

Frequency Fitness Assignment

Thomas Weise, *Member, IEEE*, Mingxu Wan, *Member, IEEE*, Pu Wang, *Member, IEEE*, Ke Tang, *Member, IEEE*, Alexandre Devert, and Xin Yao, *Fellow, IEEE*

This is a preview version of paper [1] (see page 19 for the reference). It is posted here for your personal use and not for redistribution. The final publication and definite version is available from IEEE (who hold the copyright) at <http://www.ieee.org/>. See also <http://dx.doi.org/10.1109/TEVC.2013.2251885>.

Abstract—Metaheuristic optimization procedures such as Evolutionary Algorithms are usually driven by an objective function which rates the quality of a candidate solution. However, it is not clear in practice whether an objective function adequately rewards intermediate solutions on the path to the global optimum and it may exhibit deceptiveness, epistasis, neutrality, ruggedness, and a lack of causality. In this paper, we introduce the Frequency Fitness H , subject to minimization, that rates how often solutions with the same objective value have been discovered so far. The ideas behind this method are that *good solutions are hard to find* and that if an algorithm gets stuck at a local optimum, the frequency of the objective values of the surrounding solutions will increase over time, which will eventually allow it to leave that region again. We substitute a Frequency Fitness Assignment process (FFA) for the objective function into several different optimization algorithms. We conduct a comprehensive set of experiments: the synthesis of algorithms with Genetic Programming (GP), the solution of MAX-3SAT problems with Genetic Algorithms, classification with Memetic Genetic Programming, and numerical optimization with a (1+1) Evolution Strategy, in order to verify the utility of FFA. Given that they have no access to the original objective function at all, it is surprising that for some problems (e.g., the algorithm synthesis task) the FFA-based algorithm variants perform significantly better. However, this cannot be guaranteed for all tested problems. We thus also analyze scenarios where algorithms using FFA do not perform better or even worse than with the original objective functions.

Index Terms—Fitness Assignment, Frequency, Diversity, Combinatorial Optimization, Numerical Optimization, Genetic Programming

I. INTRODUCTION

Single-objective optimization is a process with the goal of finding good (ideally best, i.e., optimal) solutions x from within a space \mathbb{X} of possible solutions. An objective function f serves as quality measure guiding the search. Black-box metaheuristic approaches are methods that only require such an objective function and search operations to solve an optimization problem without any further insight into their structure. The most prominent family of these methods

are Evolutionary Algorithms, which have wide applications ranging from engineering, planning and scheduling, numerical optimization, to even program synthesis.

Most metaheuristic optimization methods start with a randomly generated set of candidate solutions. New points in the search space are derived by modifying or combining promising existing solutions. *Promising* here means having a better objective value than the other points visited so far, maybe combined with some considerations about diversity. The rationale is that:

- 1) In the ideal case, solutions which have better objective values should be closer to the global optimum or, at least, may have even better solutions in their vicinity.

The principle of tending to choose areas of the solution space for sampling where points with better objective values have previously been discovered is one of the most universally applied ideas in black-box optimization. Lehman and Stanley [2] argued that “increasing fitness does not always reveal the best path through the search space”. Building on their work, we believe that there is at least one other fundamental principle inherent to non-trivial optimization problems that can be exploited to solve them:

- 2) Good solutions (and hence, good objective values) are hard to find.

If we consider optimization as sampling of the search space, then we would expect the frequency of discoveries of good or optimal solutions to be low. We will show that the objective function f used by an optimization algorithm can be replaced by the frequency $H[d(f(x))]$ of previous discoveries of (discretized) objective values $f(x)$ during the optimization process. We will refer to that measure as *Frequency Fitness*. Here, a solution is considered the more promising the lower its Frequency Fitness is, i.e., H is subject to minimization.

This indirect fitness measure does *not correlate* with f and changes over time (whereas f usually is static). In the following, we first describe the features of this new fitness measure and show that it can be substituted in form of a *Frequency Fitness Assignment* (FFA) procedure into various optimization algorithms in a straightforward way (Section II). We then outline the works of Lehman and Stanley [2, 3] and Legg et al. [4–6], on which our approach is building, along with other related works in Section III. In Section IV we present a comprehensive experimental study showcasing situations where FFA leads to similar, better, and worse performance than using the original objective function f directly in different

T. Weise (tweise@ustc.edu.cn), M.X. Wan (mingxu@mail.ustc.edu.cn), P. Wang (wuyou308@mail.ustc.edu.cn), K. Tang (ketang@ustc.edu.cn), and A. Devert (marmakoide@yahoo.fr) are with the USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI), School of Computer Science and Technology, University of Science and Technology of China; Hefei, Anhui, China, 230027. X. Yao (x.yao@cs.bham.ac.uk) is with UBRI at USTC and CERCIA in the School of Computer Science, The University of Birmingham; Birmingham, U.K.

Manuscript received June 28, 2012.

Copyright (c) 2012 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

optimization algorithms and scenarios. This study comprises algorithm synthesis with Genetic Programming, combinatorial optimization with a Genetic Algorithm, classifier synthesis with Memetic Genetic Programming, and numerical optimization with an Evolution Strategy. We show that the Frequency Fitness can be utilized as an alternative optimization criterion, even though its nature is entirely differently from the original objective function f . We conclude our paper by summarizing our findings and discussing future work in Section V.

II. FREQUENCY FITNESS ASSIGNMENT (FFA)

A. Definition

In the context of this work, we aim at optimizing a single objective function $f : \mathbb{X} \mapsto \mathbb{R}^+$ that maps the solution space \mathbb{X} containing the candidate solutions $x \in \mathbb{X}$ to the positive real numbers \mathbb{R}^+ (including 0). Depending on the nature of the optimization problem, f may either be subject to minimization or maximization.

Especially in the context of Evolutionary Algorithms, objective functions and fitness are distinguished [7, 8]. Objective values are a direct measure of utility for single candidate solutions with a meaning also outside of the optimization process, e.g., to a human user. Fitness is a heuristic used only inside the optimization process, with a meaning depending on the current state, maybe incorporating arbitrary information sources such as density metrics or the rank inside a population. It is used to determine which candidate solutions are promising for further investigation.

Frequency Fitness is such a fitness measure. It is subject to minimization, i.e., the smaller the Frequency Fitness \mathbf{H} , the better. The Frequency Fitness \mathbf{H} of a candidate solution $x \in \mathbb{X}$ can be defined as $\mathbf{H}[\mathfrak{d}(f(x))]$, where

- 1) $f(x) \in \mathbb{R}^+$ is the objective value of x ,
- 2) $\mathfrak{d} : \mathbb{R}^+ \mapsto \mathbb{D}$ discretizes the real-valued objective function to a finite space $\mathbb{D} \subset \mathbb{N}_0$, and
- 3) $\mathbf{H} : \mathbb{D} \mapsto \mathbb{N}_0$ is the history of the search accumulated in a lookup table containing the absolute frequencies of discovery of the discretized values.

For each element in \mathbb{D} , \mathbf{H} holds the number of times it was discovered during the optimization process. The possible indices into table \mathbf{H} must be discrete and the total number of table entries must be well below the maximum number \widehat{FE} of function evaluations in order to allow these frequency values to become meaningful. FFA was originally designed for problems where the number of possible objective values is small (see the experiment Section IV-A), i.e., where these conditions are automatically met. Then, the table \mathbf{H} can be used to directly count the occurrences of the objective values $f(x)$ and \mathfrak{d} can formally be replaced by an identity mapping.

In problems with many objective values or even continuous codomains, \mathfrak{d} is used both for reducing the number of indices into \mathbf{H} (the size of \mathbb{D}) and for discretization. This approach was chosen in this work to investigate whether FFA can also work on problems for which it was not designed, such as continuous optimization (see Section IV-D).

B. Features & Assumption about FFA

In FFA, a candidate solution is considered more promising if it has an objective value less often encountered. Modifying such a candidate solution may lead to the discovery of elements with new features in terms of the objective values. Therefore, like Novelty Search [2], FFA drives the search forward to explore the objective space. Whereas traditional approaches follow a trend in the objective functions, our method follows a trend in the frequency landscape.

1) Features of FFA:

- a. Sampling a candidate solution x increases its frequency counter $\mathbf{H}[\mathfrak{d}(f(x))]$ as well as the Frequency Fitness values of all candidate solutions with the same objective value.
- b. The global optima of the objective function are also global optima of the frequency landscape – at least until the first discovery of a global optimum: The global optimum has the best possible objective value. Until such a value, i.e., a corresponding candidate solution, has been discovered, its sampling frequency and thus, its Frequency Fitness value is 0 (optimal).
- c. FFA both maximizes and minimizes the objective function at the same time, as both hard-to-find optima as well as hard-to-find bad solutions will be rewarded in the same way.
- d. The approach is general and can be integrated into arbitrary black-box methods.
- e. The approach has a low computational complexity, as the Frequency Fitness data can be gathered within a simple lookup table in $\mathbf{O}(n)$ per algorithm iteration where n is the number of candidate solutions to evaluate per iteration.

2) *Assumptions about FFA:* Although being uncorrelated to the objective function, we will show that the Frequency Fitness is an efficient and sufficient optimization criterion that can drive a search process to discover good solutions in many cases. In this paper, we want to investigate the following hypotheses as foundation of that assumption:

- i. With FFA, candidate solutions with a frequently occurring objective value will quickly degenerate in fitness (receive high \mathbf{H} values). This drives the search away from areas of low information.
- ii. In optimization problems where the amount of bad solutions is much higher than the amount of good solutions, this creates selection pressure towards solutions with good objective values.
- iii. As Frequency Fitness is uncorrelated to the objective function f , it does not depend on whether f rewards solutions which are “stepping stone” towards the global optimum properly. In other words, similar to the related approaches discussed in Sections III-B and III-C, it does not make the assumption that a better objective value means that the corresponding solution is closer to the global optimum.
- iv. This feature should also enable it to deal with deceptive and epistatic objective functions properly.

- v. Optimization based on the Frequency Fitness should therefore be able to yield good results in the presence of difficult problem features such as low causality, high ruggedness or deceptiveness of the objective function, neutrality, or epistasis [9].
- vi. On the other hand, Frequency Fitness-based methods should perform worse than methods relying on f directly in problems where these features are not present, where f is smooth, has a low total variation, where the Building Block Hypothesis [10] can be exploited, or where it is harder to find bad solutions than finding good solutions. FFA may then also hide existing meaningful causality.
- vii. Frequency Fitness provides a simple self-adaptive balance between exploration and exploitation. If an optimization process using FFA always exploits the solutions with best fitness, then this means that an optimum of the original objective function will be exploited as long as it is new. As soon as no improvement in the objective functions can be achieved anymore, the frequency of that optimum will just keep increasing which will drive the optimization process away from it and a new phase of exploration in the search space sets in.
- viii. A process of oscillation between exploitation and exploration, as described above, can also be considered as soft restarts *without* losing the aggregated knowledge. An area Q in the objective or search space may be exploited for some time. However, as a result, its Frequency Fitness will deteriorate which will drive the search away from it. If other areas have sufficiently been explored, their Frequency Fitness values may become higher and Q may become attractive again.
- ix. FFA jointly penalizes solutions that share the same objective values. This means when an optimum is exploited and its fitness degenerates in the process, the fitness of similar optima will degenerate as well. This can be a bad feature if two optima are immediately adjacent or are exploited at the same time. In other cases, it may be beneficial as it forces the search to sample worse solutions in between two exploitation cycles, i.e., to conduct exploration, which may enable it to escape from deceptive local optima.

C. Implementation

FFA can easily be inserted into any black-box optimization algorithm \mathcal{A} . For this purpose, first a frequency table \mathbf{H} needs to be allocated and filled with zeros. In all places of \mathcal{A} where new candidate solutions x are created, code needs to be inserted that calls the objective function f and increases $\mathbf{H}[\text{d}(f(x))]$ by one. All references to the objective function in \mathcal{A} need to be replaced with lookups to \mathbf{H} . It should be noted that the Frequency Fitness is dynamic even for static problems: the fitness of candidate solutions will deteriorate when similar solutions are discovered. This means that storing fitness values in variables in \mathcal{A} is *not permitted* and all such variables need to be replaced with frequency table lookups as well. \mathcal{A} now only accesses the Frequency Fitness $\mathbf{H}[\text{d}(f(x))]$ and

not the true objective values $f(x)$. Therefore it is necessary to revise stopping criteria and fitness-based self-adaptation methods (see Section IV-D), and to store the best solution encountered (according to f) in an additional variable. As the original optimization process can no longer assess the true quality of an individual, the FFA implementation itself must remember the best candidate solution ever encountered (in terms of objective value!) in a variable outside of the original optimization algorithm.

III. RELATED WORK

An optimization process has converged if it keeps producing candidate solutions from within the same limited area of the solution space and cannot explore other regions anymore. Premature convergence is convergence to a local optimum and can be caused by fitness landscape features such as multimodality, ruggedness, neutrality, and epistasis [9]. The idea to prevent the optimization process from converging to a single basin of attraction thus suggests itself, as it is usually not a priori known whether the best candidate solution discovered so far is a global optimum or not [8, 11].

A. Sharing and Niching

There exists much related work in population-based approaches such as EAs on this issue, including sharing, niching, and clearing-based methods [8, 12–14] as well as clustering [15] of the populations. In short, these methods have in common with FFA that they try to prevent premature convergence by driving the search away from areas of the search space that have frequently been sampled.

However, the differences are that (1) The main criterion for optimization under these techniques is still the objective function. It may be modified by a niche count or with a (death-)penalty for crowded areas. Still deceptive objective functions f remain deceptive and local optima of f are still local optima under sharing and niching methods. In FFA, the objective function is not visible to the optimization algorithm, the Frequency Fitness is uncorrelated to the objective values, and these problems may (gradually) disappear. (2) These methods only reflect the *current* state of the population, whereas FFA considers the whole history of the search process, i.e., utilizes more information to (hopefully better) guide the search. (3) Sharing, niching, and similar techniques are usually defined over the solution space and based on a distance metric. FFA is based on the equality of (discretized) objective values only, i.e., makes much fewer assumptions about the properties of solution space.

B. Fitness Uniform Selection and Deletion

The Fitness Uniform Selection Scheme (FUSS) [4–6] is a selection procedure for population-based algorithms (such as EAs), which works approximately as follows:

- i. Sort the population according to the objective value.
- ii. Obtain the minimum and maximum objective values (f_{min}, f_{max}) in the population
- iii. For each slot in the mating pool,

- a) pick a random number U uniformly distributed between f_{min} and f_{max} .
- b) from all the individuals which have an objective value closest to U , randomly pick one

Fitness Uniform Deletion Scheme (FUDES) [11, 16] works quite similarly. Instead of selecting individuals for reproduction, the FUSS idea is used to delete individuals when more space is required in the population of an EA, e.g., in order to integrate the offspring of a generation into the population.

