# When Frequency Fitness Assignment Fails: Trapped States in Frequency-Guided Local Search

Jiazheng Zeng
zengjiazheng@stu.hfuu.edu.cn
School of Artificial Intelligence and Big Data,
Hefei University
Hefei, Anhui, China

Thomas Weise
tweise@hfuu.edu.cn
School of Artificial Intelligence and Big Data,
Hefei University
Hefei, Anhui, China

Zhize Wu
wuzz@hfuu.edu.cn
School of Artificial Intelligence and Big Data,
Hefei University
Hefei, Anhui, China

Markus Wagner
Markus.Wagner@monash.edu
Faculty of Information Technology, Monash University
Clayton, Australia

## Abstract

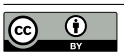
Frequency Fitness Assignment (FFA) offers an alternative take on metaheuristic optimization. Here, the encounter frequencies of objective values are used to make the selection decisions. This leads to a variety of interesting algorithm features, such as an invariance under all injective transformations of the objective function value and a very strong focus on exploration of the search space. In this article, for the first time, we discover a condition under which purely FFA-guided search can actually get stuck, even on a problem as simple as OneMax. To tackle this issue, we suggest hybrid approaches combining objective-guided and FFA-guided search. We propose using crossover for solution transfer between the two component algorithms of the hybrids. Our experiments show that (1) the original FFA allows us to solve problems like Trap, TwoMax, and Jump in (experimentally observed) polynomial time; (2) the suggested hybrids address FFA's shortcomings and are occasionally orders of magnitude faster; and (3) we report several new best-known solutions for the $\mathcal{NP}$-hard low-autocorrelation binary sequences problem.

## CCS Concepts

• **Theory of computation → Approximation algorithms analysis**; **Theory of randomized search heuristics**; • **Mathematics of computing** → Discrete mathematics; Probabilistic algorithms; • **Computing methodologies → Discrete space search**; **Randomized search**; *Heuristic function construction*; Learning paradigms.

## Keywords

Discrete Optimization, (1+1) EA, Randomized Local Search, Frequency Fitness Assignment, (1+1) FEA, OneMax, LeadingOnes, TwoMax, Jump, Trap, Plateau, Low-Autocorrelation Binary Sequence




*GECCO '26, San Jose, Costa Rica*

ACM ISBN 979-8-4007-2487-9/2026/07
https://doi.org/10.1145/3795095.3805098

**ACM Reference Format:**
Jiazheng Zeng, Thomas Weise, Zhize Wu, and Markus Wagner. 2026. When Frequency Fitness Assignment Fails: Trapped States in Frequency-Guided Local Search. In *Genetic and Evolutionary Computation Conference (GECCO '26), July 13–17, 2026, San Jose, Costa Rica.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3795095.3805098

## 1 Introduction

Discrete optimization benchmark problems are an important cornerstone of fundamental research in black-box optimization. They help us to reveal, understand, or even prove algorithm behavior and performance. Without loss of generality, we consider them as functions $f : \{0,1\}^n \mapsto \mathbb{N}_0$ that accept a string of $n$ bits as input, compute a natural number as result, and are subject to minimization. The commonly used discrete optimization benchmark problems include OneMax, TwoMax, LeadingOnes, Trap, Ising models, the N-Queens problem, Jump, Plateau, the W-Model, the Linear Harmonic functions, and the Binary Integer problem [15, 16]. These problems exhibit a wide range of search space characteristics, such as deceptiveness, neutrality, non-separability, ruggedness, and multimodality. They do so in an obvious and easy-to-understand way.

And they should. It is not easy to divine how an algorithm will perform on a complicated problem. The simple structure of such problems has enabled us to widen our understanding of optimization algorithms both theoretically and experimentally. In this work, we use such benchmark problems to experimentally explore the behavior of a counter-intuitive idea that eludes theoretical analysis.

Metaheuristics are built upon the following paradigm: *"Iteratively create modified copies of selected solutions and the better a solution is, the higher be its probability to get selected."* This biased selection toward better objective values has given us a myriad of highly-efficient algorithms.

Frequency Fitness Assignment (FFA) [56, 58, 59] offers a different take. Plugging FFA into an existing optimization algorithm means to modify its selection step: Every time before a solution $x$ enters the selection stage, the encounter frequency $H[f(x)]$ of its objective value $f(x)$ is incremented by one. The selection then prefers solutions with lower objective value encounter frequencies (regardless whether they are better or worse).

This creates optimization processes that are (1) *not* biased toward better solutions. It also renders optimization algorithms (2) invariant under all injective transformations of the objective function value [58]. The only other algorithms with even one of these properties are random walks, random sampling, and exhaustive enumeration—none of which are useful optimization methods. Yet, FFA has delivered surprisingly good performance on a wide range of hard optimization problems [8, 10, 33, 35, 52, 54, 55, 59].

The existing works on FFA exhibit a couple of blind spots. First, two of the most fundamental metaheuristics for discrete optimization are the (1+1) EA and the randomized local search (RLS), which only differ in their unary search operator. FFA has been plugged into the (1+1) EA yielding the (1+1) FEA, which performed significantly better on some hard (but not $\mathcal{NP}$-hard) discrete benchmarks *and* on the $\mathcal{NP}$-hard maximum satisfiability problem (Max-Sat) [58, 59]. However, it has not yet been plugged in the even simpler RLS on this domain. We ask: *What happens if FFA is applied in* RLS*, where the search operator spans a very small neighborhood?*

Second, several hybrid algorithms that combine FFA-guided strand (A) and objective-guided strand (B) exist [33, 59]. Here, (A) usually copies its solution over to (B) if some conditions are met. This always was a "hard copy" that overwrites the current solution in that branch. However, we actually have a very nice operator for combining two solutions, namely crossover. We ask: *Is crossover a better way to hybridize FFA- and objective-guided search?*

Third, only one single $\mathcal{NP}$-hard optimization problem with a bitstring domain has been considered as application area for FFA so far, namely the aforementioned Max-Sat. Here, all investigated FFA-based black-box optimization algorithm variants outperformed their objective-guided original versions in the large study [59]. Max-Sat has the advantage that its instances tend to exhibit very few different objective values. This is favorable for FFA, which spreads the search effort evenly over the range of the objective function [34]. We now explore another important $\mathcal{NP}$-hard domain, namely the discovery of low-autocorrelation binary sequence (LABS). LABS instances tend to exhibit more different objective values than Max-Sat instances of the same scale $n$. We ask: *Does FFA yield good results on a second discrete $\mathcal{NP}$-hard domain?*

With the present study, we close three research gaps and significantly expand our understanding of FFA. In particular, we expose one weakness of the method: While FFA-based algorithms are shown to oscillate forever in the objective space and thereby often eventually hit the optimum, they actually can get stuck, too. This requires very specific conditions to appear, which we will uncover in our experiments.

The remainder of this paper is organized as follows: Section 2 briefly discusses related work. In Section 3, we introduce the algorithms used in this study. In Section 4, we introduce the problems investigated alongside the results of the algorithms when applied to them. Finally, our conclusions and outlook on future work are given in Section 5.

An immutable reproduction package (including code and instructions) is available at doi:10.5281/zenodo.18333914.

## 2 Related Work

If we plug FFA into an optimization method, it removes the bias towards better solutions and instead prefers solutions whose objective values are encountered less frequently during the search. This can be viewed as putting selection pressure towards diversity. Viewed from this perspective, we find a long line of related works spanning several decades, starting with niching in evolutionary algorithms [24, 36, 44]. Like FFA, niching seeks diversity, but FFA achieves it via frequency tracking rather than explicit subpopulation partition.

Quality-Diversity (QD) methods are a family of algorithms that aim to generate a diverse set of high-performing solutions [9, 47, 48]. The QD method Novelty search (NS) selects solutions based on the divergence of their behavior from previous solutions [32]. The observed solution behaviors are retained in an archive as reference to judge behavior novelty [31].

Another QD method is Surprise Search (SS) [25, 26], which values solutions based on the deviation between their observed and predicted behaviors. The more different a solution behaves from what one expects, the more *surprising* it is, the more we can learn from exploring its neighborhood, and thus, the more likely it is for being investigated further. SS maintains a record of discovered solution behaviors to predict the behavior of new solutions.

MAP-Elites (ME) [39] is a QD algorithm that combines an objective function with a user-defined feature space $\mathbb{F}$. $\mathbb{F}$ is a low-dimensional space of behavioral descriptors. Each solution from the search space is mapped to a feature vector. The algorithm tries to find the highest-performing solutions within each grid cell in $\mathbb{F}$.

The algorithms mentioned above differ conceptually from FFA. All of them except niching are complete optimization algorithms. In contrast, FFA is a module that can be incorporated into a wide range of optimization algorithms. By making algorithms invariant under all injective transformations of the objective function value, including such that change the *order* of objective values, FFA offers the strongest theoretically attainable invariance property by any algorithm that actually uses the objective function value. None of the above methods has a similar property.

