proof of 3rd statement of Theorem 3.4. First, let us assume that we already have a fitness of at least 3/4 to explain the main idea. At the end of the proof, we will deal with a general initialization.

In the following, we study the effect of the global, harmonic mutation in the analysis of the fitness improvements and progress of the line towards its optimal state at bias $b^* := \sqrt{2}/2$ and angle $\pi/4$. Note that the line might change globally in one step, from a state with bias less than b^* and the whole positive arc and some negative parts above it to a state with bias greater than b^* and a strict subset of the positive arc and no negative points above it. However, the assumption of fitness strictly larger than 3/4 gives us

two invariants: (1) the bias must be non-negative (otherwise, more than 1/4 of the negative parts would be above the line and classified incorrectly) and (2) that there will always be some positive (sub)arc of Quarter above the line (as already exploited in the proof of the 2nd statement).

To analyze fitness improvements, we are in the same mindset as in the proof of the first two parts and distinguish between changes of angle and changes of bias. Changes of angle are beneficial if the bias is less than b^* and the positive arc is partly below the line. Let ξ be length of positive arc part below the arc. Then rotating the line by ξ immediately improves fitness by ξ. Changes of bias become relevant if both end points of the positive arc are on the same side of the line. Then fitness improves by changing bias, either by bringing more positive points (if bias is decreased) or fewer negative points (if bias is increased) above or on the line.

With this rough strategy in mind, we shall study the fitness distance $g_t = 1 - f(x_t) \geq 0$ over time and conduct a multiplicative drift analysis. To this end, we express g_t as a sum of two (approximately) linear functions and distinguish between different cases and subcases. In the notation of Lemma 3.5, we consider the case $d_b^{(t)} < 0$ first, which means that the whole positive arc of Quarter has room to lie completely above the line. Recalling the lemma, we have the following characterization of the fitness distance: $g_t = \eta_t + \max\{0, d_\varphi^{(t)} - \eta_t/2\}$.

We want to show the multiplicative drift $\mathrm{E}[g_t - g_{t+1} \mid g_t] \geq \delta g_t$ for some $\delta = \Theta(1/\ln^2 r)$. Here we distinguish between two subcases. If $\max\{0, d_\varphi^{(t)} - \eta_t/2\} > \eta_t/2$ (i. e., $d_\varphi^{(t)} > \eta_t$), then, as explained in Lemma 3.5, the two endpoints of the positive arc are on different sides of the line and we analyze steps keeping the bias but moving the angle closer to $\pi/4$. Any change of angle by an amount $\xi \in \Delta^* := [0, d_\varphi^{(t)} - \eta_t/2]$ is accepted (assuming that the bias is not mutated in the same step) and decreases g_t by ξ. Changes of larger amounts (up to $d_\varphi^{(t)}$) are also accepted, as we shall exploit in a special case below, but do not lead to an additional improvement of the second component of g_t.

In our discretized representation, the changes of angle in the set Δ^* correspond to steps of size j of the angle component in the search point for $j = 1, \ldots, \lfloor (r/(2\pi))(d_\varphi^{(t)} - \eta_t/2) \rfloor$. If $d_\varphi^{(t)} - \eta_t/2 < 2\pi/r$, the previous floor function is 0, but then already a step of size 1 in the search space, changing angle in the right direction and not changing bias, reduces the $d_\varphi^{(t)}$-value sufficiently to have $d_\varphi^{(t+1)} < \eta_{t+1}/2$ and therefore 0 contribution of the second component of g_t. Otherwise, i. e., if $d_\varphi^{(t)} - \eta_t/2 \geq r/(2\pi)$, then $\lfloor (r/(2\pi))(d_\varphi^{(t)} - \eta_t/2) \rfloor \geq (r/(4\pi))(d_\varphi^{(t)} - \eta_t/2)$ and the expected progress in the potential space through the steps of size j is at least

$$\sum_{j=1}^{\lfloor (r/(2\pi))(d_\varphi^{(t)} - \eta_t/2) \rfloor} \frac{1}{jH_r} \cdot j \geq \frac{(r/(4\pi))(d_\varphi^{(t)} - \eta_t/2)}{H_r},$$

so along with the probability of not mutating bias, which is $1 - 1/(2N) = 1/2$, we have

$$\mathrm{E}[g_t - g_{t+1} \mid g_t] \geq \frac{(d_\varphi^{(t)} - \eta_t/2)}{8H_r} \geq \frac{g_t}{24H_r}$$

using that $g_t \leq 3(d_\varphi^{(t)} - \eta_t/2)$ by assumption (following from the representation of g_t and the condition for the present case). Hence, we even have $\delta \geq 1/(24 \ln r + 24)$ in this case.

