Self-adaptation via Multi-objectivisation: An Empirical Study

Xiaoyu Qin ✉[1] [0000−0002−9720−3220] and Per Kristian Lehre ✉[1] [0000−0002−9521−1251]

University of Birmingham, Birmingham B15 2TT, United Kingdom
{xxq896, p.k.lehre}@cs.bham.ac.uk

Abstract. Non-elitist evolutionary algorithms (EAs) can be beneficial in optimisation of noisy and or rugged fitness landscapes. However, this benefit can only be realised if the parameters of the non-elitist EAs are carefully adjusted in accordance with the fitness function. Self-adaptation is a promising parameter adaptation method that encodes and evolves parameters in the chromosome. Existing self-adaptive EAs often sort the population by first preferring higher fitness and then the mutation rate. A previous study (Case and Lehre, 2020) proved that self-adaptation can be effective in certain discrete problems with unknown structure. However, the population can be trapped on local optima, because individuals in "dense" fitness valleys which survive high mutation rates and individuals on "sparse" local optima which only survive with lower mutation rates cannot be simultaneously preserved.

Recently, the Multi-Objective Self-Adaptive EA (MOSA-EA) (Lehre and Qin, 2022) was proposed to optimise single-objective functions, treating parameter control via multi-objectivisation. The algorithm maximises the fitness and the mutation rates simultaneously, allowing individuals in "dense" fitness valleys and on "sparse" local optima to co-exist on a non-dominated Pareto front. The previous study proved its efficiency in escaping local optima with unknown sparsity, where some fixed mutation rate EAs become trapped. However, the performance is unknown in other settings.

This paper continues the study of MOSA-EA through an empirical study. We find that the MOSA-EA has a comparable performance on unimodal functions, and outperforms eleven randomised search heuristics considered on a bi-modal function with "sparse" local optima. For NP-hard problems, the MOSA-EA increasingly outperforms other algorithms for harder NK-Landscape and k-Sat instances. Notably, the MOSA-EA outperforms a problem-specific MaxSat solver on several hard k-Sat instances. Finally, we show that the MOSA-EA self-adapts the mutation rate to the noise level in noisy optimisation. The results suggest that self-adaptation via multi-objectivisation can be adopted to control parameters in non-elitist EAs.

Keywords: Evolutionary algorithms · Self-adaptation · Local optima · Combinatorial optimisation · Noisy optimisation.

1 Introduction

Non-elitism is widely adopted in continuous EAs, and has recently shown to be promising also for combinatorial optimisation. Several runtime analyses have shown that non-elitist EAs can escape certain local optima efficiently [9,10] and can be robust to noise [11,32]. There exist a few theoretical results to investigate how non-elitist EAs can cope with local optima [9,10,14]. SparseLocalOpt [10] is a tunable problem class that describes a kind of fitness landscapes with *sparse deceptive regions* (local optima) and *dense fitness valleys*. Informally, search points in a dense fitness valley have many Hamming neighbours in the fitness valley, while search points in sparse deceptive regions have few neighbours within the deceptive region. Non-elitist EAs with *non-linear selection mechanisms* are proven to cope with sparse local optima [9, 10]. In non-linear selection mechanisms [30, 34], the selection probability is decreasing with respect to the rank of the individual, e.g., tournament and linear ranking selection [22]. The fitter individual has a higher probability to be selected, but worse individuals still have some chance to be selected. Thus, while individuals on sparse local optimum have higher chance of being selected, fewer of their offspring stay on the peak under a sufficiently high mutation rate. In contrast, even if individuals in dense fitness valley individuals have less chance of being selected, a larger fraction of their offspring stay within the fitness valley. However, it is critical to set the "right" mutation rate, which should be sufficiently high but below the *error threshold*. Non-elitist EAs with mutation rate above the error threshold will "fail" to find optima in expected polynomial time, assuming the number of global optima is polynomially bounded [29]. Finding the appropriate mutation rate which allows the algorithm to escape not too sparse local optima is non-trivial. In noisy environments, non-elitist EAs using the "right" mutation rate beat the current state of the art results for elitist EAs [32] on several settings of problems and noise models. However, the "right" mutation rate depends on the noise level, which is usually unknown in real-world optimisation.