There are several essential differences between FFA and FUSS/FUDES: (1) FFA can be integrated into arbitrary optimization algorithms, ranging from Hill Climbing to Evolutionary Algorithms or Memetic Algorithms. FUSS can only be used inside population-based algorithms. (2) FFA is a fitness assignment procedure which can be employed together with arbitrary selection schemes. FUSS is a selection scheme. (3) FUDES and FUSS aim to achieve a uniform distribution of objective values in the population of an Evolutionary Algorithm at each generation. FFA tries to achieve a uniform distribution of objective values throughout the entire optimization process. (4) FFA aims for an ideally never ending alternation between exploration and exploitation, allowing the search to temporarily fully concentrate on a local optimum while forcing it away from it after stagnation. In both FUSS and FUDES, individuals from all objective value levels are steadily generated so a collapse of the population is impossible, but also a fast exploitation of a new optimum (as possible with FFA) will probably not happen. In FFA, a collapse to one objective value level is possible, but may be amended by increased exploration after Frequency Fitness degeneration. (5) FFA incorporates the whole history of the search with the goal to prevent convergence. FUSS/FUDES only utilize the current state of the algorithm and maintain no explicit search history. (6) FFA requires discretization of the objective values to a finite set with a small cardinality with respect to the maximum number \widehat{FE} of function evaluations. FUSS does not have such a requirement.

However, there are also some similarities, such as: (a) Both FUSS/FUDES and FFA try to prevent the optimization process from converging. (b) Both take into account that sometimes, the path to the global optimum leads over a set of inferior intermediate steps. (c) Both aim for a uniform distribution of objective values, just in different time scales.

C. Novelty Search

Novelty Search [2, 3] is the approach most closely related to FFA. If Novelty Search is integrated into an EA, the concept is to abandon the objective function f . The reason is that, on one hand, in case of deceptive problems, f may be misleading and guide the search away from the optima. On the other hand, as mentioned in [2, 3] and Section II-B2, it is also not clear whether f adequately rewards stepping stones, i.e., intermediate solutions, between some initially chosen starting points and the global optimum.

Novelty Search applies a measure ρ of behavior novelty, called a novelty metric, which is different from the objective function. Instead of rewarding absolute performance, it

rewards divergence from previous behaviors. This novelty of a candidate solution is computed with respect to an archive of behaviors of past (novel) individuals x and the current population of the EA. It could be measured as the mean behavior difference to the k nearest neighbors in these sets.

For example, in a maze navigation domain [2, 3], where the goal is to find a controller steering a robot out of the maze, a straightforward *objective function* would measure the distance of the robot to the maze's exit after a fixed amount of time. The *behaviors*, however, could be the coordinates of the location of the robot at that time and the *novelty* measure would be the mean Euclidean distance to the k nearest endpoints reached by the other robot controllers in the archive or population.

The differences between Novelty Search and FFA are: (1) Novelty Search aims to make the concept of objective functions unnecessary. The objective function f is still the core of FFA – it is just hidden and indirectly presented to the optimization algorithm. Even though it may be ill-designed, f still represents the user's wishes, so its indirect use in FFA may be better than completely abandoning it. (2) Novelty Search preserves candidate solutions or their behaviors in an archive as reference set for computing the novelty metric. The fitness in FFA comes from the fixed-size frequency table which represents every single candidate solution ever evaluated (without the need to preserve any one of them). (3) Novelty Search tries to circumvent deceptive objective functions by omitting them. In FFA, local optima or deceptive basins of attraction are likely to be “filled” and degenerated in Frequency Fitness, hence driving the search away from them after some exploitation. (4) The application of Novelty Search is largely focused on topics such as the evolution of virtual creatures [17], walking behaviors [2], robot controllers [2, 3], etc., though not limited to these domains. FFA, from the start, is aimed at general optimization tasks that exhibit problematic aspects such as epistasis, neutrality, ruggedness, low causality, etc.

The similarities between Novelty Search and FFA are: (a) Novelty Search as well as FFA try to discover novel behaviors according to some metric. FFA applies an absolute metric (objective function) which is “relativized” via the history information. In Novelty Search, any possible set of metrics can be applied. (b) FFA and Novelty Search both aim at open-endedness of evolution. (c) Both methods employ relative measures of novelty.

In some works [17–19], Novelty Search is combined with other fitness measures to obtain better solutions. In FFA, this does not seem to be *a priori* necessary since it already (indirectly) utilizes the objective function. Nevertheless, it may be a good idea in order to better guide the search.

Although the authors of Novelty Search propose abandoning f , they also test using f as behavior definition [2]. Then, the novelty measure represents a mean distance ρ to k neighbors (or all solutions ever found) in the objective space, instead of the number \mathbf{H} of occurrences of the same objective values in FFA. Using such a mean distance would rely on the implicit assumption that differences of objective values are meaningful. FFA does not make this assumption as it only involves comparison for equality. Furthermore, if the distance

sum ρ is computed over only the k nearest neighbors of a solution, then *only these* can influence its fitness, whereas *all* identical individuals ever discovered will determine the fitness of a solution in FFA.

D. Tabu Search

Tabu Search [20, 21] extends Hill Climbing by declaring candidate solutions which have already been visited as tabu, i.e., preventing them from being sampled again. The simplest realization of this approach is to use a list which stores the last ν candidate solutions that have been tested, hence preventing cycles of at most length ν . More commonly, instead of storing the phenotypes directly, the search moves leading to their creation are stored.

It is clear that Tabu Search and FFA are similar in some points: (1) Both methods try to avoid producing similar solutions, i.e., try to avoid premature convergence. (2) Both methods keep a history of the search.

Some of the essential differences between Tabu Search and FFA are: (1) While Tabu Search stores whole solutions, solution features, or search operation applications, i.e., complex data structures, FFA uses a simple frequency lookup table which can be indexed with integers. (2) Tabu Search or other algorithms that use principles like the tabu list still essentially optimize based on the objective function f . If two solutions are not tabu, then they are compared based on their objective values. In FFA, this is not the case: Frequency Fitness is entirely uncorrelated with the objective function and the sole criterion utilized for driving the search. (3) Tabu Search is an optimization algorithm, FFA is a fitness assignment scheme that can be integrated into optimization algorithms.

IV. EXPERIMENTAL STUDIES

In this section, we will describe four experiments in which we integrate FFA into different optimization algorithms and solve optimization problems of different types. Our goal is to test the generality of FFA and to understand the behaviors of FFA on different problems. The first experiment concerns algorithm synthesis with Genetic Programming (Section IV-A), followed by an experiment on solving MAX-3SAT instances with a Genetic Algorithm in Section IV-B. In the third experiment (discussed in Section IV-C), we synthesize classifiers with MGP. We then investigate numerical optimization with a $(1 + 1)$ Evolution Strategy in Section IV-D. In each of these experiments, we compare the performance of FFA with the original algorithms and, where appropriate, other approaches. The basic features of these experiments and the settings used are described in Table I.

A. Genetic Programming for Algorithm Synthesis

In our previous works [22, 32], we used Genetic Programming (GP) to synthesize non-trivial, exact (i.e., non-approximative), deterministic algorithms which compute discrete results and use addressable memory. This task is very different from symbolic regression, where the solutions (formulas) are approximations and where GP excels [33]. In algorithm synthesis

problems, traditional program representations exhibit strong epistatic effects [32, 34]: It is not possible to modify one part of a program without affecting the behavior of the other parts [32]. As a consequence, the fitness landscapes of the problems under consideration are often very rugged and likely contain large neutral areas surrounding steep ascends or descends: Most possible programs are dysfunctional, and a program which solves a problem partly can be rendered dysfunctional with only a single modification due to the weak causality. If a program generates the right outputs in a larger subset of the training cases than another one, this does not necessarily mean that its structure is also more similar to a correct solution. Such features pose a problem for any black-box optimization method [2] and provide the ideal test bed for FFA, which is intended for exactly this scenario.

In [22], we compared the performance of five different loop representations on four different discrete algorithm synthesis problems. We briefly discuss them here and then perform experiments similar to [22] with FFA.

1) *Problem Description*: We try to synthesize discrete algorithms that use memory for four different problems, three of which require the use of a loop structure. For each problem, we use tc training cases t_i which are stored in the first memory cell(s) on program startup. After executing an evolved program x , we expect its result $x(t_i)$ to be stored in its last memory cell v_q , where $q = 2$ for the first three and 3 for the last problem.

As objective function f_{gp} , given in Equation 1, we use the number of hits, i.e., the number of solved training cases t_i where the result $x(t_i)$ computed by program x equals the expected result $\phi(t_i)$. f_{gp} is subject to maximization:

$$f_{gp}(x) = |\{(i \in 1..tc) \wedge (\phi(t_i) = x(t_i))\}| \quad (1)$$

a) *Polynomial Problem (poly)*: First, we propose a trivial polynomial problem $\phi_{poly}(t_i) = t_i^3 + t_i^2 + 2 * t_i$, designed to test whether the Genetic Programming system works correctly if FFA is applied. We use the $tc = 100$ training cases from $t_1 = 1$ to $t_{100} = 100$.

b) *Sum Problem (sum)*: The second problem is to find the sum $\phi_{sum}(t_i) = \sum_{j=1}^{t_i} j$ of the first t_i natural numbers. We omit the division operation from the instruction set, thus prohibiting Gauss' formula from being discovered and forcing GP to synthesize loops to solve this problem. We again use the training cases from 1 to 100.

c) *Factorial Problem (fact)*: The factorial problem, i.e., synthesizing a program which can compute $\phi_{fact}(t_i) = t_i!$ of a natural number t_i , also requires at least one loop. We use $tc = 12$ training cases $t_i \in 1 \dots 12$. Here, the last memory cell is always initialized with 1.

d) *GCD Problem (gcd)*: In the GCD problem, we try to find an algorithm which can compute the greatest common divisor $\phi_{gcd}(t_{i,1}, t_{i,2}) = \text{gcd}(t_{i,1}, t_{i,2})$. A training case $t_i = (t_{i,1}, t_{i,2})$ this time consists of the two natural numbers $t_{i,1}$ and $t_{i,2}$. We randomly create $tc = 100$ cases in a way so that a wide range of different result values are covered.

2) *Experimental Settings*: In order to synthesize algorithms for the above problems, we provide GP with the following operators: $+$, $-$, $*$, modulo, a node $a \circ b$ which executes two child instructions a and b sequentially, the $=$ operator

Table I

EXPERIMENT DESCRIPTION AND PARAMETER SETTINGS. In this table we describe the setups of the four experiments carried out in this research. A setup includes the type of problem to be solved, the basic algorithm chosen (FFA is then integrated into this algorithm), the configuration of the algorithm, as well as the number of function evaluations \widehat{FE} per run and the number of runs performed.

	Section IV-A	Section IV-B	Section IV-C	Section IV-D
Problem:	Algorithm Synthesis	Combinatorial	Classification	Function Optimization
Section:	Section IV-A	Section IV-B	Section IV-C	Section IV-D
Search Space:	trees, 5 different loops	bit strings $\{0, 1\}$	trees, SGDTs	continuous in $[-10, 10]$
Dimension:	max-depth 17	length 150	max-depth 17	dimension $m \in 1..5$
Instances:	4 from [22]	100 from [23, 24]	7 from [25]	8 from [26, 27]
Objective:	max training hits	max true clauses	max entropy f_E	min function value
Discretization d :	discrete by nature, i.e., $d(y) = y$		$d(y) = \lfloor 200y \rfloor$	$d(y) \sim \ln(y)$ (see Eq. 5)
Algorithm:	Genetic Programming [22]	Genetic Algorithm [10]	MGP [28]	Evolution Strategy [29–31]
Comparison:	GP \pm FFA	GA \pm FFA, FUSS, Rand. Walk	MGP \pm FFA	ES \pm FFA
Population Size n:	$n = 1000$	$n \in \{500, 5000, 50\,000\}$	$n = 100$	$\lambda = \mu = 1$
Selection:	Tournament	Tournament / FUSS	Tournament	$(\mu + \lambda)$
Selection parameter:	$t = 7$	$t \in 1..12$	$t = 4$	$\lambda = \mu = 1$
Initialization:	ramped-half-&-half (RHAH)	random uniform	RHAH, max-depth 4	random uniform
Mutation:	sub-tree replacement	single-bit flip	sub-tree replacement	normally distributed
Mutation rate:	0.1	0.5	0.1	1
Crossover:	sub-tree exchange	uniform	sub-tree exchange	—
Crossover rate:	0.9	0.5	0.9	0
Adaptation:	—	—	dynamic termination	$1/5^{th}$ Rule
parameters:	—	—	—	$a \in \{0.5, 0.85, 0.975\}$, $L \in \{10, 100\}$
Max F. Evals. \widehat{FE}:	$\widehat{FE} = 100n$	$\widehat{FE} = 100n$	$\widehat{FE} \leq 150n$ (self-adaptive)	$\widehat{FE} \in 10^{2..7}$
Runs:	100 per instance	1 per instance	20 \times (5-fold cross-valid.)	100 per instance/setting

assigning the value of an expression to a memory cell, the two constants 0 and 1, as well as one terminal v_i for each of the q memory cells. To this basic instruction set, we add one loop instruction for each experiment: the counter loop CL executing its loop body (sub-tree 2) as often as determined by one expression (sub-tree 1), the while loop WL executing a loop body (sub-tree 2) until its condition (sub-tree 1) becomes 0, and the memory loop ML that decreases a variable by one for each iteration of its sub-tree until it reaches 0. Two indirect loop structures are tested as well: In the CA method, “=” is replaced by a conditional assignment and the whole program is executed repetitively until no variable changes anymore. The implicit loop IL executes its body until no variable changes anymore. More explanations regarding these representations are given in [22] and further experimental settings can be found in Table I.

3) *Experimental Results:* In Figure 1, we display the box plots for the results of the five different program representations on the four algorithm synthesis problems. In each diagram, we put the boxes for the original Genetic Programming approach that directly uses the objective values (GP-DIR) next to the results achieved with the FFA-based method (GP-FFA) for easier comparison. At first glance, we can see that the majority of the runs for one setting usually yield very similar results in terms of objective values, resulting in the collapse of the corresponding box, often leaving only 5% and 95% quantiles and extrema. This is typical for the tested kind of hard problems which are highly epistatic and neutral and where many local optima actually are “dead ends”.