In the other subcase, i. e., if $\max\{0, d_\varphi^{(t)} - \eta_t/2\} \leq \eta_t/2$, we consider fitness improvements made by bringing the angle closer to its optimum $\pi/4$ and an increase of bias that becomes possible as a consequence of the new angle. More precisely, the step should result in $-d_b^{(t+1)} \in [|d_b^{(t)}/2|, |d_b^{(t)}|]$ and $d_\varphi^{(t+1)} \leq d' - \pi/4$, where $d' := \arccos(b^* + d_b^{(t)}/2)$, which means that the endpoints of the positive arc cannot lie below the line after the considered improvement of bias. We analyze the probability of the desired change of angle first. Note that any $(r/2\pi)\varphi_{t+1} \in [\pi/2 - d', d']$ fulfills the desired change. We bound the length of the interval and obtain, using $\arccos(b^*) = \pi/4$ and a Taylor approximation for the arccos, that $d' - (\pi/2 - d') = 2d' - \pi/2 = 2\arccos(b^* + d_b^{(t)}/2) - \pi/2 \geq c|d_b^{(t)}|$ for a constant $c > 0$. Hence, the target interval for the angle has length at least $(r/(2\pi))c|d_b^{(t)}|$ in the search point representation. Without loss of generality, this bound is an integer by choosing c appropriately. Moreover, since we assume for the current angle that $d_\varphi^{(t)} < \eta_t = 2(\arccos(b^* + d_b^{(t)}) - \pi/4) \leq c'|d_b^{(t)}|$ for another constant $c' > c$, the maximum change of the angle over the target interval is bounded from above by $(r/(2\pi))c'|d_b^{(t)}|$, again in the search point representation. Again, the bound may be assumed as an integer. Together, the probability of changing the angle as desired is at least

$$\sum_{j=(r/(2\pi))(c'-c)|d_b^{(t)}|}^{(r/(2\pi))c'|d_b^{(t)}|} \frac{1}{jH_r} = \Omega(1/H_r)$$

using Lemma 3.7, which even holds if $d_b^{(t)}$ depends on r. Assuming the desired change of angle, the expected decrease in bias is at least

$$\sum_{j=1}^{\lfloor (r/2)(d_b^{(t)}/2) \rfloor} \frac{1}{jH_r} \cdot \frac{2j}{r} \geq \frac{|d_b^{(t)}|}{2H_r},$$

using similar arguments as above to analyze the rounding effects. (If $\lfloor (r/2)(d_b^{(t)}/2) \rfloor = 0$, then a step of size 1 in the search space suffices.)

Combining with the probability of changing the angle as desired, the unconditional drift is $\Omega(|d_b^{(t)}|/\ln^2 r)$. Finally, we note that $|d_b^{(t)}| = b^* - \cos(\arccos(2b_t/r - 1)) = b^* - \cos(\eta_t/2 + \pi/4)$, so using $\cos(\pi/4) = b^*$ and a Taylor expansion, we have $|d_b^{(t)}| = \Omega(\eta_t)$. Altogether, since $g_t \leq (3/2)\eta_t$ in the present subcase, we have

$$\mathrm{E}[g_t - g_{t+1} \mid g_t] = \Omega(g_t/\ln^2 r).$$

We still have to deal with the case $d_b^{(t)} > 0$, i. e., the bias is greater than its optimum value so that at least one end point of the positive arc of Quarter is below the line. Accordingly, the formula for g_t derived from Lemma 3.5 reads