## 3 Studied Algorithms

To answer our research questions, we select eight algorithms. The baseline are the classical (1+1) EA and Randomized Local Search (RLS) [7]. The (1+1) EA (Algorithm 1) starts by randomly sampling an initial bit string $x_c$ from the search space $\{0,1\}^n$ and evaluating its fitness $y_c = f(x_c)$. At each iteration, a new solution $x_o$ is generated using a unary operator $\texttt{flip}_\ell(x_c)$ which creates a modified copy of $x_c$ where exactly $\ell$ bits are flipped. Here, $\ell \sim \text{Bin}_{>0}(n, 1/n)$, where $n$ is the problem dimension and $\text{Bin}_{>0}$ means that $\ell$ is drawn from a binomial distribution conditioned on $\ell > 0$ [7, 16]. This ensures that $x_o \neq x_c$ in each iteration. The objective value of $x_o$ is evaluated as $y_o \leftarrow f(x_o)$. If $y_o \leq y_c$, the current solution is replaced. This process is repeated until a termination condition is met.

The (1+1) FEA (Algorithm 2) introduces FFA into the (1+1) EA [58]. It allocates a frequency table $H$ that counts the encounter frequencies of objective values. The process of obtaining the current and

**procedure** (1+1) EA($f : \{0,1\}^n \mapsto \mathbb{N}_0$) **Algorithm 1**

randomly sample $x_c$ from $\{0,1\}^n$; $y_c \leftarrow f(x_c)$
**while** ¬ terminate **do**
  sample $\ell \leftarrow \text{Bin}_{>0}(n, 1/n)$
  $x_o \leftarrow \texttt{flip}_\ell(x_c)$; $y_o \leftarrow f(x_o)$
  **if** $y_o \leq y_c$ **then**
    $x_c \leftarrow x_o$; $y_c \leftarrow y_o$
**return** $x_c, y_c$

**procedure** (1+1) FEA($f : \{0,1\}^n \mapsto \mathbb{N}_0$) **Algorithm 2**

$H \leftarrow (0, 0, \ldots, 0)$
randomly sample $x_c$ from $\{0,1\}^n$; $y_c \leftarrow f(x_c)$
$x_b \leftarrow x_c$; $y_b \leftarrow y_c$
**while** ¬ terminate **do**
  sample $\ell \leftarrow \text{Bin}_{>0}(n, 1/n)$
  $x_o \leftarrow \texttt{flip}_\ell(x_c)$; $y_o \leftarrow f(x_o)$
  $H[y_c] \leftarrow H[y_c] + 1$; $H[y_o] \leftarrow H[y_o] + 1$
  **if** $H[y_o] \leq H[y_c]$ **then**
    $x_c \leftarrow x_o$; $y_c \leftarrow y_o$
    **if** $y_c < y_b$ **then**
      $x_b \leftarrow x_c$; $y_b \leftarrow y_c$
**return** $x_b, y_b$

new solutions $x_c$ and $x_o$ as well as their respective objective values $y_c$ and $y_o$ remains the same. The selection step is changed to first increment both $H[y_c]$ and $H[y_o]$ by 1. The new solution $x_o$ then replaces $x_c$ if $H[y_o] \leq H[y_c]$. These major differences are highlighted in blue. Since this may lead to the loss of the best-so-far solution, variables are used to remember it ($x_b$) and its objective value ($y_b$). They do not influence the search and are just returned as final result.

The RLS and FRLS work like the (1+1) EA and (1+1) FEA, respectively, but always flip exactly one bit, i.e., perform $x_o = \texttt{flip}_1(x_c)$ instead of $\texttt{flip}_\ell(x_c)$. This operator spans a neighborhood of exactly $n$ solutions around $x_c$, whereas the (1+1) EA can theoretically reach each solution from every other solution in a single step.

FFA-based search tends to be slow but eventually finds good solutions, whereas objective-guided search can be fast but may converge to local optima [58, 59]. It makes sense to combine both ideas by distributing objective function evaluations (FEs) evenly among them. The straightforward implementation of this hybrid algorithm concept would thus first perform one FE of the (1+1) EA, then one FE of the (1+1) FEA, then one of the (1+1) EA again, and so on. For toggling between the two methods, it could use a Boolean variable `useFFA` which would be negated after each FE. This hybrid algorithm would solve any problem that either of its component algorithms can solve (in twice the time of the faster component algorithm, that is). But it would not have any additional ability and be rather uninteresting.

EAFEA$_\text{A}$ and EAFEA$_\text{B}$ additionally introduce one-directional solution transfer from the (1+1) FEA branch to the (1+1) EA branch [33, 59]. If the (1+1) FEA branch in the EAFEA$_\text{A}$ (Algorithm 3) encounters a solution with previously unseen objective value, it will copy it to the (1+1) EA branch, thereby overwriting that branch's current solution [33]. If (1+1) FEA branch of the EAFEA$_\text{B}$ encounters

**procedure** EAFEA$_\text{A}$($f : \{0,1\}^n \mapsto \mathbb{N}_0$) **Algorithm 3**

$H \leftarrow (0, 0, \ldots, 0)$
randomly sample $x_c$ from $\{0,1\}^n$; $y_c \leftarrow f(x_c)$
$x_b \leftarrow x_c$; $y_b \leftarrow y_c$
$x_d \leftarrow x_c$; $y_d \leftarrow y_c$; `useFFA` $\leftarrow$ `false`
**while** ¬ terminate **do**
  **if** `useFFA` **then** $x_p \leftarrow x_d$ **else** $x_p \leftarrow x_c$
  sample $\ell \leftarrow \text{Bin}_{>0}(n, 1/n)$
  $x_o \leftarrow \text{flip}_\ell(x_p)$; $y_o \leftarrow f(x_o)$
  **if** $y_o < y_b$ **then** $x_b \leftarrow x_o$; $y_b \leftarrow y_o$
  **if** `useFFA` **then**
    $H[y_d] \leftarrow H[y_d] + 1$; $H[y_o] \leftarrow H[y_o] + 1$
    **if** $H[y_o] \leq H[y_d]$ **then**
      $x_d \leftarrow x_o$; $y_d \leftarrow y_o$
      **if** $H[y_o] = 1$ **then** $x_c \leftarrow x_o$; $y_c \leftarrow y_o$
  **else if** $y_o \leq y_c$ **then** $x_c \leftarrow x_o$; $y_c \leftarrow y_o$
  `useFFA` $\leftarrow$ ¬ `useFFA`
**return** $x_b, y_b$

**procedure** EAFEA$_\text{B}$($f : \{0,1\}^n \mapsto \mathbb{N}_0$) **Algorithm 4**

$H \leftarrow (0, 0, \ldots, 0)$
randomly sample $x_c$ from $\{0,1\}^n$; $y_c \leftarrow f(x_c)$
$x_d \leftarrow x_c$; $y_d \leftarrow y_c$; `useFFA` $\leftarrow$ `false`
**while** ¬ terminate **do**
  **if** `useFFA` **then** $x_p \leftarrow x_d$ **else** $x_p \leftarrow x_c$
  sample $\ell \leftarrow \text{Bin}_{>0}(n, 1/n)$
  $x_o \leftarrow \text{flip}_\ell(x_p)$; $y_o \leftarrow f(x_o)$
  **if** `useFFA` **then**
    $H[y_d] \leftarrow H[y_d] + 1$; $H[y_o] \leftarrow H[y_o] + 1$
    **if** $H[y_o] \leq H[y_d]$ **then** $x_d \leftarrow x_o$; $y_d \leftarrow y_o$
  **if** $y_o \leq y_c$ **then** $x_c \leftarrow x_o$; $y_c \leftarrow y_o$
  `useFFA` $\leftarrow$ ¬ `useFFA`
**return** $x_c, y_c$

a solution that has a better or equally good objective value than the current solution of the (1+1) EA branch, it will transfer it as well [59]. These transfers are highlighted in blue in the algorithms. We expect that the solution transfer will happen much more often in the EAFEA$_\text{A}$ than in the EAFEA$_\text{B}$. On the one hand, it takes a long time until the (1+1) FEA can overtake the (1+1) EA in solution quality. On the other hand, the (1+1) FEA tends to discover solutions of previously unseen quality more often.