A closer look reveals that GP-FFA, for most program

representations, actually outperforms the GP-DIR settings and often achieves a wider spread of different results.

In order to better visualize the behavior of FFA in this context, we present the results of two-tailed Mann-Whitney U test at significance level 0.02 in Figure 2. This figure displays four graphs, one for each experiment. Each graph holds a box for each representation for both, GP-FFA and GP-DIR. An arrow from box a to box b means that setting b outperforms setting a significantly. Arrows (and thus, performance relationships) are transitive [35]. From these figures we can see that

- i. for each experiment, there is always one GP-FFA setting losing to no other setting,
- ii. there is only one experiment (sum) with a GP-DIR setting not beaten by an GP-FFA setting and
- iii. there is only one such setting,
- iv. no GP-FFA setting ever loses against the corresponding GP-DIR setting with the same program representation, and
- v. in about half of the cases the GP-FFA settings outperform the corresponding GP-DIR settings significantly, in the rest there is no significant difference.

4) *Behavior of FFA:* We now take a closer look on how single runs under FFA perform in Figure 3. We repeat the gcd experiment for 500 generations in Figure 3.a for the memory loop structure ML. As there are 100 training cases, f_{gp} here ranges from 0 to 100. In the left part of the figure, we plot how often programs achieving each of these values were sampled *per generation*. This resembles the absolute change $\Delta \mathbf{H}[f_{gp}]$ of the Frequency Fitness $\mathbf{H}[f_{gp}]$, which we plot on the right

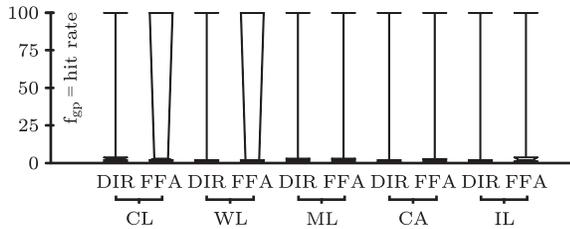


Fig. 1.a: Performance on poly

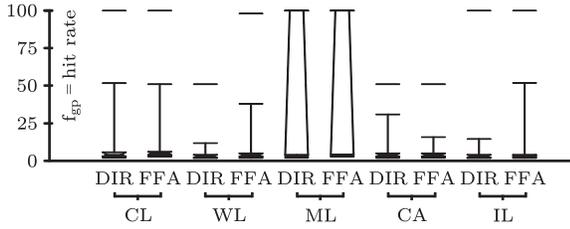


Fig. 1.b: Performance on sum

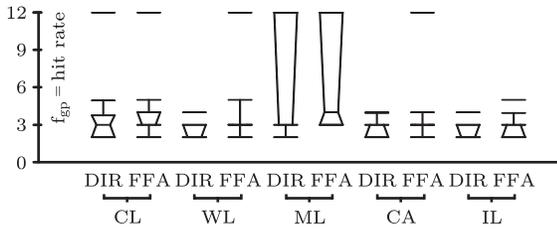


Fig. 1.c: Performance on fact

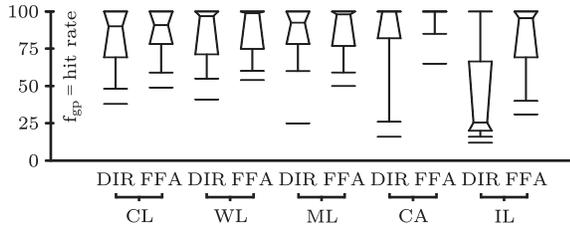


Fig. 1.d: Performance on gcd

Fig. 1. Box plots showing the experimental outcomes for GP-DIR and GP-FFA of the four algorithm synthesis experiments over 100 runs in terms of the hit rate f_{gp} . GP-FFA tends to have a higher hit rate than GP-DIR.

hand side.¹

As can be seen, candidate solutions with $f_{gp} = 0$ and $f_{gp} \in \{5, 17\}$ occur particularly often, as entirely dysfunctional programs or programs that always guess 1 or 2 to be the greatest common divisor of two random numbers are easy to generate and often the result of mutation or crossover. For the other f_{gp} values, however, we can observe that the related Frequency Fitnesses $\mathbf{H}[f_{gp}]$ become nearly identical once corresponding programs have been found. This is what distinguishes FFA from FUSS: While the former tries to achieve uniform sampling of objective values throughout the complete search process, the latter tries to achieve this in the current population. FFA thus permits temporary over-sampling of a given category of programs, which is manifested by steep spikes in the left part of the figures ($\Delta\mathbf{H}[f_{gp}]$), i.e., in exploiting the best currently known candidate solutions (as

¹The discretizer d has been omitted here, as it is the identity mapping.

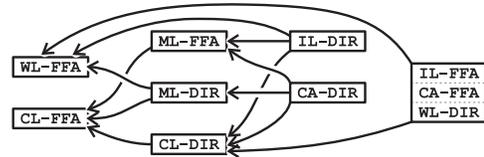


Fig. 2.a: Statistical comparison on poly.

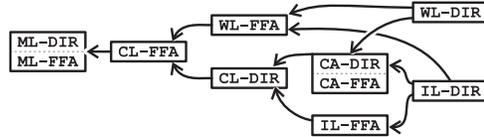


Fig. 2.b: Statistical comparison on sum.

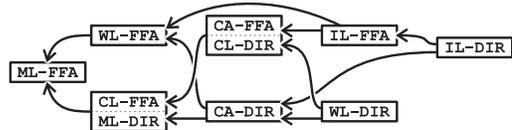


Fig. 2.c: Statistical comparison on fact.

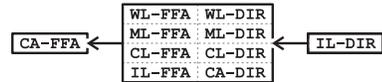


Fig. 2.d: Statistical comparison on gcd.

Fig. 2. Comparisons of the different combinations of GP-FFA and GP-DIR with the five loop structures in terms of f_{gp} . The test result graphs representing partial orders of f_{gp} according to two-tailed Mann-Whitney U test at significance level 0.02 over 100 runs. A (transitive) arrow from approach a to b means that b is significantly better than a . The GP-FFA methods tend to be better than GP-DIR.

these here also are the rarest ones).

In the left part of the figure we can also see one oscillation cycle: After about 165 generations, solutions with $196 < f < 99$ have been discovered and are sampled heavily – but only until approximately generation 200. At around generation 250, even solutions with $f \geq 93$ are not sampled anymore. Their fitness has deteriorated and GP now focuses on exploration. At around generation 310, however, the fitness of the weaker solutions has increased enough and the search can re-discover and exploit solutions of better objective values again.

In Figure 3.b we apply GP-FFA with ML to the factorial problem *fact* with $tc = 12$ training cases. Similar trends can be observed: programs computing $z = z!$ for $z \in 1 \dots 2$ or being completely dysfunctional are sampled from the beginning. Their Frequency Fitness increases quickly. As soon as programs which are able to solve more training cases are discovered, they are sampled more often to achieve an overall balanced frequency table \mathbf{H} where possible.

The balanced Frequency Fitness is what enables GP-FFA to escape local optima that trap GP-DIR. The afore-mentioned trivial programs receive the same fitness and after the local optima that can solve three and four cases have been exploited, they are treated equally, which prevents premature convergence and finally leads to the discovery of new traits and the solution of the problem near the end of the run.

FFA does not a priori prefer a partial solution which is correct on 20% of the training cases over one that is correct

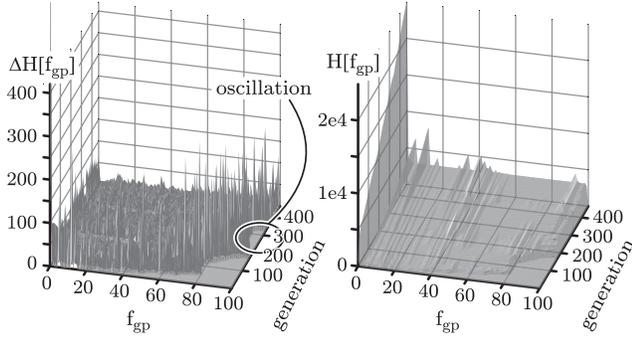


Fig. 3.a: One run of GP-FFA with ML on gcd.

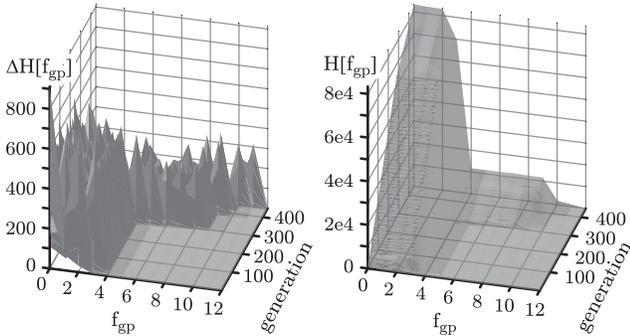


Fig. 3.b: One run of GP-FFA with ML on fact.

Fig. 3. Visualization of two example runs of GP-FFA in terms of frequency $\mathbf{H}[f_{gp}]$ with which a certain hit rate f_{gp} is sampled accumulated over the generations and for a given generation ($\Delta\mathbf{H}[f_{gp}]$).

on 10% only. In the presented problems, this is good, as it is not clear which one can be modified to a globally optimal solution with fewer steps. Together, these effects result in good FFA performance.

B. Combinatorial Optimization: MAX-3SAT

In satisfiability problems (SATs), a formula B in Boolean logic is defined over v Boolean variables x_1, x_2, \dots, x_v , each of which can be either `true` or `false`. The goal is to find a setting for these variables, i.e., an element of the solution space $\mathbb{X} = \{0, 1\}^v$ so that B becomes `true`. According to [36], MAX-SAT problems are deceptive with many neutral regions. For a Genetic Algorithm, it is therefore hard to find the global optimum.

1) *Problem Description*: In CNF 3-SAT problems, the formula B consists of c clauses C_1, C_2, \dots, C_c :

- a. Each clause consists of three literals.
- b. A literal can either be a variable (e.g., x_5) or a negated variable (e.g., $\neg x_5$).
- c. In a clause, the literals are combined with logical *or* (\vee).
- d. In the formula B , all c clauses are combined with logical *and* (\wedge).

As such a formula B becomes `true` if and only if all clauses are `true`, the problem can be rephrased from its Boolean nature (B is `true` or `false`) to a combinatorial optimization task with the goal of maximizing the number of clauses that

are `true`. The objective function f_{3s} for such a MAX-3SAT problem is defined in Equation 2.

$$f_{3s}(x) = \sum_{i=1}^c \begin{cases} 1 & \text{if } C_i(x) \text{ is true} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

As benchmark instances of this problem, we chose the set `uf150-645` from SATLIB [23, 24], which consists of 100 instances. Each instance has $c = 645$ clauses defined over $v = 150$ variables and at least one correct solution.

2) *Experimental Settings*: Because of the binary nature of the search space, we chose a Genetic Algorithm (GA) as optimization method in this experiment. We used three population sizes $n \in \{500, 5000, 50000\}$ and twelve tournament sizes $t \in 1 \dots 12$. Notice that tournament size $t = 1$ transforms the GAs into parallel random walks. Each setting was executed for exactly 100 generations once for each of the 100 benchmark instances. We compared the original GA without any modifications (GA-DIR), a GA using our Frequency Fitness assignment (GA-FFA), and a GA with the related approach FUSS as selection scheme (GA-FUSS).

3) *Experimental Results*: For the MAX-3SAT problem, the key statistic is the number of cases that an approach can solve. From [36], we know that GAs are weak in solving these problems. The best six GA-DIR setups (all with a population size $n = 50000$) can solve at most 5 of 100 cases, which is more likely the result of random mutations than targeted search. The best GA-FFA settings are even worse than that and can only solve a single case each. GA-FUSS did not discover a single solution. We will therefore focus on the approximation quality provided by the approaches.

In Figure 4, it can be seen that the original Genetic Algorithm, GA-DIR, performs best when solving the MAX-3SAT problem. Low tournament sizes t improve its performance for the smaller populations. For larger populations, the spread of solutions gets smaller and the influence of t decreases. All GA-FFA settings lose in a two-tailed Mann-Whitney U test at significance level 0.02 against the corresponding GA-DIR setup. As also discussed in [11], the GA with FUSS (GA-FUSS) here is worse than GA-FFA but still clearly better than parallel random walks ($t = 1$).

The results delivered by the GA with FFA (GA-FFA) for population size $n = 500$ tend to have better median, upper quartile, 95% quartile, and maximum objective value than FUSS but cannot reach the quality of GA-DIR. They have a larger spread of results. For tournament size $t = 2$, the performance is worse than GA-FUSS. For higher population sizes, the performance of GA-FFA becomes more similar to the original GA. Their lower quartile of the objective values is often better than the median results of GA-FUSS. The tournament size also loses its influence on the performance, except for $t = 2$ which remains a dysfunctional setting.

4) *Behavior of FFA*: We now compare the change in the population structures in the GAs during the course of the evolution in Figure 5. Each of the eight small plots display the best, worst, median, lower and upper quartile objective values in the population at a specific generation of randomly chosen

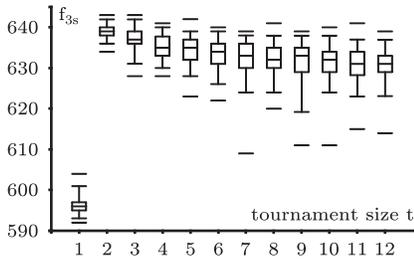


Fig. 4.a: GA-DIR ($n = 500$).

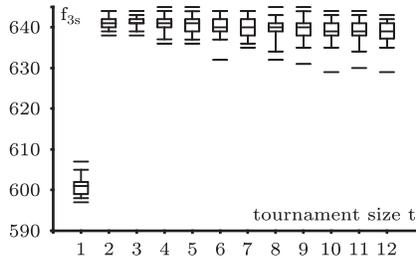


Fig. 4.b: GA-DIR ($n = 5000$).

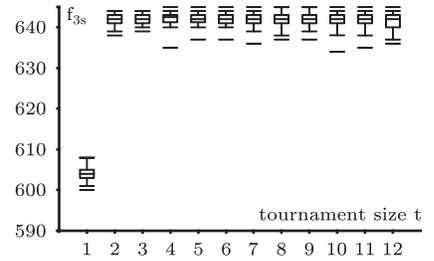


Fig. 4.c: GA-DIR ($n = 50000$).