$$g_t = -\eta_t + \max\{0, d_\varphi^{(t)} + \eta_t/2\}.$$

The analysis proceeds as before, except for some flipped signs. More precisely, the subcase that $d_\varphi^{(t)} + \eta_t/2 > |\eta_t|/2$ is handled in the same way and we obtain a drift of $\Omega(g_t/\ln^2 r)$. In the complementary

subcase $d_\varphi^{(t)} + \eta_t/2 \leq |\eta_t|/2$, we consider mutations of angle that allow a decrease of bias to a value in $[b^* + d_b^{(t)}/2, b^* + d_b^{(t)}]$. Also this probability is bounded in the same way as above. Finally, such a decrease of bias changes the absolute η_t-value as described above, and we obtain the same asymptotic drift bound for the g_t-value.

Altogether, having established a multiplicative drift for g_t with $\delta = \Omega(1/\ln^2 r)$ in all cases, the bound $O(\log^3 r)$ on the optimization time follow from the multiplicative drift theorem and $x_{\min} = 1/r$, noting that the smallest possible fitness distance is $\Theta(1/r)$.

We still have to consider an arbitrary initialization. As soon as the fitness is strictly larger than 3/4, we are in the setting from above and have an expected optimization time $O(\log^3 r)$. Hence, it suffices to analyze the expected time until reaching fitness larger than 3/4. For this it is sufficient to mutate the current bias to a value in $(0, 0.5]$ and an the angle to a value strictly in between $a := \arcsin(0.5)$ and $\pi/2 - a$, reusing the analyses for the beneficial initialization from the proof of the 2nd property. Now, the target intervals for bias and angle both have a length of $\Omega(r)$ with respect to our search space representation. Hence, again using Lemma 3.7, such a mutation has probability $\Omega(1/\log^2 r)$ and the expected time to reach fitness larger than 3/4 is therefore $O(\log^2 r)$, which is a lower-order term. □

We now turn to the case of more than one neuron, which is necessary to achieve the best possible fitness 1 on the problem TwoQuarters. As mentioned above, with only 1 neuron, the fitness cannot exceed 3/4.

Theorem 3.8. *Let $r = 8k$ for an integer k. With at least constant probability, the optimization time of the standard (1+1) NA and the local (1+1) NA with $N = 2$ on the problem TwoQuarters is $O(\log^3 r)$ and $O(r)$, respectively.*

For $N = 1$, the same bounds on the optimization time as in as Theorem 3.4 apply.

Proof. The second paragraph follows in the same way as Theorem 3.4. The only difference is that there are two global optima for the location of the line. Again, if fitness improvements are found by increasing the bias and moving the line towards the boundary of the unit circle, then we analyze the time to arrive at a beneficial angle that allows fitness improvements by lowering the bias again.

We now turn to the first paragraph of the theorem, again using similar arguments as in the proof of Theorem 3.4. The difference is now that there are two lines that the algorithm can move and rotate. The global optimum is taken when the angles are $\pi/4$ and $5\pi/4$ and both biases are $\sqrt{2}/2$. We will consider an initialization where the two lines are initialized in the two "basins of attraction" belonging to this optimal setting. Moreover, the distance of the lines (measured within the circle) will be at least constant to allow an application of Lemma 3.3.

Let (φ_1, b_1) and (φ_2, b_2) be the initial angles and biases of the two neurons. We consider the joint event that $2rb_1 - 1 \in [0.6, 0.65]$, $2rb_2 - 1 \in [-0.6, -0.65]$, $2\pi r\varphi_1 \in [\pi/4 - \alpha^*, \pi/4 + \alpha^*]$ and $2\pi r\varphi_2 \in [5\pi/4 - \alpha^*, 5\pi/4 + \alpha^*]$ for $\alpha^* = \arccos(0.6) - \pi/4 = 0.141897\ldots$. This event happens with constant probability. Then, by simple trigonometry, the positive arc of TwoQuarters in the first quadrant lies above the line of the first neuron and the arc in the third quadrant above the line of the second neuron. Moreover the choice of bias leaves strictly less than a quarter of the points above either line wrongly classified. Altogether, this initialization gives a total fitness of more than 3/4. This has the following implications on the future placement of the two lines. To achieve at least the same fitness with a different placement of the lines, at least a part of each of the two positive arcs of Quarter must lie above a line, and each line must have a positive part above it. Otherwise, a positive arc of length $\pi/2$ would lie below both lines and the fitness could not be greater than 3/4. Hence, one line cannot move completely above the other. Moreover, with the assumed fitness, it is impossible to reach placements such that the lines intersect each other within the unit circle. If such an intersection happened, since both lines have positive arc pieces above them, the angle between the lines, taken in the area above them, would be at least π. Then also a negative part of the unit circle of length at least $\pi/2$ would lie above a line, contradicting the fitness strictly larger than 3/4.