Strictly overwriting the current solution of the (1+1) EA branch might not be the best of ideas, as it probably has some good building blocks. Therefore, we introduce two new hybrid algorithms: The EAFEA$_\text{AX}$ and EAFEA$_\text{BX}$ replace the copying procedure in the EAFEA$_\text{A}$ and EAFEA$_\text{B}$ with uniform crossover [14, 50]. For each element of the new solution it produces, Uniform Crossover randomly decides from which of the two parent solutions it should be copied. This new solution is then stored in $x_c$ and we set $y_c = f(x_c)$, therewith consuming 1 FE. We hypothesize that recombination enables that diverse solution structures from the (1+1) FEA branch merge with the good building blocks of the solution in the (1+1) EA branch.

$$\text{OneMax}(x) = n - \|x\|_1 \quad \text{where} \quad \|x\|_1 = \sum_{i=1}^{n} x[i] \tag{1}$$

$$\text{LeadingOnes}(x) = n - \sum_{i=1}^{n} \prod_{j=1}^{i} x[j] \tag{2}$$

$$\text{LinHarm}(x) = \frac{1}{2}n(n+1) - \sum_{i=1}^{n} i \cdot x[i] \tag{3}$$

$$\text{BinInt}(x) = 2^n - 1 - \sum_{i=1}^{n} 2^{i-1} \cdot x[i] \tag{4}$$

$$\text{Trap}(x) = \begin{cases} 0 & \text{if } \|x\|_1 = 0 \\ n - \|x\|_1 - 1 & \text{otherwise} \end{cases} \tag{5}$$

$$\text{TwoMax}(x) = \begin{cases} 0 & \text{if } \|x\|_1 = n \\ 1 + n - \max\left\{ \|x\|_1, n - \|x\|_1 \right\} & \text{otherwise} \end{cases} \tag{6}$$

$$\text{Jump}(x) = \begin{cases} n - \|x\|_1 & \text{if } (\|x\|_1 = n) \vee (\|x\|_1 \leq n - \mathrm{w}) \\ \mathrm{w} + \|x\|_1 & \text{otherwise} \end{cases} \tag{7}$$

$$\text{Plateau}(x) = \begin{cases} n - \|x\|_1 & \text{if } (\|x\|_1 = n) \vee (\|x\|_1 \leq n - \mathrm{w}) \\ \mathrm{w} & \text{otherwise} \end{cases} \tag{8}$$

$$\text{Ising}(x, T) = \sum_{(i,j) \in T} |x[i] - x[j]| \tag{9}$$

$$T_{1\mathrm{D}} = \{(i+1, (i+1 \bmod n) + 1) \mid i \in [0..n-1]\} \tag{10}$$

$$T_{2D} = \{(\alpha + \beta N, \gamma + \delta N) \mid \alpha, \beta, \gamma, \delta \in 0..N-1 \wedge [(\gamma = (\alpha+1) \bmod N \wedge \delta = \beta) \vee (\gamma = \alpha \wedge \delta = (\beta+1) \bmod N)]\} \text{ with } n = N^2 \tag{11}$$

$$\text{N-Queens}(x) = N - \|x\|_1 - N \sum_{\xi} \max\left\{0, Q_\xi(x) - 1\right\}; \quad n = N^2 \tag{12}$$

$$Q_\xi(x) = \text{number of queens in horizontal, vertical, or diagonal } \xi \tag{13}$$

$$\text{LABS}(x) = \sum_{k=1}^{n-1} \left( \sum_{i=0}^{n-1-k} s[i] \cdot s[i+k] \right)^2 \text{ with } s[j] = 2x[j] - 1 \tag{14}$$

**Figure 1: The discrete benchmark functions used in this study (with the exception of the W-Model).**

## 4 Benchmarks and Experiments

We now investigate the performance of our eight algorithms on several of the most well-researched non-$\mathcal{NP}$-hard standard benchmark functions for discrete optimization before applying them to the $\mathcal{NP}$-hard LABS problem. The objective functions are given in Figure 1. For each problem, we choose instances across a wide range of problem scales $n$, including powers of 2, 3, and 10, squares, prime numbers, etc. We deliberately mix structured sizes (powers and squares) with primes and other values to reduce the risk that conclusions are an artifact of arithmetic regularities (e.g., parity or factorization effects) that can influence neighborhood structure and objective value degeneracy on some benchmarks. For each instance-algorithm combination, we conduct at least 99 runs. Each run has a maximum computational budget of $10^8 = 100{,}000{,}000$ FEs.

We use the empirically estimated expected running time ERT [28] as performance metric. The ERT for a problem instance is estimated as the ratio of the sum of all FEs that all the runs consumed until they *either* have discovered a global optimum *or* exhausted their budget, divided by the number of runs that discovered a global optimum. The ERT is the mean expected runtime if we assume independent restarts after failed runs, which then may either succeed (consuming the mean runtime of the successful runs) or fail again (with the observed failure probability, after consuming $10^8$ FEs). If all runs succeed, it is the average first hitting time.

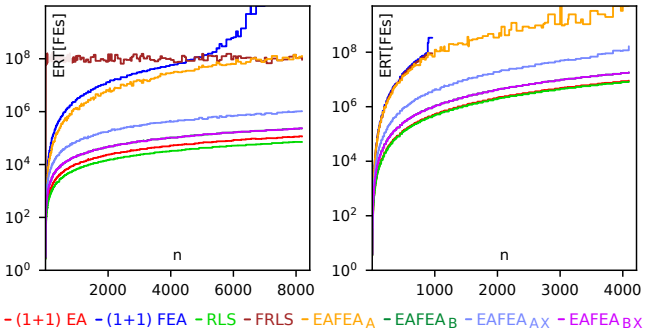



**Figure 2: The ERTs on OneMax (left) and LeadingOnes (right).**

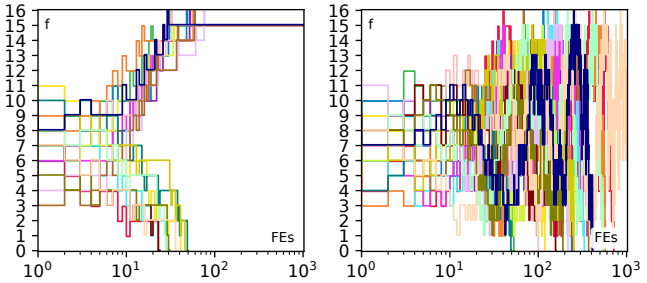



**Figure 3: Traces of 19 independent runs of the FRLS (left) and (1+1)FEA (right) on OneMax with $n = 16$.**

### 4.1 OneMax Problem

Solving an instance of the OneMax means to find the bit string composed of all 1s [19, 41]. This problem, given in Equation 1, is unimodal and fully separable. The left side of Figure 2 illustrates the ERT of different algorithms solving the OneMax problem. The horizontal axis represents the problem size $n$, i.e., the length of the bit string, and the vertical axis presents the ERT measured in FEs on a logarithmic scale. On this problem, it is obvious that (1+1)EA and RLS cannot be outperformed and RLS is better than the (1+1)EA [7].

The fully FFA-based algorithms perform worst. Since objective-guided local search on this unimodal problem has superior performance, the hybrid algorithms are better than the purely FFA-based ones. Among them, the EAFEA$_A$ is worst, as it most often transfers solutions from the FFA branch to the local search branch. Using uniform crossover instead of strictly copying in the EAFEA$_{AX}$ reduces its ERT by two orders of magnitude. The EAFEA$_B$ and EAFEA$_{BX}$ perform almost identical. Their (1+1)EA branch is better than the (1+1)FEA one, so solution transfer almost never takes place.

The FRLS has a strangely large ERT oscillating around the computational budget of $10^8$ FEs. It can solve the problem in about half of its runs, failing in the other half. The reason is that an

FFA-based method may progress either into the direction of improving or worsening objective values, changing direction from time to time [58]. Assume that FRLS happens to move towards the worst possible solution $x_\otimes$. It arrives at a solution $x_\times$ with objective value $f(x_\times) = n-1$. It will be coming from a solution with objective value $n-2$ and such solutions will probably have a slightly higher corresponding $H$-value than $x_\times$ at that point in time. It samples solutions in this neighborhood a few times, thereby increasing $H[n-1]$. Then maybe it samples $x_\otimes$, the bit string of all $\mathtt{0}$, with objective value $f(x_\otimes) = n$. It gets $H[n] = 1$ and is accepted.

From now on, all what the single-bit-flip operator $\mathtt{flip}_1(x_\otimes)$ can do is to take us back to a solution of objective value $n-1$. In the selection step, both $H[n]$ and $H[n-1]$ will be incremented, but $H[n]$ will be *lower*, because we had exactly the same situation just one step ago and it was lower there, too. So the algorithm cannot escape from $x_\otimes$. This is visible in Figure 3. Some of the traces of the FRLS runs on the OneMax problem with $n = 16$ keep sampling solutions of quality 15, which happens because they are stuck at the worst-possible solution with quality 16. They would need to flip two bits to escape, but cannot.

The following lemma formalizes the above observation and states a sufficient condition under which an FFA-based algorithm can become trapped.