Self-adaptation is a promising method to automate parameter configuration [4, 36, 41]. It encodes the parameters in the chromosome of each individual, thus the parameters are subject to variation and selection. Some self-adaptive EAs are proven efficient on certain problems, e.g., the OneMax function [17], the simple artificial two-peak PeakedLO function [12] and the unknown structure version of LeadingOnes [7]. These self-adaptive EAs sort the population by preferring higher fitness and then consider the mutation rate. They might be trapped in sparse local optima, because individuals in dense fitness valleys which survive high mutation rates and individuals on sparse local optima which only survive with lower mutation rates cannot be simultaneously preserved.

Recently, a new self-adaptive EA, the *multi-objective self-adaptive EA* (MOSA-EA) [33], was proposed to optimise single-objective functions, which treats parameter control from multi-objectivisation. The algorithm maximises the fitness and the mutation rates simultaneously, allowing individuals in dense fitness valleys and on sparse local optima to co-exist on a non-dominated Pareto front. The previous study showed its efficiency in escaping a local optimum with unknown sparsity, where some fixed mutation rate EAs including non-linear selection EAs become trapped. However, it is unclear whether the benefit of the MOSA-EA can also be observed for more complex problems, such as NP-hard combinatorial optimisation problems and noisy fitness functions.

This paper continues the study of MOSA-EA through an empirical study of its performance on selected combinatorial optimisation problems. We find that the MOSA-EA not only has a comparable performance on unimodal functions, e.g., OneMax and LeadingOnes, but also outperforms eleven randomised search heuristics considered on a bi-modal function with a sparse local optimum, i.e., Funnel. For NP-hard combinatorial optimisation problems, the MOSA-EA increasingly outperforms other algorithms for harder NK-Landscape and k-Sat instances. In particular, the MOSA-EA outperforms a problem-specific MaxSat solver on some hard k-Sat instances. Finally, we demonstrate that the MOSA-EA can self-adapt the mutation rate to the noise level in noisy optimisation.

2 Multi-Objective Self-Adaptive EA (MOSA-EA)

This section introduces a general framework for self-adaptive EAs (Alg. 1), then defines the algorithm MOSA-EA as a special case of this framework by specifying a self-adapting mutation rate strategy (Alg. 2).

Algorithm 1 Framework for self-adaptive EAs [33]

```
Require: Fitness function $f : \{0,1\}^n \to \mathbb{R}$. Population size $\lambda \in \mathbb{N}$. Sorting mechanism
    Sort. Selection mechanism $P_{\text{sel}}$. Self-adapting mutation rate strategy $D_{\text{mut}}$. Initial
    population $P_0 \in \mathcal{Y}^\lambda$.
1: for $t$ in $0, 1, 2, \ldots$ until termination condition met do
2:     Sort($P_t, f$)
3:     for $i = 1, \ldots, \lambda$ do
4:         Sample $I_t(i) \sim P_{sel}([\lambda])$; Set $(x, \chi/n) := P_t(I_t(i))$.
5:         Sample $\chi' \sim D_{\text{mut}}(\chi)$.
6:         Create $x'$ by independently flipping each bit of $x$ with probability $\chi'/n$.
7:         Set $P_{t+1}(i) := (x', \chi'/n)$.
```