We now complete the proof re-using the analyses from Theorem 3.4 under the beneficial initialization. We consider the local (1+1) NA first. Then there is for each hyperplane an event or a sequence of constant many events of constant probability that brings the positive arc in the respective quadrant (1st or 3rd) closer to the line, i. e., decreases the total length of wrongly classified arcs above the line without making the length of wrongly classified points below the hyperplane bigger. This holds until each bias deviates by at less than $2/r$ from its optimum $\sqrt{2}/2$. The expected time to reach this state (still assuming the beneficial initialization) is $O(r)$, and along with Markov's inequality the joint probability of finding the global optimum in $O(r)$ steps is at least constant.

For the standard (1+1) NA with harmonic mutation, express the fitness distance as a generalization of the expression in Lemma 3.5, where we consider the quantities $d_b^{(t)}$ and $d_\varphi^{(t)}$ separately for the two lines and add up the wrongly classified parts in the style of $\eta_t + \max\{0, d_\varphi^{(t)} - \eta_t/2\}$ for both lines, noting that the lines do not intersect each other. Then we conduct the analysis from the proof of Theorem 3.4 (conditioning on that a step only changes the parameter of one neuron) and obtain an expected time of $O(\log^3 r)$ until the fitness distance has reached its minimum (± 1 in the search point representation). We arrive at the claimed bound $O(\log^3 r)$. □

The constant success probability implies that multi-start variants of the algorithm are highly efficient. See, e. g., [29] for definitions and analyses of multi-start schemes. However, without restarts it is not clear whether there is a general finite bound for the local (1+1) NA and a bound for the harmonic (1+1) NA that is better than $O((r \log r)^c)$ for a constant $c < 1$ (see also the worst case bound from Lemma 2.1). The problem is that the two lines of the neurons may intersect in such a way that exactly one positive quadrant of TwoQuarters is classified positively. This situation essentially "tilts" and locks the lines from moving, except for a random walk of the intersecting point. See the left-hand side of Figure 6 for an illustration. Also, one line could be lying completely above the other one and therefore be irrelevant for fitness evaluation. This irrelevant line can freely perform a random walk about the relevant one, i. e., in contrast to the analysis of Theorem 3.4 (part 1), the random walk would not be limited to configurations with bias 1. See the right-hand side of Figure 6.

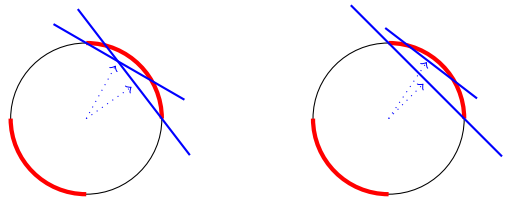

Figure 6: Left: two lines are tilted and impede further progress. Right: one line lies completely above the other and does not contribute to fitness.

As a next step, one could consider generalized definitions of TwoQuarters to problems with k equally spaced positive arcs of identical size. It is not difficult to see that these would require $N = k$ neurons to achieve fitness 1. Moreover, a generalization of Theorem 3.8 seems plausible, with a probability of at least c^{-k} for a constant c for optimizing the problem efficiently. In any case, it would be interesting to consider this problem in an analysis where both r and N are asymptotically growing.

Finally, we turn to the problem LocalOpt, which possesses a local optimum that is hard to escape and causes infinite expected optimization time for the local (1+1) NA. However, for the standard (1+1) NA with harmonic mutation, it is not difficult.

Theorem 3.9. *With at least constant probability, the local (1+1) NA with $N = 1$ on the problem LocalOpt reaches a local optimum of fitness at most $2/3$ and cannot make improvements from there.*

However, the expected optimization time of the (1+1) NA with harmonic mutation is $O(\log^3 r)$.