**Lemma 1** (Absorbing states under constant-neighborhood objective values)**.** *Consider an FFA-based* (1+1) *algorithm whose selection step is as in Algorithm 2 and* FRLS*. In each iteration, it generates an offspring $x_o$ from the current solution $x_c$, has $y_c = f(x_c)$, evaluates $y_o = f(x_o)$, increments $H[y_c]$ and $H[y_o]$, and accepts $x_o$ iff $H[y_o] \leq H[y_c]$.*

*Let $x_\otimes \in \{0,1\}^n$ be a solution with objective value $y_\otimes = f(x_\otimes)$. Assume that, when the unary search operator is applied to $x_\otimes$,* all *possible offspring $x_\times$ have the same objective value $y_\times \neq y_\otimes$, i.e.,*

$$\forall x_\times \in \mathcal{N}(x_\otimes) : f(x_\times) = y_\times \neq y_\otimes, \tag{15}$$

*where $\mathcal{N}(x_\otimes)$ denotes the set of all possible offspring that can be generated from $x_\otimes$ by the unary operator used by the algorithm. If, at the beginning of an iteration with $x_c = x_\otimes$, we have $H[y_\times] > H[y_\otimes]$, then the algorithm will never leave $x_\otimes$ again, i.e., $x_\otimes$ becomes an absorbing state.*

Proof. While the algorithm remains at $x_\otimes$, every iteration has $y_c = y_\otimes$ and $y_o = y_\times$. Let $H_t[\cdot]$ denote the frequency table at the *beginning* of some iteration $t$ with $x_c = x_\otimes$. After incrementing frequencies, the acceptance test becomes

$$H_t[y_\times] + 1 \leq H_t[y_\otimes] + 1 \iff H_t[y_\times] \leq H_t[y_\otimes]. \tag{16}$$

If $H_t[y_\times] > H_t[y_\otimes]$ holds, the offspring is rejected and the current solution stays equal to $x_\otimes$.

After the rejection we have $H_{t+1}[y_\times] = H_t[y_\times]+1$ and $H_{t+1}[y_\otimes] = H_t[y_\otimes] + 1$. Therefore, the difference $H[y_\times] - H[y_\otimes]$ is invariant as long as the algorithm stays at $x_\otimes$. The strict inequality $H[y_\times] > H[y_\otimes]$ will hold forever and no future offspring can ever be accepted. □

**Corollary 2** (FRLS can be trapped on OneMax)**.** *For the minimization version of* OneMax*, let $x_\otimes = \mathtt{0}^n$ (all zeros), so that $f(x_\otimes) = n$. Under* FRLS*, the one-bit flip operator $\mathtt{flip}_1(x_\otimes)$ can only generate neighbors with objective value $n-1$. Consequently, whenever an* FRLS *run visits $x_\otimes$ at a time where $H[n-1] > H[n]$, it becomes trapped at $x_\otimes$ and can no longer reach the optimum $\mathtt{1}^n$.*

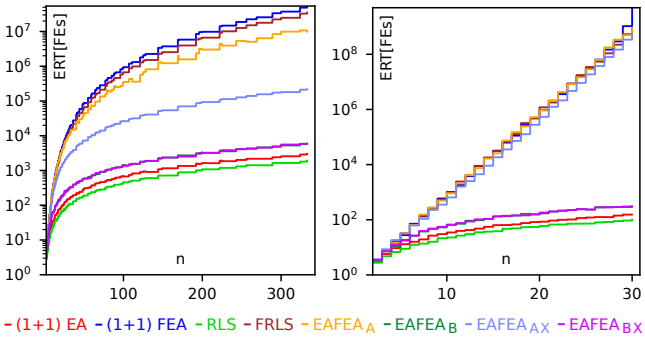



**Figure 4: The** ERT**s on the** LinHarm **(left) and** BinInt **(right).**

This explains the failed FRLS runs in Figure 3: once the run reaches $\mathtt{0}^n$ *and* the objective value $n-1$ is already more frequent than $n$, the selection rule prevents any accepted move away from $\mathtt{0}^n$. This seems to be a very small corner case, but it is an important finding. FFA based algorithms *can* get trapped, too!

## 4.2 LeadingOnes Problem

The LeadingOnes objective (Equation 2) aims to maximize the number of consecutive leading 1s [49, 60]. The problem is not separable but still unimodal and easy. Many local search methods require quadratic time for it [1, 17].

This is visible in the right side of Figure 2, where the RLS and (1+1)EA perform best and almost the same, as also observed in [7]. Prior experiments locate the (1+1)FEA median runtime around $O(n^3)$. Hence, somewhere between $n = 500$ and 1000, all runs of the (1+1)FEA fail and its ERT curve stops. The FRLS fares no better, but this time does not exhibit the pathological behavior from before, as flipping a single bit can now yield more than two different objective values.

$\text{EAFEA}_\text{A}$ has the same initial behavior as both algorithms, but its (1+1)EA strand allows it to find the optimum. The frequent solution transfer, however, causes it to be slow, too. The uniform crossover in the $\text{EAFEA}_\text{AX}$ significantly improves the speed. The $\text{EAFEA}_\text{B}$ and $\text{EAFEA}_\text{BX}$ perform better. Their (1+1)FEA branch rarely outperforms the local search, making solution transfer infrequent. Their FFA-based strand thus mainly wastes runtime.

## 4.3 Linear Harmonic Function

The Linear Harmonic problem LinHarm (Equation 3) is very similar to OneMax, but its variables are weighted linearly [15, 16, 18]. RLS and the (1+1)EA perform again best and the (1+1)FEA and FRLS are worst, as shown in the left part of Figure 4. The FRLS does not get stuck, because one FE can yield many different objective values and, as a result, it is a bit better than the (1+1)FEA. Apart from this, we see the same results as on OneMax.

## 4.4 BinInt Problem

The BinInt problem (Equation 4) considers the solutions $x$ as binary encoded integers representing the objective values [51] and

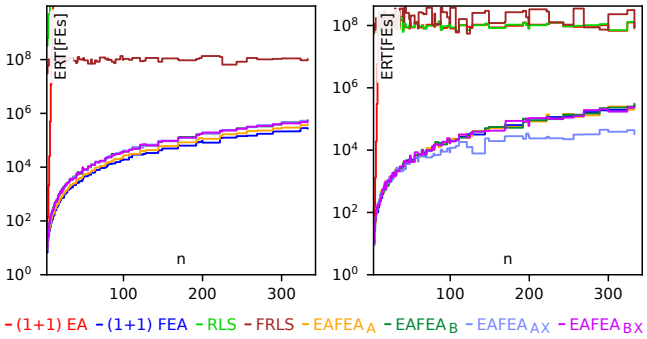



**Figure 5: The ERTs on Trap (left) and TwoMax (right).**

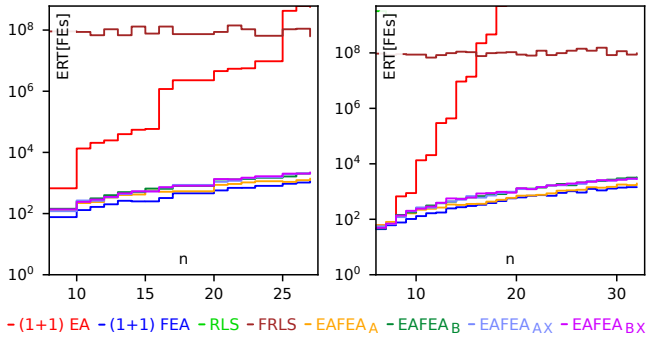



**Figure 6: The ERTs on the Jump problem with w = $\lfloor\sqrt{n}\rfloor+1$ (left) and w = $\lfloor n/2\rfloor - 1$ (right).**

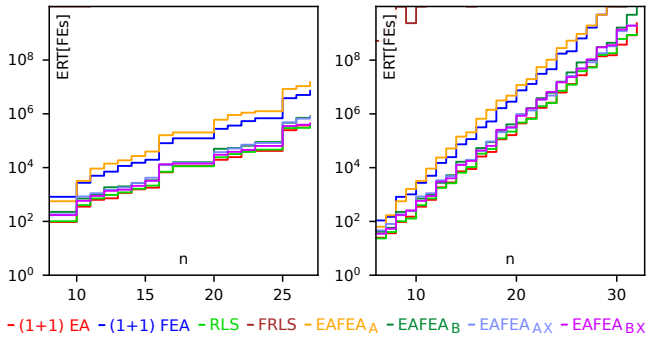



**Figure 7: The ERTs on the Plateau problem with w = $\lfloor\sqrt{n}\rfloor$ + 1 (left) and w = $\lfloor n/2\rfloor - 1$ (right).**

aims to find the string of all 1s. This is the worst possible case for FFA: Every single solution $x \in \{0,1\}^n$ has a different objective value, which makes the (1+1)FEA and FRLS degenerate to random walks. Their runtime grows exponentially, which is visible in the right part of Figure 4. Both EAFEA$_A$ variants will discover a solution $x_o$ with $H[f(x_o)] = 1$ in almost every step, thus constantly transfer $x_o$ to their (1+1)FEA branch, thus messing up any chance of this branch to find the optimum. They have exponential runtime, too. The EAFEA$_B$ variants, on the other hand, will basically never find any solution $x_o$ with $f(x_o) \leq f(x_c)$ and thus almost never engage in solution transfer.

## 4.5 Trap Problem

The Trap problem (Equation 5) is a deceptive function where the "gradient" in the entire search space points away from the global optimum [18, 42]. The optimum is a single bit string with no basin of attraction. This problem is the worst case for all objective-guided optimization methods. The (1+1)EA requires exponential runtime which shows in the left part of Figure 5. The RLS cannot escape from the local optimum, because that would require flipping all $n$ bits at once. Unless it samples the optimum or from its neighborhood at the beginning, it cannot solve this problem.