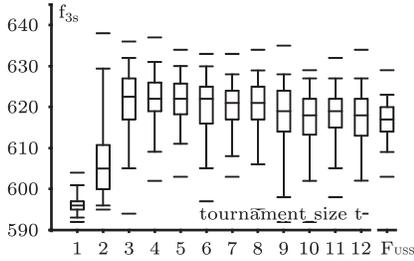


Fig. 4.d: GA-FFA, GA-FUSS ($n = 500$).

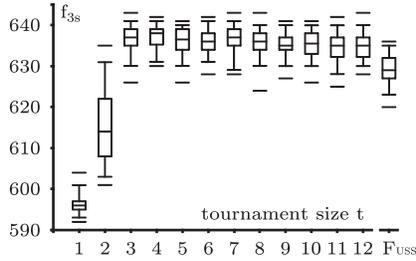


Fig. 4.e: GA-FFA, GA-FUSS ($n = 5000$).

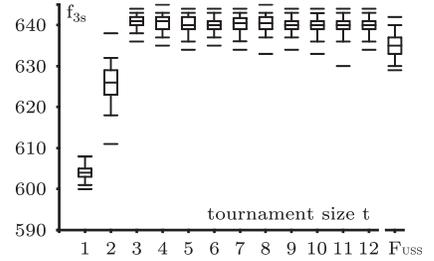


Fig. 4.f: GA-FFA, GA-FUSS ($n = 50000$).

Fig. 4. Box plots of the experimental outcomes of the MAX-3SAT experiments: Statistics for different tournament t and population sizes n are given in terms of f_{3s} for GA-DIR, GA-FFA, and GA-FUSS. GA-DIR performs better than GA-FFA, which in turn outperforms GA-FUSS.

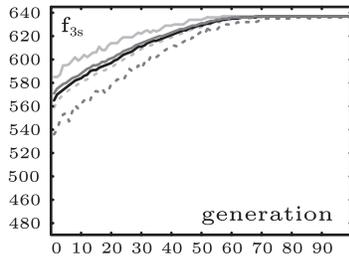


Fig. 5.a: GA-DIR ($n = 500$, $t = 2$, $\overline{FE} = 100$).

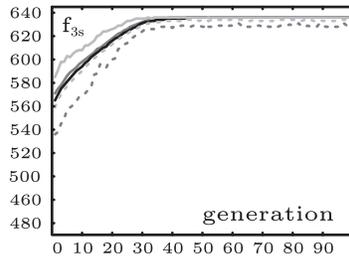


Fig. 5.b: GA-DIR ($n = 500$, $t = 3$, $\overline{FE} = 100$).

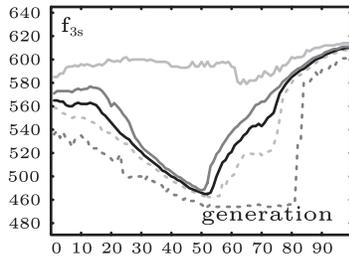


Fig. 5.c: GA-FFA ($n = 500$, $t = 2$, $\overline{FE} = 100n$).

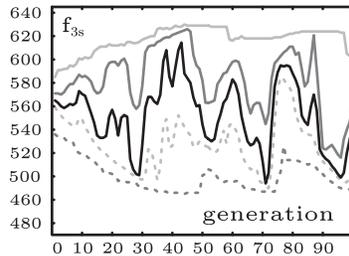


Fig. 5.d: GA-FFA ($n = 500$, $t = 3$, $\overline{FE} = 100n$).

best — 75% q. — median —
worst - - - 25% q. - - -

Each subfigure displays the population structure of one typical run for a specific algorithm configuration. For each generation of that run, the diagrams display the objective values of the best, 75% quantile, median, 25% quantile, and worst individuals in the population.

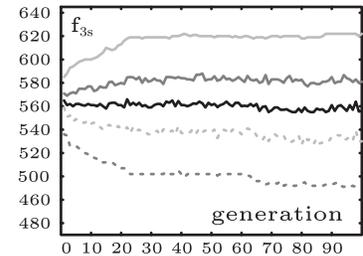


Fig. 5.e: GA-FUSS ($n = 500$, $\overline{FE} = 100n$).

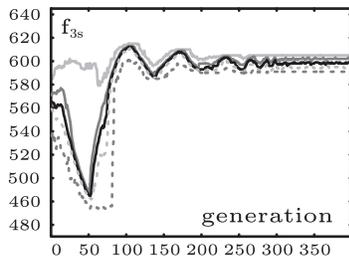


Fig. 5.f: GA-FFA ($n = 500$, $t = 2$, $\overline{FE} = 400n$).

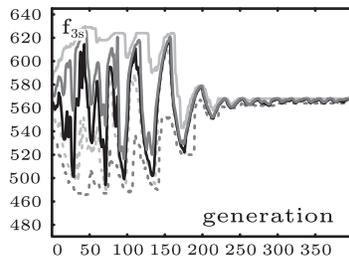


Fig. 5.g: GA-FFA ($n = 500$, $t = 3$, $\overline{FE} = 400n$).

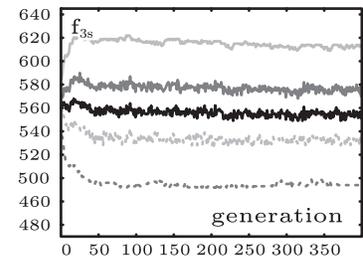


Fig. 5.h: GA-FUSS ($n = 500$, $\overline{FE} = 400n$).

Fig. 5. Population structures of typical runs of GA-FFA, GA-DIR, and GA-FUSS for different configurations in the MAX-3SAT experiments in terms of f_{3s} . The population of GA-DIR converges quickly, the population of GA-FFA oscillate between exploration and exploitation, whereas GA-FUSS retains a uniform distribution of good and bad solutions all the time.

but typical experimental runs. In the first two plots (Figure 5.a and 5.b), for instance, we can see that the population of the GA-DIR converges to good quality solutions after 50 generations for tournament size $t = 2$ and after about 30 for $t = 3$ for a population of $n = 500$ individuals.

GA-FFA for tournament size $t = 2$ and a population of $n = 500$ individuals first searches in the wrong direction (Figure 5.c), which happened with about 50% probability, as both directions are initially “novel”. However, after about 50 generations, the search turns around. Now reaching good solutions, the population makes a few small oscillations before converging. In the third row of Figure 5, we repeated the exact runs of the second row, this time granting 400 generations, to get a better visual impression, see e.g., Figure 5.d.

For a tournament size of $t = 3$, the oscillation is much more visible. The median population moves back and forth between good and bad solutions multiple times before converging at bad solutions after more than 200 generations. This convergence happens much later as in GA-DIR. Furthermore, as long as good solutions have been discovered (and are remembered outside the search process) at some point in time, the exact objective value around which the convergence finally occurs plays no role.

The graphs for one run of GA-FUSS (Figure 5.e and 5.h) show a different behavior. GA-FUSS manages to retain a wide spread of different objective values in the population all the time and never actually converges. The reason for the better results obtained by GA-FFA may be the soft restarts proposed in the introduction. Whereas the GA-FUSS tries to maintain a uniform objective value distribution and therefore only puts very little pressure towards a specific direction. In FFA, there always exists a higher selection pressure, but the characteristics favored by selection change.

When analyzing the behavior of FFA in the MAX-3SAT domain, we should remember that in the algorithm synthesis experiments, it was extremely easy to discover the global infimum, i.e., programs which do not work at all and have a hit rate f_{gp} of zero. Most of the selection pressure resulting from the Frequency Fitness thus was targeted for improvements and FFA outperformed the GP-DIR. Finding the global infimum of a MAX-3SAT problem, however, is as hard as finding the global optimum.

Even more, randomly sampled solutions often have objective values of more than 550. This leaves only about 100 different objective values for improvement, but 550 that could exist in the direction of worse solutions. The selection force of GP-FFA is directed mainly towards new and better solutions, whereas it can, at best, oscillate between good and bad solutions in GA-FFA.

As a result, GA-FFA is not as effective as the GA-DIR for the MAX-3SAT problem. Still, it leads to results which are better than those of the related FUSS approach. Furthermore, the spread of the selection pressure towards the wrong direction could probably be mitigated by introducing a function \mathcal{d} that could, for instance, put all solutions that are worse than the mean solutions of the first generation into the same bucket of \mathbf{H} . A comprehensive study of such adaptations will be part of our future work.

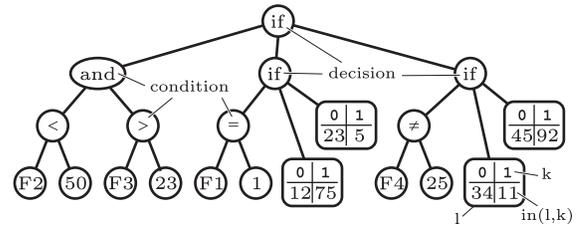


Fig. 6. Example for Statistical Genetic Decision Trees (SGDTs) for binary classification. Similar to classical decision trees, the inner nodes of SGDTs contain if-then-else instructions each holding a condition and two alternatives. The leaf nodes below alternatives hold frequencies measuring how often elements of the two classes arrived in them during the training phase. In the test/application phase, these frequencies determine which class the leaf node assigns to a sample arriving at it. More information is given in [28].

C. Classification: MGP

We integrated FFA into the Memetic Genetic Programming (MGP) system developed by Wang et al. [28]. Evolving classifiers with GP exhibits less epistasis, ruggedness, and neutrality than the synthesis of deterministic algorithms with discrete, non-approximative results but still is a computational hard problem. The MGP system additionally has another feature that makes it interesting as an application area for FFA: it is a highly specialized, fine-tuned tool with perfectly matched components. We will now outline the original MGP system briefly (while pointing to [28] for more details).

MGP [28] is used to synthesize tree-based classifiers. Usually, a decision tree contains a set of decision nodes that are traversed starting from its root for each data sample to be classified until a leaf node is reached that contains a class assignment. MGP, however, builds *statistical genetic decision trees* (SGDTs) where the leaf nodes l instead hold an estimated probability $p(l, k)$ for each class k [28] as illustrated in Figure 6 and Equation 3. These probabilities are learned during the training phase.

$$p(l, k) = \frac{in(l, k)}{in(l, 0) + in(l, 1)} \quad (3)$$

Besides the off-the-shelf mutation and crossover operators for trees in GP, MGP also utilizes two local search (i.e., memetic) operations which have specifically been designed to improve the classifier performance. MGP focuses on generating classifiers with maximum AUC (Area Under the Receiver Operating Characteristics Curve), a more discriminating measure than the accuracy metric simply counting the number of falsely classified data samples [28].

1) *Problem Description*: In our experiments, we use seven datasets from the UC Irvine Machine Learning Repository [25] which are described in detail in Table II. These datasets represent binary classification problems that exhibit different numbers and types of features (ranging from six to 100) and scales (ranging from 270 to 4600 instances).

As the AUC is too computationally expensive to be used as an objective function directly, the function f_E from [28] that has a strong positive relationship with AUC is optimized instead. For each of the class leaf nodes l of a SGDT x , we first compute the relative frequencies $p(l, k)$ with which training

Table II

DATA SETS [25] USED FOR THE CLASSIFICATION EXPERIMENTS. For each of the 7 data sets, we provide the shortcut and name, the number of instances, the class distribution (0/1), a short summary on the features, and a reference to the corresponding literature.

	Name	Instances (Distribution)		Features	Reference
bands	Cylinder Bands	512	(312/200)	39: 19 nominal, 20 continuous	[37]
crx	Credit Approval	690	(307/383)	15: 9 nominal, 6 continuous, some missing	[38]
heart	Statlog (Heart)	270	(150/120)	13: 3 binary, 1 ordered, 3 nominal, 6 continuous	[39]
hv	Hill-Valley (both training sets)	1212	(612/600)	100 continuous	[40]
mammo	Mammographic Mass	961	(516/445)	6 integer/nominal, some missing	[41]
monks	MONK's Problem #1 (test sets)	432	(216/216)	6 integer (+ 1 id)	[42]
spam	Spambase	4601	(2788/1813)	48 continuous	[43]

cases belonging to the different classes k arrived at that node ($in(l, k)$, Equation 3). As all problems given in Table II are binary classification tasks, k can either be 0 or 1.

$$f_E(x) = \frac{\sum_{\forall l \in \text{leaves of } x} \left(1 + \sum_{k=0}^1 p(l, k) \log_2 p(l, k)\right) \left(\sum_{k=0}^1 in(l, k) - 1\right)}{tc - |\text{leaves of } x|} \quad (4)$$

The objective function f_E , given in Equation 4, uses this information to compute the entropy. f_E will be 0 for random classifiers and 1 for classifiers with maximum AUC [28].

2) *Experimental Settings:* At first, we run the experiments with the original algorithm, here denoted as MGP-DIR. We then directly introduce FFA into the MGP source code, replacing all accesses to the objective function with accesses to the map \mathbf{H} , yielding algorithm MGP-FFA. In the experiments, we apply the dynamic stopping criterion of the latest MGP release that terminates the procedure if, after more than 50 generations, no further improvement in terms of fitness could be achieved for ten generations and the AUC on the training data is 1. All further algorithm settings are the same as in [28] and described in Table I.

3) *Experimental Results:* In Figure 7, we sketch the Box plots of the classification experiment for the final achieved results in terms of AUC on the training data (Figure 7.a) and test data (Figure 7.b).

There is a small advantage of MGP-DIR over MGP-FFA on the training data and MGP-FFA has a much wider spread of obtained solution quality – for the worse, that is. The medians and quartiles on the test data, however, are often about the same. In the *monks* problem, the 25% quantile of MGP-FFA already reaches an AUC of 1, whereas MGP-DIR's 25% quantile and median are 0.90 and 0.99, respectively. On the test data, MGP-DIR significantly outperformed MGP-FFA in *bands* and *heart*, MGP-FFA delivers significantly better results in *monks*, while there is no significant difference in the remaining datasets, according to two-tailed Mann-Whitney U test at significance level 0.02.

Although the performance of MGP-FFA on the test data is not much different from MGP-DIR, it produces more diverse classifiers with less overfitting – a wider spread of training AUC maps to basically the same range of test AUC. This makes MGP-FFA an interesting option for synthesizing ensemble classifiers where diversity is favorable [44]. In summary, FFA can indeed be used as a replacement for a direct fitness measure – even in a highly specialized and complex system such as MGP.