Proof. We consider an initialization where

- the line belonging to the neuron intersects the two negative arcs of the unit circle with polar angle in the interval $[\pi/3, 2\pi/3]$ and $[11\pi/6, 2\pi]$ and
- the positive area of LocalOpt corresponding to angles in the interval $[0, \pi/3]$ is completely below the line.

This corresponds to a location shown in the right-hand side of Figure 4 up to shifts of the hyperplane within the negative area mentioned above. The fitness of such an initial search point is between $5/12$ and $2/3$. A sufficient condition for this initialization is a bias within $[-\cos(18\pi/48), -\cos(19\pi/48)]$ and an angle within $[7/12\pi - a, 7/12\pi + a]$, where $a = \pi/48$. Since the considered initialization specifies intervals of length $\Omega(r)$ for both angle and radius, its probability is at least a constant.

To improve the fitness to above $2/3$ it is necessary to change the angle of the line in one step by at least $\pi/6$, which is impossible with the ± 1 mutation of the (1+1) NA.

For the harmonic mutation, we analyze the event of a mutation leading to fitness strictly larger than $2/3$. For this it is sufficient to mutate the bias to an interval of sufficiently small but constant size around 0 and the angle to an interval of sufficiently small constant size around $11\pi/6$. By Lemma 3.7, the expected time for this to happen is $O(\log^2 r)$. Afterwards, we can express the fitness distance in a way analogous to the proof of the first statement of Theorem 3.4), except for that there is always a negative arc of length $\pi/6$ above the line that is classified wrongly. The expected time to reach optimal bias and angle (up to the error ± 1 in the search point representation) is $O(\log^3 r)$ by the same multiplicative drift analysis as in the proof of Theorem 3.4. □

Theorem 3.9 does not exclude that multi-start variants of the local (1+1) NA are efficient; in fact we think that with at least constant probability, it finds the global optimum of fitness $3/4$ in polynomial time.

4 EXPERIMENTS

We ran the (1+1) NA with local and hybrid mutation on the problems Half, Quarter and LocalOpt with $N = 1$ and on TwoQuarters with $N = 2$, each 100 times. We canceled the runs after stagnation phases without fitness improvement of length $100r \log r$, where the latter choice was inspired by the lower bound $1/(rH_r)$ of the harmonic mutation operator hitting an arbitrary state. Finally, for Quarter with $N = 2$ we considered not only the ANN from Figure 2 with hard-wired OR in the output layer (which was the network assumed in Theorem 3.8), but also tried a variant of the ANN where the parameters of the final neuron were evolved in the same way as the first two neurons.

Experiments for $D = 2$ and r growing between 120 and 1200 supplement the theoretical running time bounds from Section 3 nicely. In particular, the high standard deviation for TwoQuarters show that the polylogarithmic running time in case of beneficial initialization is not always the case and that the pathological configurations shown in Figure 6 are hard to overcome. However, with the timeouts specified above the hybrid mutation was generally efficient on all problems, while the local mutation struggled especially on LocalOpt and TwoQuarters. Detailed tables of all experimental data are given in [10].

We also ran CMA-ES, a state-of-the-art evolutionary algorithm for continuous spaces [14], on our benchmark problems, still fixing $D = 2$. To achieve a fair comparison with the (1+1) NA, we use essentially the same representation with polar angle and bias on the intervals $[0, 2\pi]$ and $[-1, 1]$, respectively, enforced in CMA-ES by bounding the components of its real-valued vectors. Experiments were performed using the `cma 3.3.0` Python package, initialized with a standard deviation of 1, expected starting solution in the origin and default stopping criterion. On Half and Quarter, CMA-ES performed generally well and found the optimum almost always within 2% of optimality and even more frequently within 5%; in very rare cases on Quarter, the algorithm was stuck more than 50 % from the optimum. The picture was similar on LocalOpt, which was solved to optimality (up to 2% tolerance) in almost all cases. The unsuccessful runs were usually stuck at fitness about $2/3$, corresponding to a local optimum.