The Trap problem is an injective transformation of OneMax. Therefore, FRLS and the (1+1)FEA exhibit the same performance as on OneMax. The ERT of FRLS is still pathological due to the neighborhood structure. The ERT of the (1+1)FEA is now excellent. All the hybrid algorithms perform similarly. Now it is the FFA-guided search that does the heavy lifting, while the FEs spent in the (1+1)EA branch are wasted.

## 4.6 TwoMax Problem

The TwoMax problem (Equation 6) has a local and a global optimum of about the same size [22, 29]. Objective-guided algorithms ((1+1)EA, RLS) will most likely converge toward the optimum in whose basin of attraction their initial solution lands. The ERTs around the computational budget in the right part of Figure 5 confirm their 50/50 chance of success.

Since the TwoMax function is otherwise similar to OneMax, FRLS still exhibits its pathological behavior. All other algorithms that use FFA can solve this problem rather well. Interestingly, the EAFEA$_{AX}$ here has a visibly lower ERT than the other methods. The fact that it is roughly half an order of magnitude faster than the EAFEA$_A$ must be attributed to the uniform crossover, since this is the only difference between the two algorithms. This is strong evidence that combining solutions from both branches instead of overwriting them is a good idea.

## 4.7 Jump Problem

While the first part of the Jump function (Equation 7) has a gradient similar to OneMax leading toward the optimum, the last w bits right before the optimum have objective values that get successively worse the closer we get to the optimum [18, 22]. Figure 6 illustrates the ERTs of our algorithms on two different parameterizations of the problem. Small Jumps with a width of about $\sqrt{n}$ are shown on the left side. Large Jumps over half of the bits are shown on the right. The (1+1)EA requires a runtime exponential in w, because the algorithm needs to flip w bits at once to jump over this region [18]. Unless it samples the optimum or from its neighborhood right at the beginning, RLS cannot solve this problem. The Jump problem is a bijective transformation of OneMax, which means that (1+1)FEA and FRLS perform exactly as on that problem. The hybrid algorithms are a bit slower than the (1+1)FEA, as their objective-guided part does not contribute towards reaching the optimum.

## 4.8 Plateau Problem

The Plateau problem (Equation 8) is structured like Jump, but instead of being deceptive, the region of w bits around the optimum

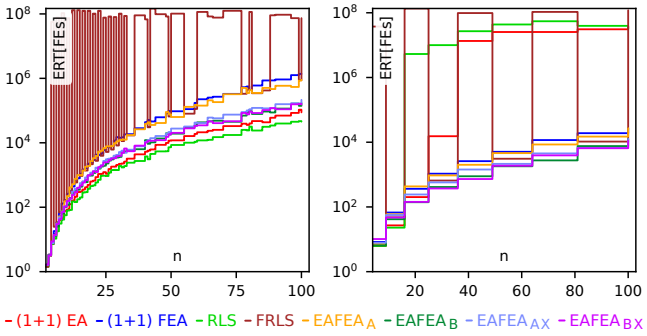



**Figure 8: The ERTs on the Ising-1D problem (left) and the Ising-2D problem (right).**

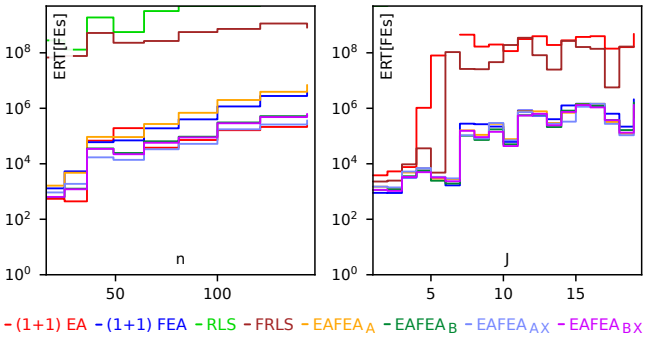



**Figure 9: The ERTs on the N-Queens problem (left) and each of the 19 W-Model problem instances $J$ from [53] (right).**

has the same objective value, i.e., is a neutral plateau [1]. Such plateaus are hard for all algorithms in our study. They neither provide a gradient to follow nor do they contain different objective values. Thus, all our algorithms require an exponential runtime, which the FFA-based ones paying the additional cost of the—here useless—search for diverse objective values.

## 4.9 The Ising Problems

The Ising problems (Equation 9) allow us to investigate how an algorithm performs if the decision variables exhibit localized epistasis. In this problem family, the contribution of one variable to the objective function depends on its neighbors in a topology $T$. Two neighboring bits of different values contribute +1 to the objective function, which is subject to minimization. The goal is that all bits have the same value. In Ising-1D [15, 16, 21], the bits are arranged in a 1-dimensional ring $T_{1D}$ (Equation 10) and in Ising-2D [15, 16, 20], the topology is the two-dimensional torus $T_{2D}$ (Equation 11).

The results of our algorithms on both problems are plotted in Figure 8. While RLS and the (1+1) EA can solve the Ising-1D problem relatively well, their runtime on the Ising-2D variant is much higher. The (1+1) FEA is faster than both algorithms on the Ising-2D but slower on Ising-1D. With the exception of the EAFEA$_A$, the hybrid algorithm variants are better than the (1+1) FEA. On Ising-2D, they are the best algorithms and on Ising-1D, they are slightly worse than the pure local searches.

The FRLS is slightly faster than the (1+1) FEA if the number $n$ of bits is odd and much slower if it is even, in which case it fails to solve the problem in roughly half of the runs. If the number $n$ of bits is odd, say $n = 11$, then the worst possible solution $x_\otimes$ —alternating 0 and 1-bits—has six 0 and five 1-bits (or the other way around). If such solution is reached, the neighborhood comprises two possible objective values: Either, we flip a 1-bit to 0, thereby improving the solution, or we flip a 0-bit to 1, which yields a solution of exactly the same objective value. If the number $n$ of bits is even, then all solutions neighboring $x_\otimes$ are better and none has the same objective value, meaning that the algorithm can get trapped by the mechanism of Lemma 1.

## 4.10 N-Queens Problem

The goal of the N-Queens Problem (Equation 12) is to place $N$ queens on a chessboard with side-length $N$ such that no queen can attack another [15, 16]. If the value of one of the $n = N^2$ fields of the chessboard is 1, then a queen is placed into it. The objective function is the sum of attack positions. N-Queens exhibits strong epistasis, because the contribution of each bit to the objective function depends on the states of the bits in all horizontal, vertical, and diagonals $\xi$ that could attack it. Neither RLS nor FRLS are good in this problem, as is visible in the left side of Figure 9. The (1+1) EA often outperforms the (1+1) FEA. The EAFEA$_{AX}$, which uses crossover, is much better than the EAFEA$_A$ and even often beats the (1+1) EA. Uniform crossover helps maybe because it preserves partial solutions (valid placements), whereas overwriting does not.

## 4.11 W-Model

The W-Model is a benchmark problem whose goal is to find a bit string consisting of alternating 0s and 1s [16, 57]. Filters are stacked on top of this base problem to introduce ruggedness, deceptiveness, epistasis, and uniform neutrality. The filters are tunable, so that precise amounts of these features can be introduced. In [53], 19 instances of the W-Model were chosen such that different algorithms are likely to exhibit different behavior on them.

As the right side of Figure 9 shows, due to their ruggedness and epistasis, RLS cannot solve a single W-Model instance. The (1+1) EA requires a high runtime for most instances and only solves them sometimes. All the algorithms using FFA are much better. On most instances, the hybrid algorithms have advantages. Crossover effects on W-Model instances are mixed. The question of which problem features determine crossover utility remains open.

## 4.12 Low-Autocorrelation Binary Sequences

The goal of the low-autocorrelation binary sequence problem (LABS) is to find a sequence whose autocorrelation is as low as possible. It is defined over vectors $s$ with $s[j] \in \{-1, 1\}$. We transform bit strings $x$ to such vectors via $s[j] = 2x[j] - 1$. The energy $f(x)$ of a string $x$ is the objective function (Equation 14).