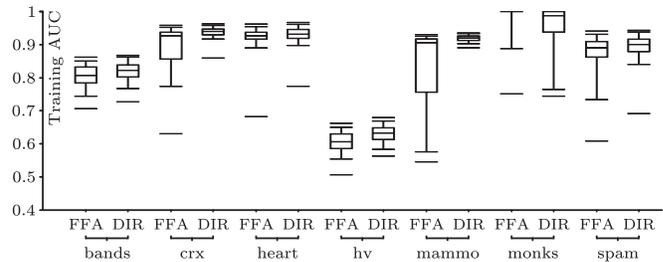


Fig. 7.a: AUC performance on the training data.

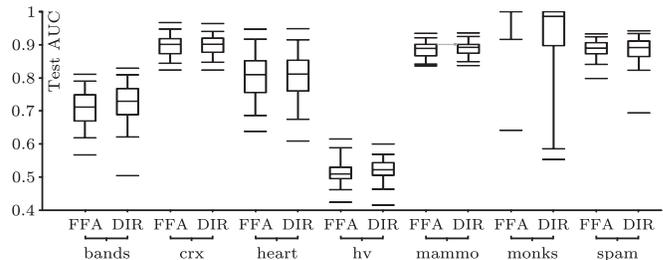


Fig. 7.b: AUC performance on the test data.

Fig. 7. Box plots of the AUC of MGP-FFA and MGP-DIR over 100 runs ($\equiv 20 \times$ five-fold cross-validation). Whereas MGP-DIR tends to perform better on the training data (Figure 7.a, FFA seemingly decreased the overfitting and improved the diversity of the evolved classifiers, so that the performance on the test data (Figure 7.b) is very similar.

D. Numerical Optimization: (1+1) ES

FFA intuitively seems to only be suitable for problems with discrete objective values and indeed, all the previous experiments were of this type. In this section we present a straightforward extension to continuous optimization. We assume the minimization of continuous, m -dimensional functions $f_i : \mathbb{R}^m \mapsto [0, +\infty)$ on a computer with limited precision.²

If the computations are done with IEEE 754 double precision floating point numbers [45], the range of positive finite representable values spans from roughly $4.9 \cdot 10^{-324}$ to $1.8 \cdot 10^{308}$. As $\lfloor \ln(4.9 \cdot 10^{-324}) \rfloor = -745$, one possible way to discretize such values into a tractable interval would be to apply the mapping d given in Equation 5.

$$d(y) = \begin{cases} 746 + \text{round}(\ln(y)) & \text{if } y > 0 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

As $\lfloor \ln(1.8 \cdot 10^{308}) \rfloor = 710$, all possible outputs of f can

²Strictly speaking, this makes the problems discrete again. However, such problems are commonly regarded as continuous.

be discretized into the interval $0 \dots 1456$, i.e., the Frequency Fitness mapping can be maintained as a lookup table with at most 1457 entries.

1) *Experimental Settings*: To evaluate whether FFA can actually be applied in such a crude way to numerical optimization, we chose a well-studied algorithm as basis for our experiments: The $(1 + 1)$ Evolution Strategy (ES) with self-adaptation via the $1/5^{th}$ rule [29, 30]. This algorithm keeps the best candidate solution x^* discovered so far and in each generation generates one new vector by adding normally-distributed random numbers (mean 0, standard deviation σ) to the elements of x^* . The outcome is accepted if it has better or the same fitness as x^* .

The step width σ is initially set to $\sigma_0 = 1$ and adapted every L function evaluations (FE s), here with $L \in \{10, 100\}$. The adaptation sets $\sigma = \sigma * a$ if less than 20% = $1/5$ of the mutation are *successful*, i.e., leading to *fitness* improvements during the last L steps and to $\sigma = \sigma/a$ otherwise, where $a \in \{0.5, 0.85, 0.975\}$. We chose the $(1 + 1)$ ES because of its simplicity, the mature state of the research on it, and for maximum replicability of experiments. We will refer to the original form of this algorithm as ES-DIR.

We apply ES-DIR to eleven benchmark functions (see Table III) for dimensions³ m ranging from 1 to 5 in the interval $[-10, 10]$. We perform 100 runs for each setting and collect the best results obtained per run after $\widehat{FE} = 10^t$ with $t \in 2 \dots 7$ FE s.

We then substitute FFA for the original objective function into the algorithm and repeat the experiments. We will call the modified algorithm ES-FFA. Notice that the substitution of FFA for the objective function will also affect the definition of *success* which now means generating a candidate solution with a less-often discovered scale of fitness compared to the currently best solution.

2) *Experimental Results*: In this experiment, the results are quite clear. Similar to the MAX-3SAT experiments, not a single ES-FFA configuration can strictly outperform the corresponding ES-DIR setting on any function. However, some settings come quite close: For $m = 5$ dimensions at $\widehat{FE} = 10^5, 10^6, \text{ or } 10^7$ with $L = 10$ and $a = 0.975$, ES-FFA manages to provide statistically significantly better results than ES-DIR on three functions while being worse at four (according to two-tailed Mann-Whitney U test at significance level 0.02). The best ES-DIR setting with $L = 100$ and $a = 0.5$ outperforms all other settings 1944 times, loses 361 times, and is not significantly different 1259 times, over all functions, dimensions, and \widehat{FE} settings. The best ES-FFA setting ($L = 10$ and $a = 0.975$), in turn outperforms the other settings 1568 times, is not different 1094 times, and loses 902 times – thus being better than one third of the ES-DIR settings.

We here summarize the results of this experiment from the perspective of the function subject to minimization (Table IV) and the dimensionality m (Table V). Each of these tables compares the ES-FFA settings with the corresponding ES-DIR setting with the same configuration (L, a) and is divided

into three parts: statistical comparison, comparison in terms of the best discovered solution over *all* runs of all settings, and comparison in terms of solution variance.

From Table IV, we can see that using FFA instead of the direct objective values leads to at least as many or more statistical significant improvements than losses over all m and \widehat{FE} settings per function in roughly 18% of the configurations. In about one fourth of the settings, FFA can discover more better *best* results over the 100 runs. From the third part of this table, it becomes clear that FFA basically always leads to a larger solution variance. This may be the reason for its worse performance compared to the plain ES-DIR, as the variance in function optimization tends to be skewed towards worse solutions: Since the minimum value of the functions is zero, the only reason for a larger variance can be worse results. This is likely caused by the fact that FFA rewards solutions which have particularly bad fitness as well as those with very good objective values.

Table IV gives information about which problem is suitable for FFA. ES-FFA seems to perform comparably well on the two neutral problems f_{10} and f_{11} (Equations 15 and 16) that have been designed with the Genetic Programming experiments in mind. They have large areas with the same objective values. f_{10} is discrete whereas f_{11} only has a non-zero gradient in a small area of the search space. During 100 runs, ES-FFA also discovered better best results on Levy's and Griewank's function more often than ES-DIR.

Table V shows that with rising dimensionality m of the objective functions, there is a clear trend towards more statistical losses of ES-FFA to ES-DIR. The reason for this maybe is that the span width of different possible objective values for the points within $[-10, 10]^m$ increases with m and so does the number of entries in the frequency table that are used. This probably disperses the selection pressure.

3) *Behavior of FFA*: We now show how FFA works within ES-FFA for $L = 10$ and $a = 0.975$.

a) *Sphere Function*: As the first example we chose the Sphere function (f_8 in Table III) because of its simplicity. In Figure 8, we illustrate the Sphere function itself (Figure 8.a). This function is unimodal, smooth, and symmetric around 0, so we can also plot one half of it by using a logarithmic scale for x (right part of the subfigure).

In the second plot (Figure 8.b), the trace of a typical run of ES-FFA on f_8 is shown. On the x-axis, we plot the function evaluations FE . The y-axis is logarithmically scaled. We show the distance of the point x sampled by the ES-FFA at a specific FE to the global optimum 0, i.e., $|x|$. It can be seen that this distance tends to get smaller in scale. The $(1 + 1)$ -ES always holds the best point x^* found so far, and for ES-FFA, best is defined in terms of the Frequency Fitness. With the thick black line, we illustrate $|x^*|$. As it can be seen, this value also tends to get logarithmically smaller, but not monotonically – FFA here sometimes prefers to make a step back. We also plot $f_8(x^*)$ which (due to $f_8(x) = x^2$) has the same characteristics. The $1/5^{th}$ rule leads to a monotonous decrease in scale of σ .

The shape of this run thus is very similar to what we get from ES-DIR. We now investigate how the frequency

³Rosenbrock's function only makes sense for dimensions $m \geq 2$.

Table III

THE ELEVEN NUMERICAL BENCHMARK FUNCTIONS. We define all benchmark functions used in the ES-FFA/ES-DIR experiments with their well-known name, mathematical definition, and reference to literature (if any). The two neutral problems in Equations 15 and 16 have been designed for this study with the Genetic Programming experiment (Section IV-A) in mind.

Ackley's Function	$f_1(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{m} \sum_{i=1}^m x_i^2}\right) - \exp\left(\frac{1}{m} \sum_{i=1}^m \cos(2\pi x_i)\right) + 20 + e$	Eq. (6), see [26]
Griewank's Function	$f_2(x) = \frac{1}{4000} \sum_{i=1}^m x_i^2 - \prod_{i=1}^m \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Eq. (7), see [26]
Levy's Function	$f_3(x) = \sin^2(\pi y_1) + \sum_{i=1}^{m-1} (y_i - 1)^2 \cdot [1 + 10 \sin^2(\pi y_i + 1)] + (y_m - 1)^2 [1 + \sin^2(2 * \pi x_n)]$ where $y_i = 1 + \frac{x_{i-1}}{4}$	Eq. (8), see [27]
Rastrigin's Function	$f_4(x) = \sum_{i=1}^m [x_i^2 - 10 \cos(2\pi x_i) + 10]$	Eq. (9), see [26]
Rosenbrock's Function	$f_5(x) = \sum_{i=1}^{m-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	Eq. (10), see [26]
Schwefel's Problem 1.2	$f_6(x) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j\right)^2$	Eq. (11), see [26]
Schwefel's Problem 2.21	$f_7(x) = \max_{i \in \{1, \dots, m\}} \{ x_i \}$	Eq. (12), see [26]
Sphere Function	$f_8(x) = \sum_{i=1}^n x_i^2$	Eq. (13), see [26]
Step Function	$f_9(x) = \sum_{i=1}^n (\lfloor x_i + 0.5 \rfloor)^2$	Eq. (14), see [26]
Neutral Problem 1	$f_{10}(x) = \frac{1}{m} \sum_{i=1}^m \sin^2 \frac{\lfloor i x_i \rfloor}{i}$	Eq. (15)
Neutral Problem 2	$f_{11}(x) = \frac{1}{m} \sum_{i=1}^m \begin{cases} x_i^2 & \text{if } x_i^2 \leq \frac{1}{i} \\ 1 & \text{otherwise} \end{cases}$	Eq. (16)

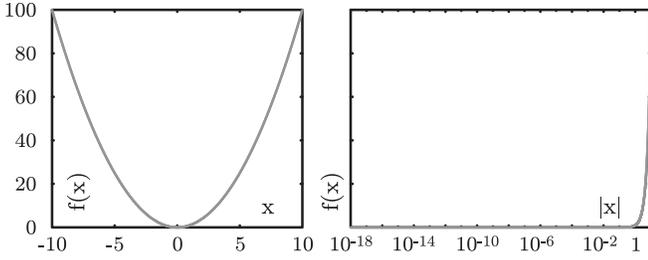


Fig. 8.a: The shape of the Sphere function (f_8).

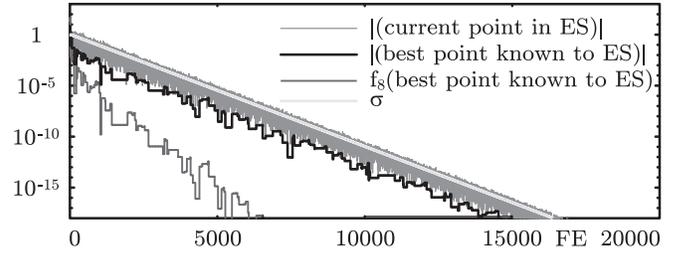


Fig. 8.b: The trace of a typical ES-FFA run on f_8 .

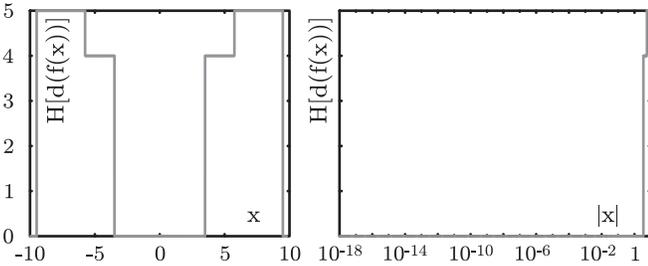


Fig. 8.c: The frequency state of that run after 10 FE.

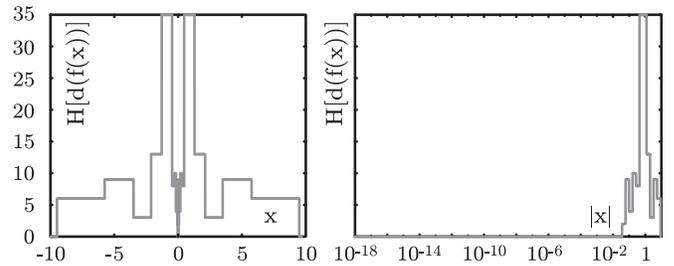


Fig. 8.d: The frequency state of that run after 100 FE.

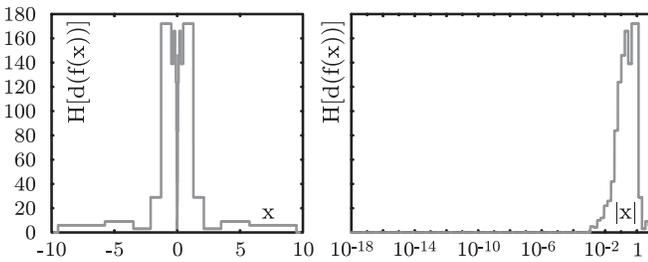


Fig. 8.e: The frequency state of that run after 1000 FE.