On TwoQuarters with a hard-wired OR in the output layer (as shown in Figure 2), CMA-ES frequently missed the optimum and was stuck at fitness $3/4$; the success rate was only about 33.3 %. Again, we also tried the variant of the ANN where the parameters of the final neuron were evolved as well. Here only a success rate of 1.8 % was observed.

The experiments on TwoQuarters where the (1+1) NA with hybrid mutation had a higher success rate than CMA-ES may be biased by the default stopping criterion in CMA-ES. A longer stagnation phase may enable it to find the global optimum more frequently. We

also experimented with a Cartesian representation for the neurons' weights in the CMA-ES, but did not observe improvements.

Additional experiments showed that the approach taken in this paper is successful also in higher-dimensional settings. The hyperplanes (neurons) are still represented by a normal vector in spherical coordinates (angles) and a bias. The angles and the bias are changed with fixed step width. Two-layer networks were used, where the single neuron in the output layer computed a Boolean function. The test were run on point sets. A simple example is a set of eight points located at the corners of a cube. Four non-neighboring corners are being labeled "1" and the others "0". In most cases, the network was able to reach a perfect classification efficiently.

We also have experimentally investigated using a continuous, heavy-tailed distribution to mutate angle and bias instead of the discrete harmonic distribution. The experiments do not show a qualitative difference, i. e., optimal solutions are found with similar frequency. However, the number of steps to reach the optimum varies depending on the chosen distribution (shifted Pareto, exponential, Cauchy) and the setting of the parameters for the distribution. We found parameter settings which gave a performance similar to the one when using the discrete, harmonic distribution but did not achieve a significant performance gain over the harmonic distribution. This clearly is a field for further investigation, both theoretical and experimental.

5 CONCLUSIONS

We have proposed an algorithmic framework for the runtime analysis of problems in neuroevolution. The framework comprises a simple evolutionary algorithm called (1+1) NA for the optimization of parameters of neurons, more precisely weights and biases, and a scalable network structure with two layers (hidden and output) as search space for optimization problems. We also have proposed simple benchmarks based on labeled points on the unit hyperspere and used them to illustrate typical behavior and challenges for the search trajectory of the (1+1) NA. We have identified problems with local optima and compared two types of mutation operators, where the so-called harmonic mutation often gives exponentially better runtime bounds. Experimental supplements show that the proven runtime bounds and performance difference are pronounced in practice already for small problem sizes.

In this first study of the runtime of neuroevolutionary algorithms, we have only scratched the surface of the rich structure arising already from very simple problems. So far we are working with fixed structures for the artificial neural networks, while state-of-the-art neuroevolutionary algorithms would also evolve the networks' topology. Moreover, the present runtime analyses are limited to the case of 2 dimensions, while the general problem definitions call for an analysis in higher dimensions. Furthermore, more advanced classification problems could be considered. We see also room for improvement in the search operators. For example, in some cases, self-adaptation of the mutation strength may lead to a runtime of $O(\log(1/\epsilon))$ to achieve an ϵ-approximation of the optimum. We leave all these considerations as subjects for future research.

Acknowledgement. This work was supported by a grant from the Independent Research Fund Denmark (grant no. 2032-00101B).