The merit factor $F(x) = n^2/(nf(x))$ is consistent with $f$, but subject to maximization. Merit factors $F \geq 4$ always indicate a relatively low autocorrelation regardless of the problem scale $n$, whereas energy naturally always grows with $n$ and thus, there is no "good energy" threshold independent of $n$. We therefore plot the mean best merit factor discovered by the algorithms over the

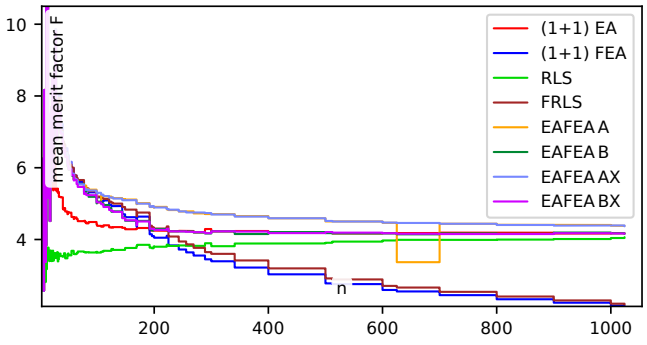



**Figure 10: The mean merit factor $F(x) = n^2/(nf(x))$ on the LABS problem, larger values are better.**

**Table 1: A set of best-known solutions (BKSes) discovered during our experiment vs. [2–6, 11–13, 23, 27, 30, 37, 38, 40, 43, 45, 46].**

| $n$ | algorithm | $f$, $F$ | $x$ in hexadecimal representation |
|---|---|---|---|
| 600 | $\text{EAFEA}_{\text{AX}}$ | 37392, 4.81386 | cb5ca407384e255e95e53e1fc06749a15298e5af65f3993cc8b0152748e76d8c11d6d3e611696ca75d050c9a233e54e0b3b72e48b20ad2020b3f10128031543e476eee0e2e217fe9fd141f |
| 625 | (1+1) EA | 41680, 4.68600 | 10f5be20caa1fffaad2bcfa97afcd2a0006ed5ed0c2fd1bd8e72cf2e8abe04d7d5f11798cb36063cde66ff4f626736c042de3294d368249e3882cf11968c847a2b570d89ad1dd666ccaf159852e0d |
| 700 | $\text{EAFEA}_{\text{BX}}$ | 52798, 4.64033 | 17f6ada4407fbeec022a90b15bf20ff16d9aaa0f6345617df958351c914cb3d63edf2348c760c3828c84994e5f5cfe98fb722f4db8fa53469b7132ed42a7e01c636268ebc721a19b5c60e2720948709c88f254a66324ac7 |
| 800 | (1+1) EA | 69104, 4.63070 | bb5fea73149af1a509ef9ecb7708284088e4c2494d6a18eedc6e4953d73b9087d67267a66e81631e2801eac987dce84be3a0a866af0701b006c13e0d1f1298786236bd0e724c03970b7daa766555523be66589fa8ea4937e472c0f4dbf64a6c584fb9651 |
| 900 | $\text{EAFEA}_{\text{A}}$ | 86830, 4.66429 | d5ffd1dddc6c9167e06dcdf9337f3ae95bbf997847067d7c0144a96b0b05105bf54d1892f7b1d31ba31ca37b3cae6f87838206114bf7eb5cba2cf03b4da2c383c6dc578d848b976b39c23a04518dbc472b88fd141ceb4ab6821df4a44ef25ec88a27691136edc07e43a721e129d6d28be |
| 1000 | $\text{EAFEA}_{\text{A}}$ | 108352, 4.61459 | c3498d656c2ed9987a231ef20c5b432943af5ac9ed460088caa2daacdd9ca9178662431894639b4f92f88f49c859147a5429a5e15f0d195ad025df26a54b8501199fa713433e49a7847fb0b61cc97a88543de9fcd23ce1d405dceec5157459af188a7ea8176fe8f0c05006de0b30001eb625155345ffe93d1e4fee1732 |
| 1024 | $\text{EAFEA}_{\text{AX}}$ | 114876, 4.56395 | e97368fe14ce608467d0ccc067c751e4e36954babd0ab127f8363909607a63967d3d2d8957378d9fa3a7c823b74a8f2e0422b7b73a742a526cca91c31f88e4561e6ef001a6f23a59ef85f7d62adaf24b80a2aea5975798befae4eee5560bef9ae93bf1cf5df03f0cb7e690784580409ef4ef12d391f51267ccedae7b6284889b |

problem scale $n$ in Figure 10.[1] The purely FFA-based (1+1)FEA and FRLS deliver worse results than the other methods. The large number of possible objective values slows them down too much, so that they cannot overtake the objective-guided search, which is why there hardly is any solution transfer in the $\text{EAFEA}_{\text{B}}$-type hybrids. However, after a scale $n$ of about 100, the best performing algorithms are the $\text{EAFEA}_{\text{A}}$ and $\text{EAFEA}_{\text{AX}}$. This is the first experiment on a bitstring-based $\mathcal{NP}$-hard problem where hybrid optimization combining FFA and objective-guided search can outperform both pure FFA- *and* objective-guided search. Whether using crossover or not does not make a large difference here in any hybrid algorithm.

With the 17 works [2–6, 11–13, 23, 27, 30, 37, 38, 40, 43, 45, 46] representing the state-of-the-art on LABS, we show in Table 1 the cases where we achieved new best known solutions. Despite using simple general-purpose operators and no domain-specific heuristics, we reached new best-known energies for these $n$.

[1]The downward dent for $\text{EAFEA}_{\text{A}}$ at $n = 625$ is caused by an unusually bad run.

## 5 Conclusions

In this paper, we have significantly expanded upon the previous studies on Frequency Fitness Assignment (FFA). FFA excels on (1) deceptive problems where objective-guided search stalls, (2) sparse objective landscapes with few distinct values, and (3) any objective function that can injectively be transformed to an easy function. We confirmed that (4) FFA is slow on unimodal problems like OneMax and (5) degenerates towards random walks on dense objective landscapes like BinInt. We *now know* that (6) FFA can stall in some rare situations, too, because we uncovered a sufficient condition under which FFA-based search can become trapped (Lemma 1): If, at some current solution $x_c$, the unary search operator can only generate offspring of a single objective value $y_o \neq f(x_c)$ and this value has already been encountered more frequently than the current objective value, i.e., $H[y_o] > H[f(x_c)]$, then the FFA selection rule will reject all offspring forever and the algorithm will not depart from $x_c$. This finding is significant, as we expected that FFA would keep exploring until the computational budget elapses.

Hybrid algorithms like the $\text{EAFEA}_{\text{A}}$ or $\text{EAFEA}_{\text{AX}}$ can be recommended as a safe default. They combine FFA's robustness on hard landscapes with the efficiency of local search on easy ones. We introduced a new hybridization method, namely the combination of the solutions in different branches via crossover. The FFA branch of these algorithms explores diverse objective values, including intermediate values between different optima. The objective-guided branch optimizes toward a local or global optimum. Crossover blends these and enables the objective-guided branch to benefit from FFA's broader exploration more efficiently. This led to an improved performance on many of the benchmark problems. The $\text{EAFEA}_{\text{AX}}$, for example, is several times faster than all other studied algorithms on the TwoMax problem.

We applied the algorithms of our study to the low-autocorrelation binary sequence problem (LABS). While previous works [58, 59] showed that FFA performs much better than objective-guided search on the $\mathcal{NP}$-hard Max-Sat problem, it does not do so on LABS. LABS is the first bitstring-based $\mathcal{NP}$-hard problem where hybrid algorithms, namely the $\text{EAFEA}_{\text{A}}$ and the $\text{EAFEA}_{\text{AX}}$, outperformed both purely objective-guided *and* FFA-guided local search. They yielded the best results on larger scales, some of them being BKSes.

Future work should: (1) develop stalling detection mechanisms, (2) provide landscape-based characterization predicting FFA utility, and (3) analyze hybrid variants theoretically.

## Acknowledgements

The authors acknowledge support from the project of National Natural Science Foundation of China 62406095, the Project of Natural Science Foundation of Anhui Province 2308085MF213, the Hefei Specially Recruited Foreign Expert program, and the Hefei Foreign Expert Office program. The authors disclose the use of a generative AI tool during the development of Lemma 1 and Corollary 2, including generating candidate statements and draft proof sketches in response to author prompts. The authors independently verified these results, revised the statements and proofs, and remain fully responsible for the originality, correctness, and integrity of all content in this paper.

## References


[1] Denis Antipov and Benjamin Doerr. 2018. Precise runtime analysis for plateaus. In *15th International Conference on Parallel Problem Solving from Nature (PPSN XV), Sept. 8-12, 2018, Coimbra, Portugal, Part II (Lecture Notes in Computer Science, Vol. 11102)*. Springer, Berlin/Heidelberg, Germany, 117–128. doi:10.1007/978-3-319-99259-4_10

[2] Borko Bošković and Janez Brest. 2018. A Heuristic Algorithm for a Low Autocorrelation Binary Sequence Problem with Odd Length and High Merit Factor. *IEEE Access* 6 (Jan. 2018), 4127–4134. doi:10.1109/ACCESS.2018.2789916

[3] Borko Bošković and Janez Brest. 2022. Computational Search of Long Skew-Symmetric Binary Sequences with High Merit Factors. *MENDEL – Soft Computing Journal* 28, 2 (Dec. 2022), 17–24. doi:10.13164/mendel.2022.2.017

[4] Borko Bošković, Franc Brglez, and Janez Brest. 2016. *A GitHub Archive for Solvers and Solutions of the* labs *Problem*. GitHub, San Francisco, CA, USA. https://github.com/borkob/git_labs

[5] Borko Bošković, Franc Brglez, and Janez Brest. 2017. Low-Autocorrelation Binary Sequences: On Improved Merit Factors and Runtime Predictions to Achieve Them. *Applied Soft Computing* 56 (July 2017), 262–285. doi:10.1016/J.ASOC.2017.02.024