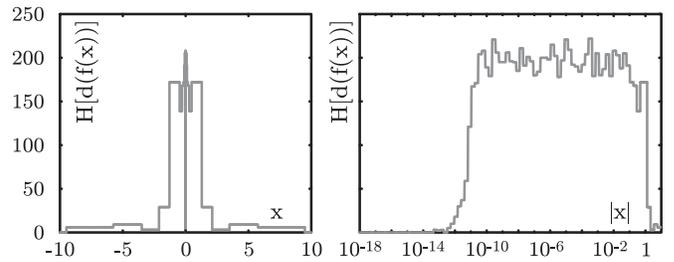


Fig. 8.f: The frequency state of that run after 10000 FE.

Fig. 8. Behavior of FFA on the Sphere function (f_8). Here we illustrate the behavior of a typical run of ES-FFA when optimizing the Sphere function (here f_8) plotted in Figure 8.a. Figure 8.b shows how the search progresses in terms of solution quality whereas Figures 8.c to 8.f illustrate the contents of the frequency table \mathbf{H} after different numbers function evaluations.

Table IV

PERFORMANCE IN TERMS OF STATISTICS AND PEAK QUALITY OF ES-FFA VS. THE CORRESPONDING ES-DIR SETTING BY FUNCTION. We compare ES-FFA and ES-DIR according to several different performance measures for different objective functions. ES-DIR often performs better than ES-FFA, yet ES-FFA still is able to find good solutions for the different benchmark functions.

function	total	L = 10			L = 100		
		a = .5	a = .8	a = .975	a = .5	a = .8	a = .975
statistical wins:evens:losses over functions. We used two-tailed Mann-Whitney U test at significance level 0.02 to compare the performance of ES-FFA settings over 100 runs with corresponding ES-DIR settings and aggregated the results in terms of <i>wins:evens:losses</i> . Per function, there are 6 different \overline{FE} values \times 5 dimensions \equiv 30 tests (Rosenbrock's function: only 4 dimensions ³).							
Ackley's	0:40:140	0:0:30	0:0:30	0:15:15	0:5:25	0:10:20	0:10:20
Griewank's	7:18:155	0:0:30	0:1:29	3:14:13	1:0:29	2:1:27	1:2:27
Levy's	30:113:37	0:6:24	0:18:12	19:10:1	0:30:0	6:24:0	5:25:0
Rastrigin's	0:35:145	0:0:30	0:0:30	0:15:15	0:0:30	0:10:20	0:10:20
Rosenbrock's	8:33:103	0:0:24	0:0:24	8:13:3	0:6:18	0:12:12	0:2:22
Schwefel 1.2	0:40:140	0:0:30	0:0:30	0:22:8	0:4:26	0:9:21	0:5:25
Schwefel 2.21	0:37:143	0:0:30	0:0:30	0:8:22	0:5:25	0:10:20	0:14:16
Sphere	0:70:110	0:6:24	0:6:24	0:24:6	0:7:23	0:15:15	0:12:18
Step	0:49:131	0:0:30	0:0:30	0:10:20	0:0:30	0:15:15	0:24:6
Neutral 1	18:137:25	0:12:18	0:24:6	9:21:0	0:30:0	4:26:0	5:24:1
Neutral 2	32:83:65	0:6:24	0:7:23	9:19:2	0:18:12	5:23:2	18:10:2
better:equal:worse best solutions over functions. For each ES-FFA configuration, the best solution over all results of the corresponding 100 runs is compared with the corresponding ES-DIR setup. (same 30 measurement points as before)							
Ackley's	12:57:111	0:5:25	1:8:21	2:11:17	2:12:16	3:9:18	4:12:14
Griewank's	90:20:70	12:0:18	12:5:13	18:3:9	17:4:9	18:4:8	13:4:13
Levy's	36:70:74	0:5:25	8:8:14	12:11:7	4:16:10	7:15:8	5:15:10
Rastrigin's	35:54:91	0:5:25	12:5:13	4:12:14	1:8:21	10:12:8	8:12:10
Rosenbrock's	52:10:82	6:0:18	5:0:19	15:4:5	2:0:22	13:3:8	11:3:10
Schwefel 1.2	25:43:112	0:0:30	0:6:24	9:5:16	4:12:14	9:10:11	3:10:17
Schwefel 2.21	15:34:131	0:3:27	1:6:23	3:7:20	1:6:23	3:8:19	7:4:19
Sphere	22:77:81	0:6:24	1:15:14	9:10:11	3:18:9	7:14:9	2:14:14
Step	0:165:15	0:24:6	0:24:6	0:30:0	0:29:1	0:29:1	0:29:1
Neutral 1	80:65:35	18:6:6	24:6:0	12:12:6	12:12:6	12:12:6	2:17:11
Neutral 2	57:37:86	13:0:17	3:9:18	12:8:10	6:9:15	8:6:16	15:5:10
smaller:equal:larger solution variances over functions. For each ES-FFA configuration, the variance over all results of the corresponding 100 runs is compared with the corresponding ES-DIR setup. (same 30 measurement points as before)							
Ackley's	47:12:121	24:0:6	18:0:12	2:6:22	0:0:30	1:3:26	2:3:25
Griewank's	5:0:175	0:0:30	0:0:30	5:0:25	0:0:30	0:0:30	0:0:30
Levy's	71:2:107	0:0:30	6:0:24	24:2:4	12:0:18	12:0:18	17:0:13
Rastrigin's	5:3:172	0:0:30	0:0:30	5:3:22	0:0:30	0:0:30	0:0:30
Rosenbrock's	6:2:136	0:0:24	0:0:24	6:2:16	0:0:24	0:0:24	0:0:24
Schwefel 1.2	5:8:167	0:0:30	0:0:30	5:8:17	0:0:30	0:0:30	0:0:30
Schwefel 2.21	13:7:160	6:0:24	0:0:30	0:2:28	0:0:30	0:2:28	7:3:20
Sphere	4:47:129	0:6:24	0:6:24	4:14:12	0:6:24	0:9:21	0:6:24
Step	0:21:159	0:0:30	0:0:30	0:0:30	0:0:30	0:0:30	0:21:9
Neutral 1	90:0:90	0:0:30	0:0:30	22:0:8	23:0:7	23:0:7	22:0:8
Neutral 2	131:2:47	24:0:6	24:0:6	19:0:11	19:0:11	19:0:11	26:2:2

information $\mathbf{H}[\mathbf{d}(\cdot)]$ within the ES-FFA is constructed and therefore plot the frequency table at four distinct function evaluations ($FE \in \{10, 100, 1000, 10000\}$) in Figures 8.c to 8.f. Each subfigure has two parts, on the left side we plot $\mathbf{H}[\mathbf{d}(f_8(x))]$ over the whole search space $x \in [-10, 10]$. On the right side, we use a logarithmically scaled x-axis from 10^{-18} to 10 (as f_8 is symmetric, i.e., $f_8(x) = f_8(-x) = f_8(|x|)$), we can join the positive and negative side into one diagram by plotting values for $|x|$).

After sampling ten points (Figure 8.c), high absolute values of x (and thus, f_8) have been discovered around five times, whereas the area around the optimum has not yet been sampled. Together with the worst possible areas on the borders of the search space, there are three undiscovered local optima in the frequency landscape.

Table V

PERFORMANCE IN TERMS OF STATISTICS AND PEAK QUALITY OF ES-FFA VS. THE CORRESPONDING ES-DIR SETTING BY DIMENSION m . We compare the ES-FFA and ES-DIR for rising function dimensions m and find that ES-FFA tends to lose more comparisons with rising m , likely because of an increasing span of possible objective values.

m	total	L = 10			L = 100		
		a = .5	a = .8	a = .975	a = .5	a = .8	a = .975
statistical wins:evens:losses over dimensions. We used two-tailed Mann-Whitney U test at significance level 0.02 to compare the performance of ES-FFA settings over 100 runs with corresponding ES-DIR settings and aggregated the results in terms of <i>wins:evens:losses</i> . Per dimension, there are 6 different \overline{FE} values \times 11 functions \equiv 66 tests (Rosenbrock's function: only 4 dimensions ³).							
1	18:192:150	0:18:42	0:25:35	8:40:12	1:34:25	3:38:19	6:37:17
2	22:154:220	0:12:54	0:12:54	8:42:16	0:22:44	4:37:25	10:29:27
3	20:136:240	0:0:66	0:12:54	10:37:19	0:19:47	5:36:25	5:32:29
4	18:95:283	0:0:66	0:6:60	9:31:26	0:12:54	5:26:35	4:20:42
5	17:78:301	0:0:66	0:1:65	13:21:32	0:18:48	0:18:48	4:20:42
better:equal:worse best solutions over dimensions. For each ES-FFA configuration, the best solution over all results of the corresponding 100 runs is compared with the corresponding ES-DIR setup. (same 66 measurement points as before.)							
1	86:206:68	13:30:17	12:40:8	19:32:9	13:38:9	13:34:13	16:32:12
2	112:158:126	18:6:42	25:26:15	21:27:18	12:35:19	17:33:16	19:31:16
3	88:135:173	12:6:48	12:17:37	19:24:23	10:29:27	21:28:17	14:31:21
4	48:88:260	0:12:54	6:9:51	18:18:30	3:16:47	16:15:35	5:18:43
5	90:45:261	6:0:60	12:0:54	19:12:35	14:8:44	23:12:31	16:13:37
smaller:equal:larger solution variances over dimensions. For each ES-FFA configuration, the variance over all results of the corresponding 100 runs is compared with the corresponding ES-DIR setup. (same 66 measurement points as before.)							
1	56:62:242	6:6:48	6:6:48	11:17:32	12:6:42	8:11:41	13:16:31
2	94:14:288	12:0:54	12:0:54	24:8:34	13:0:53	13:0:53	20:6:40
3	71:14:311	12:0:54	6:0:60	17:6:43	11:0:55	11:3:52	14:5:47
4	88:8:300	12:0:54	12:0:54	23:4:39	12:0:54	12:0:54	17:4:45
5	68:6:322	12:0:54	12:0:54	17:2:47	6:0:60	11:0:55	10:4:52

After 100 FE s (Figure 8.d), the frequency landscape now looks very different from the original f_8 . ES-FFA has entered the area round 0 and sampled many points around it, which created two big walls dividing it from the rest of the search space. If we remember that objective values are discretized logarithmically (see Equation 5) and that σ is constantly reduced due to self-adaptation, the right side of the plot gives better information. From the perspective of the ES-FFA, $|x| \in [1, 10]$ would be interesting, but is divided from the current x^* by walls of solutions with very bad fitness ($|x| \in [0.1, 1]$). On the left side of these solutions, however, there is a large local optimum, a (logarithmically) big space of solutions with a zero frequency.

This pool is now explored during the remaining FE s (Figures 8.e and 8.f). Whenever ES-FFA samples a point, it can either be in the unexplored area closer to the global optimum (thus having a low frequency), or on the wall separating the outside of the search space – hence having a bad fitness. The search is driven towards the global optimum 0. It samples many objective values of smaller and smaller scale, making the wall to the outside of the search space wider and wider.

b) Ackley's Function: On Ackley's function (f_1 in Table III), we illustrate an example run where FFA corrects the direction of the search. In Figure 9, we first plot the shape of Ackley's function (Figure 9.a) and the first 100 function evaluations of a typical run (Figure 9.b). f_1 is

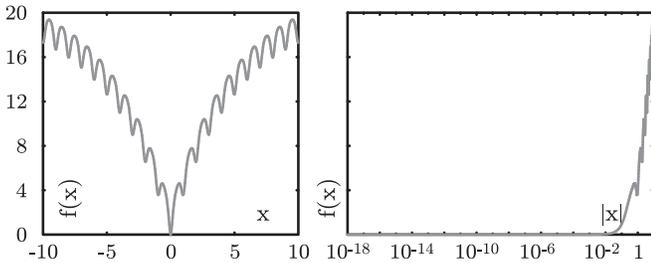
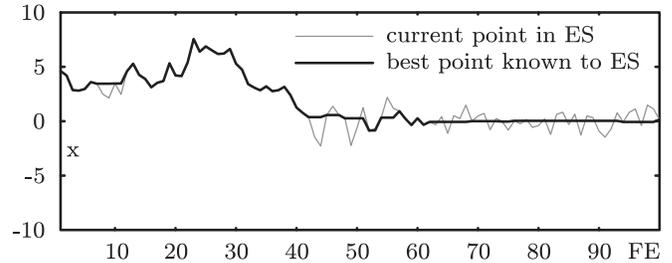
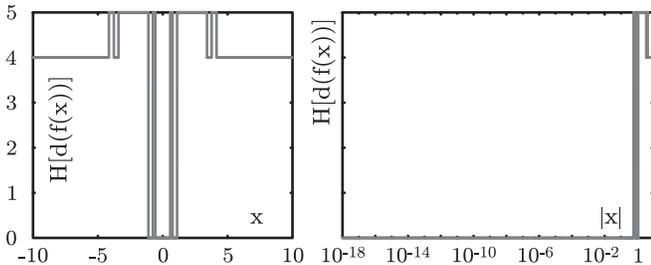
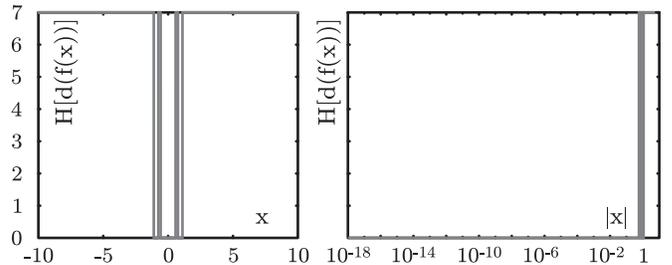
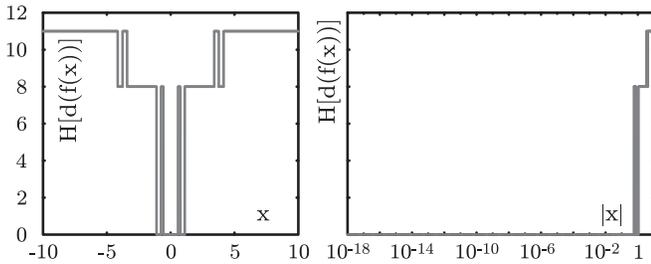
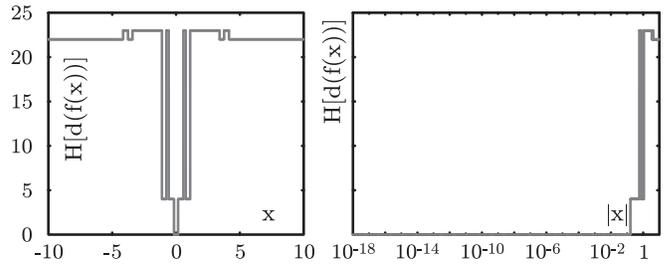
Fig. 9.a: The shape of Ackley's function (f_1).Fig. 9.b: The trace of a typical ES-FFA run on f_1 .Fig. 9.c: The frequency state of that run after 10 FEs .Fig. 9.d: The frequency state of that run after 15 FEs .Fig. 9.e: The frequency state of that run after 20 FEs .Fig. 9.f: The frequency state of that run after 50 FEs .