REFERENCES

[1] Avrim Blum and Ronald L. Rivest. 1988. Training a 3-Node Neural Network is NP-Complete. In *Proc. of NIPS 1988*. Morgan Kaufmann, 494–501.
[2] Martin Dietzfelbinger, Jonathan E. Rowe, Ingo Wegener, and Philipp Woelfel. 2011. Precision, Local Search and Unimodal Functions. *Algorithmica* 59, 3 (2011), 301–322.
[3] Benjamin Doerr, Carola Doerr, and Timo Kötzing. 2018. Static and Self-Adjusting Mutation Strengths for Multi-valued Decision Variables. *Algorithmica* 80, 5 (2018), 1732–1768.
[4] Benjamin Doerr, Carola Doerr, and Johannes Lengler. 2021. Self-Adjusting Mutation Rates with Provably Optimal Success Rules. *Algorithmica* 83, 10 (2021), 3108–3147.
[5] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. 2012. Multiplicative Drift Analysis. *Algorithmica* 64, 4 (2012), 673–697.
[6] Benjamin Doerr and Frank Neumann (Eds.). 2020. *Theory of Evolutionary Computation – Recent Developments in Discrete Optimization*. Springer.
[7] Benjamin Doerr and Frank Neumann. 2021. A survey on recent progress in the theory of evolutionary algorithms for discrete optimization. *ACM Transations on Evolutionary Learning and Optimization* 1, 4 (2021), 1–43.
[8] Benjamin Doerr and Sebastian Pohl. 2012. Run-time analysis of the (1+1) evolutionary algorithm optimizing linear functions over a finite alphabet. In *Proc. of GECCO 2012*. ACM Press, 1317–1324.
[9] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2002. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science* 276, 1-2 (2002), 51–81.
[10] Paul Fischer, Emil Lundt Larsen, and Carsten Witt. 2023. First Steps towards a Runtime Analysis of Neuroevolution. (2023). arXiv preprint 2307.00799, https://arxiv.org/abs/2307.00799.
[11] Edgar Galván and Peter Mooney. 2021. Neuroevolution in Deep Neural Networks: Current Trends and Future Challenges. *IEEE Transactions on Artificial Intelligence* 2, 6 (2021), 476–493.
[12] Christian Gunia. 2005. On the analysis of the approximation capability of simple evolutionary algorithms for scheduling problems. In *Proc. of GECCO 2005*. ACM Press, 571–578.
[13] George T. Hall, Pietro S. Oliveto, and Dirk Sudholt. 2020. Fast Perturbative Algorithm Configurators. In *Proc. of PPSN 2020*, Vol. 12269. Springer, 19–32.
[14] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9, 2 (2001), 159–195.
[15] Jens Jägersküpper. 2007. Algorithmic analysis of a basic evolutionary algorithm for continuous optimization. *Theoretical Computer Science* 379, 3 (2007), 329–347.
[16] Thomas Jansen. 2013. *Analyzing Evolutionary Algorithms – The Computer Science Perspective*. Springer.
[17] Timo Kötzing and Martin S. Krejca. 2019. First-hitting times under drift. *Theoretical Computer Science* 796 (2019), 51–69.
[18] Timo Kötzing, Andrei Lissovoi, and Carsten Witt. 2015. (1+1) EA on Generalized Dynamic OneMax. In *Proc. of FOGA 2015*. ACM Press, 40–51.
[19] David J. Montana and Lawrence Davis. 1989. Training Feedforward Neural Networks Using Genetic Algorithms. In *Proc. of IJCAI 1989*. Morgan Kaufmann, 762–767.
[20] Frank Neumann and Carsten Witt. 2010. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer.
[21] Eduardo Carvalho Pinto and Carola Doerr. 2017. Discussion of a more practice-aware run-time analysis for evolutionary algorithms. In *Proc. of Artificial Evolution (EA 2017)*. 298–305.
[22] Franz Rothlauf. 2006. *Representations for Genetic and Evolutionary Algorithms* (2nd ed.). Springer.
[23] Kenneth O. Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. 2019. Designing neural networks through neuroevolution. *Nature Machine Intelligence* 1, 1 (2019), 24–35.
[24] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.
[25] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press, Cambridge, MA.
[26] Hamit Taner Ünal and Fatih Başçiftçi. 2022. Evolutionary Design of Neural Network Architectures: A Review of Three Decades of Research. *Artificial Intelligence Review* 55, 3 (2022), 1723–1802.
[27] Jiří Šíma. 2002. Training a Single Sigmoidal Neuron Is Hard. *Neural Computation* 14, 11 (2002), 2709–2728.
[28] Ingo Wegener. 2001. Theoretical Aspects of Evolutionary Algorithms. In *Proc. of ICALP 2001*, Vol. 2076. Springer, 64–78.
[29] Ingo Wegener. 2005. Simulated Annealing Beats Metropolis in Combinatorial Optimization. In *Proc. of ICALP 2005*, Vol. 3580. Springer, 589–601.
[30] Darrell Whitley and Thomas Hanson. 1989. Optimizing Neural Networks Using Faster, More Accurate Genetic Search. In *Proc. of ICGA 1989*. Morgan Kaufmann, 391–396.