[6] Borko Bošković, Jana Herzog, and Janez Brest. 2024. Parallel Self-Avoiding Walks for a Low-Autocorrelation Binary Sequences Problem. *Journal of Computational Science* 77 (2024), 102260. doi:10.1016/j.jocs.2024.102260

[7] Eduardo Carvalho Pinto and Carola Doerr. 2018. Towards a More Practice-Aware Runtime Analysis of Evolutionary Algorithms. *arXiv:1812.00493v1 [cs.NE] 3 Dec 2018* (2018), 30 pages. https://arxiv.org/abs/1812.00493

[8] Jiayang Chen, Zhize Wu, Sarah L. Thomson, and Thomas Weise. 2024. Frequency Fitness Assignment: Optimization Without Bias for Good Solution Outperforms Randomized Local Search on the Quadratic Assignment Problem. In *16th International Joint Conference on Computational Intelligence (IJCCI'2024), Nov. 20-22, 2024, Porto, Portugal*. SCITEPRESS, Porto, Portugal, 27–37. doi:10.5220/0012888600003837

[9] Antoine Cully and Yiannis Demiris. 2018. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation* 22, 2 (April 2018), 245–259. doi:10.1109/TEVC.2017.2704781

[10] Ege de Bruin, Sarah Louise Thomson, and Daan van den Berg. 2023. Frequency Fitness Assignment on JSSP: A Critical Review. In *26th European Conference on Applications of Evolutionary Computation (EvoApplications'2023), Held as Part of EvoStar'2023, April 12-14, 2023, Brno, Czech Republic*. Springer, Germany, Berlin/Heidelberg, 351–363. doi:10.1007/978-3-031-30229-9_23

[11] Miroslav Dimitrov. 2022. New Classes of Binary Sequences with High Merit Factor. *arXiv:2206.12070v1 [cs.IT] 24 Jun 2022* (2022), 22 pages. https://arxiv.org/pdf/2206.12070

[12] Miroslav Dimitrov, Tsonka Baitcheva, and Nikolay Nikolov. 2020. Efficient Generation of Low Autocorrelation Binary Sequences. *IEEE Signal Processing Letters* 27 (2020), 341–345. doi:10.1109/LSP.2020.2972127

[13] Miroslav M. Dimitrov. 2025. On the Skew-Symmetric Binary Sequences and the Merit Factor Problem. *Digital Signal Processing* 156, Part A (Jan. 2025), 104793. doi:10.1016/j.dsp.2024.104793

[14] Benjamin Doerr, Carola Doerr, and Franziska Ebel. 2015. From Black-Box Complexity to Designing New Genetic Algorithms. *Theoretical Computer Science* 567 (Feb. 2015), 87–104. doi:10.1016/j.tcs.2014.11.028

[15] Carola Doerr, Furong Ye, Naama Horesh, Hao Wang, Ofer M. Shir, and Thomas Bäck. 2019. Benchmarking Discrete Optimization Heuristics with IOHprofiler. In *Genetic and Evolutionary Computation Conference Companion (GECCO'2019) Companion, July 13-17, 2019, Prague, Czech Republic*. ACM, New York, NY, USA, 1798–1806. doi:10.1145/3319619.3326810

[16] Carola Doerr, Furong Ye, Naama Horesh, Hao Wang, Ofer M. Shir, and Thomas Bäck. 2020. Benchmarking discrete optimization heuristics with IOHprofiler. *Applied Soft Computing* 88 (March 2020), 106027. doi:10.1016/j.asoc.2019.106027

[17] Carola Doerr, Furong Ye, Sander van Rijn, Hao Wang, and Thomas Bäck. 2018. Towards a Theory-Guided Benchmarking Suite for Discrete Black-Box Optimization Heuristics: Profiling $(1 + \lambda)$ EA Variants on OneMax and LeadingOnes. In *Genetic and Evolutionary Computation Conference Companion (GECCO'2018), July 15-19, 2018, Kyoto, Japan*. ACM, New York, NY, USA, 951–958. doi:10.1145/3205455.3205621

[18] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2002. On the analysis of the $(1 + 1)$ Evolutionary Algorithm. *Theoretical Computer Science* 276, 1-2 (April 2002), 51–81. doi:10.1016/S0304-3975(01)00182-7

[19] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2006. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems* 39, 4 (July 2006), 525–544. doi:10.1007/S00224-004-1177-Z

[20] Simon Fischer. 2004. A Polynomial Upper Bound for a Mutation-Based Algorithm on the Two-Dimensional Ising Model. In *Genetic and Evolutionary Computation Conference (GECCO'2004), June 26-30, 2004, Seattle, WA, USA, Part I*. Springer, Berlin/Heidelberg, Germany, 1100–1112. doi:10.1007/978-3-540-24854-5_108

[21] Simon Fischer and Ingo Wegener. 2005. The one-dimensional Ising model: Mutation versus recombination. *Theoretical Computer Science* 344, 2-3 (Nov. 2005), 208–225. doi:10.1016/J.TCS.2005.04.002

[22] Tobias Friedrich, Francesco Quinzan, and Markus Wagner. 2018. Escaping Large Deceptive Basins of Attraction with Heavy-Tailed Mutation Operators. In *Genetic and Evolutionary Computation Conference Companion (GECCO'2018), July 15-19, 2018, Kyoto, Japan*. ACM, New York, NY, USA, 293–300. doi:10.1145/3205455.3205515

[23] José E. Gallardo, Carlos Cotta, and Antonio J. Fernández. 2007. A Memetic Algorithm for the Low Autocorrelation Binary Sequence Problem. In *Genetic and Evolutionary Computation Conference (GECCO'2007), July 7-11, 2007, London, England, UK*. ACM, New York, NY, USA, 1226–1233. doi:10.1145/1276958.1277195

[24] David Edward Goldberg and Jon Richardson. 1987. Genetic algorithms with sharing for multimodal function optimization. In *2nd International Conference on Genetic Algorithms on Genetic Algorithms and Their Application, July 28-31, 1987, Cambridge, Massachusetts, USA*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 41–49. ISBN: 978-0-8058-0158-3.

[25] Daniele Gravina, Antonios Liapis, and Georgios Yannakakis. 2016. Surprise Search: Beyond Objectives and Novelty. In *Genetic and Evolutionary Computation Conference (GECCO'2016), July 20-24, 2016, Denver, CO, USA*. ACM, New Yor, NY, USA, 677–684. doi:10.1145/2908812.2908817

[26] Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. 2019. Quality diversity through surprise. *IEEE Transactions on Evolutionary Computation* 23, 4 (Aug. 2019), 603–616. doi:10.1109/TEVC.2018.2877215

[27] Steven Halim, Roland H. C. Yap, and Felix Halim. 2008. Engineering Stochastic Local Search for the Low Autocorrelation Binary Sequence Problem. In *14th International Conference on Principles and Practice of Constraint Programming (CP'2008), Sept. 14-18, 2008, Sydney, Australia (Lecture Notes in Computer Science (LNPSE), Vol. 5202)*. Springer, Heidelberg/Berlin, 640–645. doi:10.1007/978-3-540-85958-1_57 ISBN: 978-3-540-85957-4.

[28] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. 2021. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *Optimization Methods and Software* 36 (2021), 114–144. Issue 1. doi:10.1080/10556788.2020.1808977 See also arXiv:1603.08785v4 [cs.AI] 9 Sep 2020.

[29] Clarissa Van Hoyweghen, David E. Goldberg, and Bart Naudts. 2002. From TwoMax to the Ising model: easy and hard symmetrical problems. In *4th Annual Conference on Genetic and Evolutionary Computation (GECCO'2002), July 9-13, 2002, New York City, NY USA*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 626–633. ISBN: 1558608788.

[30] Joshua Knauer. Oct. 8, 2004. *Merit Factor Records*. Michigan State University, East Lansing, MI, USA. Edited by Marcos Dantus. https://www2.chemistry.msu.edu/faculty/dantus/merit_factor_records.htm. A copy is contained in [4].

[31] Joel Lehman and Kenneth Owen Stanley. 2008. Exploiting Open-Endedness to Solve Problems through the Search for Novelty. In *11th International Conference on the Synthesis and Simulation of Living Systems (ALIFE'2008), Aug. 5-8, 2008, Winchester, UK*. MIT Press, Cambridge, MA, USA, 329–336. http://mitpress2.mit.edu/books/chapters/0262287196chap43.pdf ISBN: 9780262287197.