Let $f : \mathcal{X} \to \mathbb{R}$ be any pseudo-Boolean function, where $\mathcal{X} = \{0,1\}^n$ is the set of bitstrings of length n. For self-adaptation in non-elitist EAs, existing algorithms [7,12,17] optimise f on an extended search space $\mathcal{Y} := \mathcal{X} \times [\varepsilon, 1/2]$ which includes $\mathcal{X}$ and an interval of mutation rates. Alg. 1 [33] shows a framework for self-adaptive EAs. In each generation t, they first sort the population P_t using an order that depends on the fitness and the mutation rate. To sort the population, existing self-adaptive EAs [7,12,17] often prefer higher fitness and then consider the mutation rate. Then, each individual in the next population P_{t+1} is produced via selection and mutation. The selection mechanism is defined in terms of the ranks of the individuals in the sorted population. Then, the selected individual changes its mutation rate based on a self-adapting mutation rate strategy and each bit is flipped with the probability of the new mutation rate. For example, the *fitness-first sorting mechanism* in the self-adaptive (μ, λ) EA (SA-EA) [7] ensures that a higher fitness individual is ranked strictly higher than a lower fitness individual as illustrated in Fig. 1(a).

Different from the existing algorithms, the MOSA-EA sorts the population by the *multi-objective sorting mechanism* (Alg. 3 in Appendix 1). It first applies the *strict non-dominated sorting method* (Alg. 4 in Appendix 1) to divide the population into several Pareto fronts, then sorts the population based on the ranks of the fronts and then the fitness values. Unlike the non-dominated fronts used in multi-objective EAs, e.g., NSGA-II [13, 42], each strict non-dominated front only contains a limited number of individuals, i.e., no pair of individuals have the same objective values. Alg. 5 in Appendix 1 shows an alternative way to do multi-objective sorting. Fig. 1(b) [33] illustrates an example of the order of a population after multi-objective sorting.

(a) Fitness-first sorting (SA-EA) (b) Multi-objective sorting (MOSA-EA)

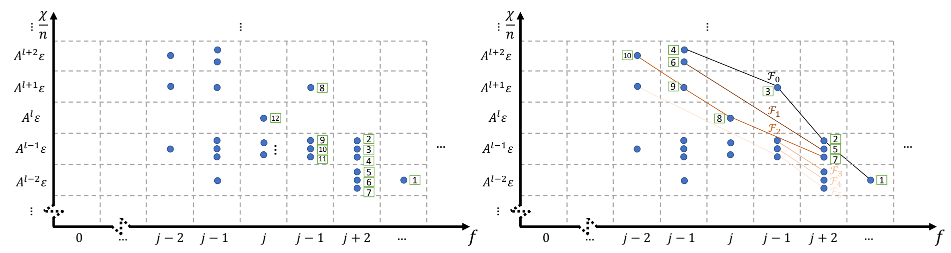

Fig. 1: Illustration of population sorting in (a) SA-EA and (b) MOSA-EA [33]. The points in the same cell have the same fitness and the same mutation rate.

In this paper, we consider comma selection (Alg. 6 in Appendix 1) which selects parents from the fittest μ individuals of the sorted population uniformly at random. To self-adapt the mutation rate, we apply the same strategy as in [33] (Alg. 2), where the mutation rate is multiplied by $A > 1$ with probability $p_{\text{inc}} \in (0, 1)$, otherwise it is divided by A. The range of mutation rates is from $\varepsilon > 0$ to $1/2$. We sample the initial mutation rate of each individual from $\{\varepsilon A^i \mid i \in [0, \lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor]\}$ uniformly at random, where i is an integer.

Most programming languages represent floats imprecisely, e.g., multiplying a number by A, and then dividing by A is not guaranteed to produce the original number. In the MOSA-EA, it is critical to use precise mutation rates to limit the number of distinct mutation rates in the population. We therefore recommend to implement the self-adaptation strategy as follows: build an indexed list containing all mutation rates $\chi/n = \varepsilon A^i$ for all integers $i \in [0, \lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor]$, and encode the index into each individual. Then mutating the mutation rate can be achieved by adding or subtracting 1 to the index instead of the mutation rate multiplying A or dividing by A.