Fig. 9. Behavior of FFA on Ackley's function (f_1). Here we illustrate the behavior of a typical run of ES-FFA when optimizing the Ackley's function (here f_1) plotted in Figure 9.a. Figure 9.b shows how the search progresses in terms of solution quality whereas Figures 9.c to 9.f illustrate the contents of the frequency table \mathbf{H} after different numbers function evaluations.

symmetric around its global optimum 0. The illustrated run starts with a random initial point around 5, then searches in the wrong direction (towards 10) for some time, until it begins to converge towards the global optimum after around 30 FEs .

The four plots in Figures 9.c to 9.f illustrate the aggregated frequency information for 10, 15, 20, and 50 FEs . It can be seen that after 10 FEs , there are three large locally optimal areas in the frequency landscape: around the true global optimum 0 and around both ends of the search space. ES-FFA then explores the upper end of the search space. The objective values of this area, due to logarithmic discretization, largely fall into the same frequency table index and thus quickly degenerate in fitness (Figures 9.d and 9.e). Only the center area remains as the global optimum in the frequency landscape. Starting at around 50 FEs , ES-FFA again aggregates walls around this locally optimal area. The algorithm now behaves exactly in the way that we already observed for the Sphere function in the previous section. This makes sense, as in the right part of Figure 9.a, we see that the logarithmic-scale shape of Ackley's function is actually quite similar to the Sphere for $|x| < 1$.

c) *Discussion:* We analyzed ES-FFA with plots on multiple benchmark functions and generally observed the

behaviors discussed in the previous section. In most of the runs that we plotted, the behavior and convergence of ES-FFA is very similar to ES-DIR.

In some cases, however, ES-FFA would prematurely converge, which accounts for its worse overall performance. This seems to be puzzling, as FFA is a technique devised exactly to prevent this. The reason is that the $1/5^{th}$ rule prescribes that the mutation step width should be decreased when too many mutations fail. If this happens too early and σ gets so small that no points outside the current "bucket" $d(\cdot)$ of \mathbf{H} can be sampled anymore, mutation will no longer succeed and σ will continue to be decreased. We observe this effect especially with low settings for L and a and even more under the more aggressive adaptation rule given in [46].

Some method d to discretize the objective values is necessary if FFA is applied to a continuous problem – there can only be finitely many indices for the table \mathbf{H} . This experiment here emphasizes the influence of d . The static, logarithm-based approach given in Equation 5 clashes with the self-adaptation of the ES. If future applications of FFA for continuous optimization are to be competitive, it is necessary to either devise better mappings, to develop a new approach not needing a table \mathbf{H} , or to apply other methods like Novelty

Search [2, 3] that do not have this requirement in the first place instead.

V. CONCLUSIONS

In this paper, we introduced FFA as a new fitness assignment method and showed that it (a) can self-adaptively and smoothly switch between exploration and exploitation, (b) can significantly improve the solution quality, (c) can increase the solution diversity, and (d) can be integrated both into basic metaheuristics as well as into complicated existing optimization systems in a straightforward way (see Section IV-C).

We found that it performs particularly well on problems where most randomly sampled candidate solutions have bad objective values. In the Genetic Programming experiment (Section IV-A), where the objective function is very rugged and there is a high degree of epistasis in the search space, it led to a higher solution quality compared to the original algorithm.

We then thoroughly explored potential pitfalls of using plain FFA. Problems arise when most randomly sampled candidate solutions already have good objective values (see Section IV-B) and in configurations where the hosting algorithm's self-adaptation capabilities work against those of FFA (see Section IV-D). Here, more research is necessary.

In summary, we replaced the objective function f with an indirect metric \mathbf{H} , which behaves entirely differently, changes dynamically, and does not correlate with f . Still the four different algorithms that we tested retained their optimization ability. Our experiments thus contribute to the body of research [2–6] indicating the possibility to utilize fitness measures for driving the optimization processes that are very different from the objective functions, which at the present time are the fuel for most applications of optimization.

Together with the works of Lehman and Stanley [2, 3] and Hutter [4–6], this raises some fundamental questions. If optimization can still work well or better even though not having direct access to the objective function f , this can either mean that (a) there is only little information in f or (b) that the optimization method is bad in exploiting this information. In tune with [2], the first take-away message of this paper thus is that the concept of objective functions may be over-emphasized in optimization. For the same reasons, one could ask whether there are classes of optimization problems where the metaheuristic idea of iteratively modifying, refining, and combining candidate solutions is not the best idea (and how many of the well-known optimization tasks and benchmarks actually fall into them). There thus definitely exists the need for further experiments which may result in the development of different, new search paradigms.

We will therefore investigate the properties of FFA more thoroughly, try to build on its advantages and mitigate the drawbacks that we have presented here. Our future work furthermore will comprise the following four points:

(a) As mentioned before, FFA turns a static optimization problem into a dynamic one. We would therefore also like to combine it with methods for dynamic problems such as [47]. (b) In the current experiments, we used FFA directly. However, it seems logical to combine it with the original objective. As the second point in our future work,

we will take a single-objective problem and translate it into a multi-objective one by adding the Frequency Fitness as the second optimization criterion. A similar approach has already been successfully developed by Mouret [19] for Novelty Search. In order to test this approach, we will use the well-known multi-objective evolutionary algorithm NSGA-II [48] together with FFA. (c) Instead of using a table for the history information gathered during an optimization process, we also want to investigate to fit and update a probability distribution to the observed objective values and assign fitness based on objective value sampling likelihood under this distribution. On one hand, this would make a discretization unnecessary. On the other hand, it makes more assumptions about the objective function and optimization process. It should be investigated whether such a method can be a viable alternative to FFA. (d) Finally, it would also be interesting to investigate the influence of the size of the search space on the performance of FFA. In Section IV-D, we found that increasing the dimension m of the real-valued search space led to a larger decrease in the quality of solutions given by ES-FFA compared to ES-DIR. However, an increase in m also led to an increase of the reachable upper bound of the f_i , i.e., increased the number of possible objective values, which too, has a negative impact on FFA. Finding which of the two issues has the larger impact would likely provide us with significant insights.

Acknowledgements. This work was supported by the 973 Program of China (Grant No. 2011CB707006), National Natural Science Foundation of China under Grants 61150110488, 61028009, and 61175065, the Special Financial Grant 201104329 from the China Postdoctoral Science Foundation, the Chinese Academy of Sciences (CAS) Fellowship for Young International Scientists #2011Y1GB01, the Natural Science Foundation of the Anhui Province under Grant 1108085J16, and the European Union 7th Framework Program under Grant 247619.

REFERENCES

- [1] T. Weise, M. Wan, K. Tang, P. Wang, A. Devert, and X. Yao, "Frequency Fitness Assignment," *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, vol. 18, no. 2, pp. 226–243, Apr. 2014.
- [2] J. Lehman and K. O. Stanley, "Abandoning Objectives: Evolution Through the Search for Novelty Alone," *Evolutionary Computation*, vol. 19, no. 2, pp. 189–223, Summer 2011. [Online]. Available: http://eplex.cs.ucf.edu/papers/lehman_ecj10.pdf
- [3] J. Lehman and K. O. Stanley, "Exploiting Open-Endedness to Solve Problems Through the Search for Novelty," in *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems (Artificial Life XI)*, S. Bullock, J. Noble, R. A. Watson, and M. A. Bedau, Eds. Winchester, Hampshire, UK: Cambridge, MA, USA: MIT Press, August 5–8, 2008, pp. 329–336. [Online]. Available: http://eplex.cs.ucf.edu/papers/lehman_alife08.pdf
- [4] S. Legg, M. Hutter, and A. Kumar, "Tournament versus Fitness Uniform Selection," Manno-Lugano, Switzerland: Dalle Molle Institute for Artificial Intelligence (IDSIA), University of Lugano, Faculty of Informatics / University of Applied Sciences of Southern Switzerland (SUPSI), Department of Innovative Technologies, Tech. Rep. IDSIA-04-04, March 4, 2004. [Online]. Available: <http://www.idsia.ch/idsiareport/IDSIA-04-04.pdf>
- [5] S. Legg, M. Hutter, and A. Kumar, "Tournament versus Fitness Uniform Selection," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'04)*. Portland, OR, USA: Los Alamitos, CA, USA: IEEE Computer Society Press, June 20–23, 2004, pp. 2144–2151. [Online]. Available: <http://arxiv.org/abs/cs/0403038v1>
- [6] M. Hutter, "Fitness Uniform Selection to Preserve Genetic Diversity," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02), 2002 IEEE World Congress on Computational Intelligence (WCCI'02)*, D. B. Fogel, M. A. El-Sharkawi, X. Yao, H. Iba, P. Marrow, and M. Shackleton, Eds., vol. 1-2, Piscataway, NJ, USA: IEEE Computer Society. Honolulu, HI, USA: Hilton Hawaiian Village Hotel (Beach Resort & Spa): Los Alamitos, CA, USA: IEEE Computer Society Press, May 12–17, 2002, pp. 783–788, also: Technical Report IDSIA-01-01, 17 January 2001. [Online]. Available: <http://arxiv.org/abs/cs/0103015>
- [7] C. Blum, R. Chiong, M. Clerc, K. A. De Jong, Z. Michalewicz, F. Neri, and T. Weise, "Evolutionary Optimization," in *Variants of Evolutionary*