[32] Joel Lehman and Kenneth Owen Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation* 19, 2 (Summer 2011), 189–223. doi:10.1162/EVCO_A_00025

[33] Tianyu Liang, Zhize Wu, Jörg Lässig, Daan van den Berg, Sarah L. Thomson, and Thomas Weise. 2024. Addressing the traveling salesperson problem with frequency fitness assignment and hybrid algorithms. *Soft Computing* 28, 17 (July 2024), 9495–9508. doi:10.1007/S00500-024-09718-8

[34] Tianyu Liang, Zhize Wu, Jörg Lässig, Daan van den Berg, and Thomas Weise. 2022. Solving the Traveling Salesperson Problem using Frequency Fitness Assignment. In *IEEE Symposium on Foundations of Computational Intelligence (IEEE FOCI'2022), part of the IEEE Symposium Series on Computational Intelligence (SSCI'2022), Dec. 4–7, 2022, Singapore*. IEEE, Piscataway, NJ, USA, 8 pages. doi:10.1109/SSCI51031.2022.10022296 ISBN: 978-1-6654-8768-9.

[35] Tianyu Liang, Zhize Wu, Matthias Thürer, Markus Wagner, and Thomas Weise. 2024. Generating Small Instances with Interesting Features for the Traveling Salesperson Problem. In *16th International Joint Conference on Computational Intelligence (IJCCI'2024), Nov. 20-22, 2024, Porto, Portugal*. SCITEPRESS, Porto, Portugal, 173–180. doi:10.5220/0012888800003837

[36] Yves Matanga, Pius Owolawi, Chunling Du, and Etienne van Wyk. 2024. Niching Global Optimisation: Systematic Literature Review. *Algorithms* 17, 10 (Oct. 2024), 448. doi:10.3390/A17100448

[37] Stephan Mertens. 1996. Exhaustive search for low-autocorrelation binary sequences. *Journal of Physics A: Mathematical and General* 29, 18 (Sept. 1996), L473. doi:10.1088/0305-4470/29/18/00

[38] Burkhard Militzer, Michele Zamparelli, and Dieter Beule. 1998. Evolutionary Search for Low Autocorrelated Binary Sequences. *IEEE Transactions on Evolutionary Computation* 2, 1 (April 1998), 34–39. doi:10.1109/4235.728212

[39] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating Search Spaces by Mapping Elites. *arXiv:1504.04909v1 [cs.AI] 20 Apr 2015* (2015), 15 pages. https://arxiv.org/abs/1504.04909

[40] Wai Ho Mow and Ke-Lin Du. 2015. New Evolutionary Search for Long Low Autocorrelation Binary Sequences. *IEEE Trans. Aerospace Electron. Systems* 51, 1

(Jan. 2015), 290–303. doi:10.1109/TAES.2014.130518

[41] Heinz Mühlenbein. 1992. How Genetic Algorithms Really Work: I. Mutation and Hillclimbing. In *2nd International Conference on Parallel Problem Solving from Nature (PPSN 1992), Sept. 28-30, 1992, Brussels, Belgium*. Elsevier, Amsterdam, The Netherlands, 15–26.

[42] Siegfried Nijssen and Thomas Back. 2003. An analysis of the behavior of simplified evolutionary algorithms on trap functions. *IEEE Transactions on Evolutionary Computation* 7, 1 (Feb. 2003), 11–22. doi:10.1109/TEVC.2002.806169

[43] Tom Packebusch and Stephan Mertens. 2016. Low Autocorrelation Binary Sequences. *Journal of Physics A: Mathematical and Theoretical* 49, 16 (March 2016), 165001. doi:10.1088/1751-8113/49/16/165001

[44] Alain Pétrowski. 1996. A Clearing Procedure as a Niching Method for Genetic Algorithms. In *IEEE International Conference on Evolutionary Computation (ICEC'1996), May 20-22, 1996, Nagoya, Japan*. IEEE, Piscataway, NJ, USA, 798–803. doi:10.1109/ICEC.1996.542703

[45] S. D. Prestwich. 2013. Improved Branch-and-Bound for Low Autocorrelation Binary Sequences. *arXiv:1305.6187v2 [cs.AI] 23 Jul 2013* (2013), 9 pages. https://arxiv.org/pdf/1305.6187

[46] Blaž Pšeničnik, Rene Mlinarič, Janez Brest, and Borko Bošković. 2025. Dual-Step Optimization for Binary Sequences with High Merit Factors. *Digital Signal Processing* 165, 1053 (Oct. 2025), 9 pages. doi:10.1016/j.dsp.2025.105316

[47] Justin K. Pugh, Lisa B. Soros, and Kenneth Owen Stanley. 2016. Quality Diversity: A New Frontier for Evolutionary Computation. *Frontiers in Robotics and AI* 3 (July 2016), 40. doi:10.3389/FROBT.2016.00040

[48] Chao Qian, Ke Xue, and Ren-Jian Wang. 2024. Quality-Diversity Algorithms Can Provably Be Helpful for Optimization. In *33rd International Joint Conference on Artificial Intelligence (IJCAI'2024), Aug. 3-9, 2024, Jeju, South Korea*. IJCAI, San Francisco, CA, USA, 6994–7002. doi:10.24963/ijcai.2024/773 ISBN: 978-1-956792-04-1.

[49] Günter Rudolph. 1997. *Convergence properties of evolutionary algorithms*. Verlag Dr. Kovač, Hamburg, Germany. ISBN: 3860645544.

[50] William M. Spears and Kenneth Alan De Jong. 1991. On the Virtues of Parameterized Uniform Crossover. In *4th International Conference on Genetic Algorithms (ICGA'91), July 13-16, 1991, San Diego, CA, USA*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 230–236. https://www.mli.gmu.edu/papers/91-95/91-18.pdf ISBN: 1-55860-208-9.

[51] Dirk Thierens, David Edward Goldberg, and Angela Guimarães Pereira. 1998. Domino Convergence, Drift, and the Temporal-Salience Structure of Problems. In *1998 IEEE International Conference on Evolutionary Computation (ICEC'1998), May 4-9, 1998, Anchorage, AK, USA*. IEEE, Piscataway, NJ, USA, 535–540. doi:10.1109/ICEC.1998.700085

[52] Sarah Louise Thomson, Gabriela Ochoa, Daan van den Berg, Tianyu Liang, and Thomas Weise. 2024. Entropy, Search Trajectories, and Explainability for Frequency Fitness Assignment. In *18th International Conference on Parallel Problem Solving From Nature (PPSN XVIII), Sept. 14-18, 2024, Hagenberg, Austria (Lecture Notes in Computer Science, Vol. 15148)*. Springer, Germany, Berlin/Heidelberg, 377–392. doi:10.1007/978-3-031-70055-2_23

[53] Thomas Weise, Yan Chen, Xinlu Li, and Zhize Wu. 2019. Selecting a Diverse Set of Benchmark Instances from a Tunable Model Problem for Black-Box Discrete Optimization Algorithms. *Applied Soft Computing* 92 (July 2019), 106269. doi:10.1016/j.asoc.2020.106269

[54] Thomas Weise, Xinlu Li, Yan Chen, and Zhize Wu. 2021. Solving Job Shop Scheduling Problems Without Using a Bias for Good Solutions. In *Genetic and Evolutionary Computation Conference Companion (GECCO'21) Companion, July 10-14, 2021, Lille, France*. ACM, New York, NY, USA, 1459–1466. doi:10.1145/3449726.3463124 ISBN: 978-1-4503-8351-6.

[55] Thomas Weise, Mingxu Wan, Ke Tang, and Xin Yao. 2014. Evolving Exact Integer Algorithms with Genetic Programming. In *IEEE Congress on Evolutionary Computation (CEC'2014), July 6-11, 2014, Beijing, China*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1816–1823. doi:10.1109/CEC.2014.6900292 ISBN: 978-1-4799-1488-3.

[56] Thomas Weise, Mingxu Wan, Pu Wang, Ke Tang, Alexandre Devert, and Xin Yao. 2014. Frequency Fitness Assignment. *IEEE Transactions on Evolutionary Computation* 18, 2 (2014), 226–243. doi:10.1109/TEVC.2013.2251885

[57] Thomas Weise and Zijun Wu. 2018. Difficult Features of Combinatorial Optimization Problems and the Tunable W-Model Benchmark Problem for Simulating Them. In *Genetic and Evolutionary Computation Conference Companion (GECCO'2018), July 15-19, 2018, Kyoto, Japan*. ACM, New York, NY, USA, 1769–1776. doi:10.1145/3205651.3208240

[58] Thomas Weise, Zhize Wu, Xinlu Li, and Yan Chen. 2021. Frequency fitness assignment: Making optimization algorithms invariant under bijective transformations of the objective function value. *IEEE Transactions on Evolutionary Computation* 25, 2 (April 2021), 307–319. doi:10.1109/TEVC.2020.3032090

[59] Thomas Weise, Zhize Wu, Xinlu Li, Yan Chen, and Jörg Lässig. 2023. Frequency fitness assignment: Optimization without bias for good solutions can be efficient. *IEEE Transactions on Evolutionary Computation* 27, 4 (Aug. 2023), 980–992. doi:10.1109/TEVC.2022.3191698

[60] L. Darrell Whitley. 1989. The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials Is Best. In *3rd International Conference on Genetic Algorithms, June 1989, Fairfax, VA, USA*. Morgan Kaufmann Publishers Inc., Burlington, MA, USA, 116–123.