Algorithm 2 Self-adapting mutation rate strategy [33]

```
Require: Parameters $A > 1$, $\varepsilon > 0$ and $p_{\text{inc}} \in (0,1)$. Mutation parameter $\chi$.
1: $\chi' = \begin{cases} \min(A\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor}) & \text{with probability } p_{\text{inc}}, \\ \max(\chi/A, \varepsilon n) & \text{otherwise.} \end{cases}$
2: return $\chi'$.
```

3 Parameter settings in MOSA-EA

One of the aims of self-adaptation is to reduce the number of parameters that must be set by the user. MOSA-EA has three parameters ε, p_{inc} and A, in addition to the population sizes λ and μ. We will first investigate how sensitive the algorithm is to these parameters. Adding three new parameters to adapt one parameter seems contradictory to the aim of self-adaptation. However, as we will see later in this paper, these parameters need not to be tuned carefully. We use the same parameter setting of the MOSA-EA for all experiments in this paper to show that the MOSA-EA does not require problem-specific tuning of the parameters.

The parameter ε is the lower bound of the mutation rate in the MOSA-EA. In fixed mutation rate EAs, we usually set a constant mutation parameter χ. To cover the range of all possible mutation rates χ/n, we recommend to set the lowest mutation rate $\varepsilon = c/(n\ln(n))$, where c is some small constant. In this paper, we set $\varepsilon = 1/(2n\ln(n))$. As mentioned before, $A > 0$ and $p_{\text{inc}} \in (0, 1)$ are two self-adapting mutation rate parameters in Alg. 2. We use some simple functions as a starting point to empirically analyse the effect of setting the parameters of A and p_{inc}. We run the MOSA-EA with different parameters A and p_{inc} on OneMax, LeadingOnes and Funnel ($n = 100$, the definitions can be found in Section 4.2) which represent single-modal and multi-modal functions. For each pair of A and p_{inc}, we execute the algorithm 100 times, with population sizes $\lambda = 10^4\ln(n)$ and $\mu = \lambda/8$. Figs. 2(a), (b) and (c) show the medians of the runtimes of the MOSA-EA for different parameters A and p_{inc} on OneMax, LeadingOnes and Funnel, respectively. The maximal number of fitness evaluations is 10^9.

From Figs. 2, the algorithm finds the optimum within 10^7 function evaluations for an extensive range of parameter settings. The algorithm is slow when A and p_{inc} are too large. Therefore, we recommend to set $p_{\text{inc}} \in (0.3, 0.5)$ and $A \in (1.01, 1.5)$. For the remainder of the paper, we will choose $p_{\text{inc}} = 0.4$ and $A = 1.01$. We also recommend to use a sufficiently large population size $\lambda = c\ln(n)$ for some large constant c. We will state the population sizes λ and μ later.

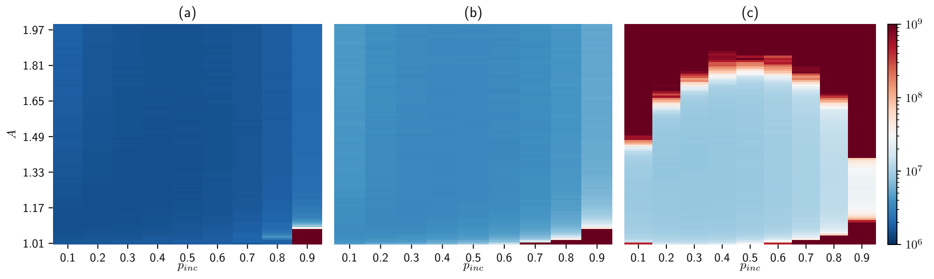

Fig. 2: Median runtimes of the MOSA-EA for different parameters A and p_{inc} on (a) OneMax, (b) LeadingOnes and (c) Funnel over 100 independent runs ($n = 100$).