- Algorithms for Real-World Applications*, R. Chiong, T. Weise, and Z. Michalewicz, Eds. Berlin/Heidelberg: Springer-Verlag, 2011, ch. 1, pp. 1–29.
- [8] T. Weise, *Global Optimization Algorithms – Theory and Application*. Germany: it-weise.de (self-published), 2009. [Online]. Available: <http://www.it-weise.de/projects/book.pdf>
- [9] T. Weise, R. Chiong, and K. Tang, “Evolutionary Optimization: Pitfalls and Booby Traps,” *Journal of Computer Science and Technology (JCST)*, vol. 27, no. 5, pp. 907–936, September 2012, special Issue on Evolutionary Computation, edited by Xin Yao and Pietro S. Oliveto.
- [10] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. [Online]. Available: <http://books.google.de/books?id=2IIJAAAACAAJ>
- [11] M. Hutter and S. Legg, “Fitness Uniform Optimization,” *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, vol. 10, no. 5, pp. 568–589, October 2006. [Online]. Available: <http://arxiv.org/abs/cs/0610126v1>
- [12] D. E. Goldberg and J. T. Richardson, “Genetic Algorithms with Sharing for Multimodal Function Optimization,” in *Proceedings of the Second International Conference on Genetic Algorithms and their Applications (ICGA’87)*, J. J. Grefenstette, Ed. Cambridge, MA, USA: Massachusetts Institute of Technology (MIT): Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc. (LEA), July 28–31, 1987, pp. 41–49.
- [13] K. Deb and D. E. Goldberg, “An Investigation of Niche and Species Formation in Genetic Function Optimization,” in *Proceedings of the Third International Conference on Genetic Algorithms (ICGA’89)*, J. D. Schaffer, Ed. Fairfax, VA, USA: George Mason University (GMU): San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., June 4–7, 1989, pp. 42–50.
- [14] A. Pétrowski, “A Clearing Procedure as a Niche Method for Genetic Algorithms,” in *Proceedings of IEEE International Conference on Evolutionary Computation (CEC’96)*, K. Jidō and S. Gakkai, Eds. Nagoya, Aichi, Japan: Nagoya University, Symposium & Toyoda Auditorium: Los Alamitos, CA, USA: IEEE Computer Society Press, 1996, pp. 798–803. [Online]. Available: <http://sci2s.ugr.es/docencia/cursosMierres/clearing-96.pdf>
- [15] T. Weise, S. Niemczyk, R. Chiong, and M. Wan, “A Framework for Multi-Model EDAs with Model Recombination,” in *Proceedings of the 4th European Event on Bio-Inspired Algorithms for Continuous Parameter Optimisation (EvoNUM’11)*, *Applications of Evolutionary Computation – Proceedings of EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Part 1 (EvoAPPLICATIONS’11)*, ser. Lecture Notes in Computer Science (LNCS), vol. 6624. Torino, Italy: Berlin, Germany: Springer-Verlag GmbH, April 27–29, 2011, pp. 304–313.
- [16] S. Legg and M. Hutter, “Fitness Uniform Deletion: A Simple Way to Preserve Diversity,” in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO’05)*, H. Beyer, U. O’Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantú-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llorà, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, J. Watson, and E. Zitzler, Eds. Washington, DC, USA: Loews L’Enfant Plaza Hotel: New York, NY, USA: ACM Press, June 25–27, 2005, pp. 1271–1278. [Online]. Available: <http://arxiv.org/abs/cs/0504035v1>
- [17] J. Lehman and K. O. Stanley, “Evolving a Diversity of Virtual Creatures Through Novelty Search and Local Competition,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO’11)*, Dublin, Ireland, July 12–16, 2011, pp. 211–218. [Online]. Available: http://eplex.cs.ucf.edu/papers/lehman_gecco11.pdf
- [18] G. Cuccu, F. Gomez, and T. Glasmachers, “Novelty-Based Restarts for Evolution Strategies,” in *Proceedings of the 12th IEEE Congress on Evolutionary Computation (CEC’11)*, New Orleans, LA, USA, June 5–8, 2011, pp. 158–163. [Online]. Available: <http://www.idsia.ch/~giuse/papers/cuccu11cec.pdf>
- [19] J. Mouret, “Novelty-based Multiobjectivization,” in *New Horizons in Evolutionary Robotics – Extended Contributions from the 2009 EvoDeRob Workshop (EvoDeRob’09)*, ser. Studies in Computational Intelligence, S. Doncieux, N. Bredèche, and J. Mouret, Eds., vol. 341. St. Louis, MO, USA: Berlin/Heidelberg: Springer-Verlag, 2009, pp. 139–154. [Online]. Available: chronos.isir.upmc.fr/~mouret/papers/novelty_based_multiobjectivization.pdf
- [20] F. W. Glover, É. D. Taillard, and D. de Werra, “A User’s Guide to Tabu Search,” *Annals of Operations Research*, vol. 41, no. 1, pp. 3–28, March 1993, special issue on Tabu search.
- [21] A. Hertz, É. D. Taillard, and D. de Werra, “A Tutorial on Tabu Search,” in *Giornate di Lavoro (Entreprise Systems: Management of Technological and Organizational Changes, “Gestione del cambiamento tecnologico ed organizzativo nei sistemi d’impresa”) (AIRO’95)*, F. Pezzella, Ed. Ancona, Italy: Associazione Italiana di Ricerca Operativa – Optimization and Decision Sciences (AIRO), September 20–22, 1995, pp. 13–24. [Online]. Available: http://www.dei.unipd.it/~fisch/ricop/tabu_search_tutorial.pdf
- [22] M. Wan, T. Weise, and K. Tang, “Novel Loop Structures and the Evolution of Mathematical Algorithms,” in *Proceedings of the 14th European Conference on Genetic Programming (EuroGP’11)*, ser. Lecture Notes in Computer Science (LNCS), S. Silva, J. A. Foster, M. Nicolau, P. Machado, and M. Giacobini, Eds., vol. 6621/2011. Torino, Italy: Berlin, Germany: Springer-Verlag GmbH, April 27–29, 2011, pp. 49–60.
- [23] H. H. Hoos and T. Stützle, “SATLIB – The Satisfiability Library,” March 22, 2005. [Online]. Available: <http://www.satlib.org/>
- [24] H. H. Hoos and T. Stützle, “SATLIB: An Online Resource for Research on SAT,” in *SAT2000 – Highlights of Satisfiability Research in the Year 2000*, ser. Frontiers in Artificial Intelligence and Applications, I. Gent, H. van Maaren, and T. Walsh, Eds. Amsterdam, The Netherlands: IOS Press, 2000, vol. 63, pp. 283–292. [Online]. Available: <http://www.cs.ubc.ca/~hoos/Publ/sat2000-satlib.pdf>
- [25] “UC Irvine Machine Learning Repository,” 2011. [Online]. Available: <http://archive.ics.uci.edu/ml/>
- [26] X. Yao, Y. Liu, and G. Lin, “Evolutionary Programming Made Faster,” *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, vol. 3, no. 2, pp. 82–102, July 1999. [Online]. Available: http://www.u-aizu.ac.jp/~yliu/publication/tcc22r2_online.ps.gz
- [27] A. V. Levy and A. J. Montalvo, “The Tunneling Algorithm for the Global Minimization of Functions,” *SIAM Journal of Scientific and Statistical Computing*, vol. 6, no. 1, pp. 15–29, 1985.
- [28] P. Wang, K. Tang, E. P. K. Tsang, and X. Yao, “A Memetic Genetic Programming with Decision Tree-Based Local Search for Classification Problems,” in *Proceedings of the 12th IEEE Congress on Evolutionary Computation (CEC’11)*, New Orleans, LA, USA, June 5–8, 2011, pp. 917–924. [Online]. Available: <http://home.ustc.edu.cn/~wyuyou308/publications/CEC2011.pdf>
- [29] H. Beyer and H. Schwefel, “Evolution Strategies – A Comprehensive Introduction,” *Natural Computing: An International Journal*, vol. 1, no. 1, pp. 3–52, March 2002. [Online]. Available: <http://www.cs.bham.ac.uk/~pxt/NIL/es.pdf>
- [30] H. Beyer, *The Theory of Evolution Strategies*, ser. Natural Computing Series. New York, NY, USA: Springer New York, May 27, 2001. [Online]. Available: <http://books.google.de/books?id=8tbnLufkTMC>
- [31] I. Rechenberg, “Evolutionstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution,” Ph.D. dissertation, Berlin, Germany: Technische Universität Berlin, 1971. [Online]. Available: <http://books.google.de/books?id=QcNNGQAACAAJ>
- [32] T. Weise and K. Tang, “Evolving Distributed Algorithms with Genetic Programming,” *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, vol. 16, no. 2, pp. 242–265, April 2012.
- [33] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, ser. Bradford Books. Cambridge, MA, USA: MIT Press, December 1992, 1992 first edition, 1993 second edition. [Online]. Available: <http://books.google.de/books?id=Bhtxo60BV0EC>
- [34] T. Weise, M. Zapf, and K. Geihs, “Rule-based Genetic Programming,” in *Proceedings of the 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems (BIONETICS’07)*, Budapest, Hungary: Radisson SAS Beke Hotel: Piscataway, NJ, USA: IEEE Computer Society, December 10–13, 2007, pp. 8–15.
- [35] T. Weise, “Illustration of Statistical Test Results for Experiment Evaluation,” Hefei, Anhui, China: University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL), Tech. Rep., March 2, 2011, superseded by: Thomas Weise and Raymond Chiong, “An Alternative Way of Presenting Statistical Test Results when Evaluating the Performance of Stochastic Approaches,” *Neurocomputing*, 147:235–238, January 5, 2015. doi:10.1016/j.neucom.2014.06.071.
- [36] S. B. Rana and L. D. Whitley, “Genetic Algorithm Behavior in the MAXSAT Domain,” in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN V)*, ser. Lecture Notes in Computer Science (LNCS), Á. E. Eiben, T. Bäck, M. Schoenauer, and H. Schwefel, Eds., vol. 1498/1998. Amsterdam, The Netherlands: Berlin, Germany: Springer-Verlag GmbH, September 27–30, 1998, pp. 785–794. [Online]. Available: www.cs.colostate.edu/~genitor/1998/ppsn98a.ps.gz
- [37] B. Evans, *Cylinder Bands Data Set*. Irvine, CA, USA: UCI

Machine Learning Repository, Center for Machine Learning and Intelligent Systems, Donald Bren School of Information and Computer Science, University of California, August 1, 1995. [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/Cylinder+Bands>

- [38] *Credit Approval Data Set*. Irvine, CA, USA: UCI Machine Learning Repository, Center for Machine Learning and Intelligent Systems, Donald Bren School of Information and Computer Science, University of California, 1987. [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/Credit+Approval>
- [39] *Statlog (Heart) Data Set*. Irvine, CA, USA: UCI Machine Learning Repository, Center for Machine Learning and Intelligent Systems, Donald Bren School of Information and Computer Science, University of California, 1997. [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29>
- [40] L. Graham and F. Oppacher, *Hill-Valley Data Set*. Irvine, CA, USA: UCI Machine Learning Repository, Center for Machine Learning and Intelligent Systems, Donald Bren School of Information and Computer Science, University of California, March 20, 2008. [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/Hill-Valley>
- [41] M. Elter and R. Schulz-Wendland, *Mammographic Mass Data Set*. Irvine, CA, USA: UCI Machine Learning Repository, Center for Machine Learning and Intelligent Systems, Donald Bren School of Information and Computer Science, University of California, October 29, 2007. [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/Mammographic+Mass>
- [42] S. B. Thrun, *MONK's Problems Data Set*. Irvine, CA, USA: UCI Machine Learning Repository, Center for Machine Learning and Intelligent Systems, Donald Bren School of Information and Computer Science, University of California, October 1, 1992. [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/MONK%27s+Problems>
- [43] M. Hopkins, E. Reeber, G. Forman, and J. Suermondt, *Spambase Data Set*. Irvine, CA, USA: UCI Machine Learning Repository, Center for Machine Learning and Intelligent Systems, Donald Bren School of Information and Computer Science, University of California, July 1, 1999. [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/Spambase>
- [44] L. I. Kuncheva and C. J. Whitaker, "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy," *Machine Learning*, vol. 51, no. 2, pp. 181–207, May 2003. [Online]. Available: <http://pages.bangor.ac.uk/~mas00a/papers/lkml.pdf>
- [45] *ANSI/IEEE Std 754-1985 – IEEE Standard for Binary Floating-Point Arithmetic*. Piscataway, NJ, USA: IEEE (Institute of Electrical and Electronics Engineers) and Washington, DC, USA: American National Standards Institute (ANSI), March 21, 1985. [Online]. Available: <http://754r.ucbtest.org/standards/754.pdf>
- [46] A. Auger, "Benchmarking the (1+1) Evolution Strategy with One-Fifth Success Rule on the BBOB-2009 Function Testbed," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO'09)*, F. Rothlauf, G. R. Raidl, A. I. Esparcia-Alcázar, Y. Chen, G. Ochoa, E. Ozcan, M. Schoenauer, A. Auger, H. Beyer, N. Hansen, S. Finck, R. Ros, L. D. Whitley, G. Wilson, S. Harding, W. B. Langdon, M. L. Wong, L. D. Merkle, F. W. Moore, S. G. Ficici, W. Rand, R. L. Riolo, N. Kharm, W. R. Buckley, J. F. Miller, K. O. Stanley, J. Bacardit i Peñarroya, W. N. Browne, J. Drugowitsch, N. Beume, M. Preuß, S. F. Smith, S. Cagnoni, A. Floares, A. Baughman, S. M. Gustafson, M. Keijzer, A. Kordon, and C. B. Congdon, Eds. Montréal, QC, Canada: Delta Centre-Ville Hotel: New York, NY, USA: Association for Computing Machinery (ACM), July 8–12, 2009, pp. 2447–2452. [Online]. Available: hal.archives-ouvertes.fr/docs/00/43/05/15/PDF/wk2037-auger.pdf
- [47] S. Yang, Y. Ong, and Y. Jin, Eds., *Evolutionary Computation in Dynamic and Uncertain Environments*, ser. Studies in Computational Intelligence. Berlin/Heidelberg: Springer-Verlag, 2007, vol. 51/2007. [Online]. Available: http://www.soft-computing.de/Jin_CEC04T.pdf.gz
- [48] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," in *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN VI)*, ser. Lecture Notes in Computer Science (LNCS), M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo-Guervós, and H. Schwefel, Eds., vol. 1917/2000. Paris, France: Berlin, Germany: Springer-Verlag GmbH, September 18–20, 2000, pp. 849–858. [Online]. Available: <https://eprints.kfupm.edu.sa/17643/1/17643.pdf>



Thomas Weise (M'10) received the Diplom Informatiker (equivalent to M.Sc.) degree from the Department of Computer Science, Chemnitz University of Technology, Chemnitz, Germany, in 2005, and the Ph.D. degree at the Distributed Systems Group of the Fachbereich Elektrotechnik und Informatik, University of Kassel, Kassel, Germany in 2009. He then took a position post-doctoral researcher at the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China. He now is Associate Professor at the USTC-

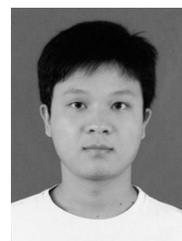
Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI) which is part of the same school. His major research interests include planning for logistics applications, Evolutionary Computation, Genetic Programming, and real-world applications of optimization algorithms. His experience ranges from applying GP to distributed systems and multi-agent systems, efficient web service composition for Service Oriented Architectures, to solving large-scale real-world vehicle routing problems for multimodal logistics and transportation. Besides being the author/co-author of over 60 refereed publications, Dr. Weise also authors the electronic book *Global Optimization Algorithms – Theory and Application* which is freely available at his website (<http://www.it-weise.de/>).

Mingxu Wan Mingxu Wan graduated from the USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI), School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China in 2012 and now pursues a career in the software industry.



Ke Tang Ke Tang (S'05—M'07—SM'12) received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2002, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2007.

In 2007, he joined the USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI), School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China, and was promoted to professor in 2011. He is also an Honorary Senior Research Fellow in the School of Computer Science, University of Birmingham, UK. He has authored/co-authored more than 50 refereed publications. His major research interests include evolutionary computation and machine learning. He is an associate editor of IEEE Computational Intelligence Magazine and served as a program co-chair of CEC2010, held in Barcelona.



Pu Wang Pu Wang received the Bachelor's degree in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 2008.

He is currently pursuing the Ph.D. degree from the USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI), School of Computer Science and Technology, USTC. His current research interests include evolutionary computation, memetic algorithms, and multiobjective optimization for classification prob-

lems in machine learning, and genetic programming for model checking in software engineering.



Alexandre Devert A French citizen, Alexandre obtained his Master and his PhD in computer science in Paris, Orsay University. Then, he did his PostDoc at the USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI), School of Computer Science and Technology, USTC in Hefei, China. He liked it enough to stay at the USTC and became both a lecturer and a researcher. By vocation, his interest are in optimization (most often stochastic optimization), especially in seeking for scalable problem

representations. His favorite problems are taken from civil engineering and operational research. By trade, he also has experience and interest is data-mining applied to bio-informatics problems.



Xin Yao (M'91—SM'96—F'03) is a Chair (Professor) of Computer Science at the University of Birmingham, UK. He is the Director of CERCIA (the Centre of Excellence for Research in Computational Intelligence and Applications), University of Birmingham, UK, and of the Joint USTC-Birmingham Research Institute of Intelligent Computation and Its Applications. He is an IEEE Fellow and a Distinguished Lecturer of IEEE Computational Intelligence Society (CIS). He won the 2001 IEEE Donald G. Fink Prize Paper Award, 2010 IEEE

Transactions on Evolutionary Computation Outstanding Paper Award, 2010 BT Gordon Radley Award for Best Author of Innovation (Finalist), 2011 IEEE Transactions on Neural Networks Outstanding Paper Award, and many other best paper awards at conferences. He won the prestigious Royal Society Wolfson Research Merit Award in 2012 and was selected to receive the 2013 IEEE CIS Evolutionary Computation Pioneer Award. He was the Editor-in-Chief (2003-08) of IEEE Transactions on Evolutionary Computation. He has been invited to give more than 65 keynote/plenary speeches at international conferences in many different countries. His major research interests include evolutionary computation and neural network ensembles. He has more than 400 refereed publications in international journals and conferences.

This is a preview version of paper [1] (see page 19 for the reference). It is posted here for your personal use and not for redistribution. The final publication and definite version is available from IEEE (who hold the copyright) at

<http://www.ieee.org/>. See also

<http://dx.doi.org/10.1109/TEVC.2013.2251885>.

```
@article{WWTWDY2014FFA,
  author    = {Thomas Weise and Mingxu Wan and Ke Tang and Pu Wang and
              Alexandre Devert and Xin Yao},
  title     = {{Frequency Fitness Assignment}},
  publisher = {Washington, DC, USA: IEEE Computer Society},
  journal   = {IEEE Transactions on Evolutionary Computation (IEEE-EC)},
  number    = {2},
  volume    = {18},
  pages     = {226--243},
  year      = {2014},
  month     = apr,
  doi       = {10.1109/TEVC.2013.2251885},
  inspec    = {14196623},
},
```