4 Experimental settings and methodology

We compare the performance of the MOSA-EA with eleven other heuristic algorithms on three classical pseudo-Boolean functions and two more complex combinatorial optimisation problems. We also empirically study the MOSA-EA in noisy environments. In this section, we will first introduce the other algorithms and their parameter settings. We will then describe the definitions of benchmarking functions and problems. We will also indicate the statistical approach applied in the experiments.

4.1 Parameter settings in other algorithms

We consider eleven other heuristic algorithms, including three single-individual elitist algorithms, *random search* (RS), *random local search* (RLS) and (1+1) EA, two population-based elitist algorithms, $(1+(\lambda,\lambda))$ GA [15] and FastGA [16], two *estimation of distribution algorithms* (EDAs), cGA [24] and UMDA [37], two non-elitist EAs, 3-tournament EA and (μ,λ) EA, and two self-adjusting EAs, SA-EA [7] and self-adjusting population size $(1,\{F^{1/s}\lambda,\lambda/F\})$ EA (SA-$(1,\lambda)$ EA) [27], and a problem-specific algorithm, Open-WBO [35]. These heuristic algorithms are proved to be efficient in many scenarios, e.g., in multi-modal and noisy optimisation [3, 6, 7, 9, 10, 16, 18, 19, 26, 31]. Open-WBO is a MaxSat solver that operates differently than randomised search heuristics. It was one of the best MaxSat solvers in *MaxSAT Evaluations* 2014, 2015, 2016 and 2017 [2].

It is essential to set proper parameters for each algorithm for a comparative study [5]. In the experiments, we use parameter recommendations from the existing theoretical and empirical studies, which are summarised in Tab. 1.

Note that to investigate the effect of self-adaptation via multi-objectivisation, the SA-EA applied the same self-adapting mutation rate strategy and initialisation method as the MOSA-EA, instead of the strategy used in [7]. The only difference between the SA-EA and the MOSA-EA in the experiments is the sorting mechanism: The SA-EA uses the fitness-first sorting mechanism [7] (Alg. 7 in Appendix 1) and the MOSA-EA uses the multi-objective sorting mechanism.

Table 1: Parameter settings of algorithms considered in this paper

Category	**Algorithm**	**Parameter Settings**
Elitist EAs	RS	-
	RLS	-
	(1+1) EA	Mutation rate $\chi/n = 1/n$
	$(1+(\lambda,\lambda))$ GA [15]	Mutation rate $p = \lambda/n$; Crossover bias $c = 1/\lambda$; Population size $\lambda = 2\ln(n)$ [6]
	FastGA [16]	$\beta = 1.5$ [16]
EDAs	cGA [24]	$K = 7\sqrt{n}\ln(n)$ [43]
	UMDA [37]	$\mu = \lambda/8$
Non-Elitist EAs	3-tournament EA	Mutation rate $\chi/n = 1.09812/n$ [9, 10]
	(μ,λ) EA	Mutation rate $\chi/n = 2.07/n$; Population size $\mu = \lambda/8$ [29, 34]
Self-adjusting EAs	SA-$(1,\lambda)$ EA [27]	Population size $\lambda_{\text{init}} = 1$, $\lambda_{\max} = enF^{1/s}$; $F = 1.5$, $s = 1$ [27]
	SA-EA	Population size $\mu = \lambda/8$; $A = 1.01$, $p_{\text{inc}} = 0.4$, $\varepsilon = 1/(2\ln(n))$
	MOSA-EA	Population size $\mu = \lambda/8$; $A = 1.01$, $p_{\text{inc}} = 0.4$, $\varepsilon = 1/(2\ln(n))$
MaxSat solver	Open-WBO [35][1]	Default

[1] We use version 2.1: `https://github.com/sat-group/open-wbo`.

4.2 Classical functions

We first consider two well-known unimodal functions, OneMax and LeadingOnes, i.e., $\mathrm{OM}(x) := \sum_{i=1}^{n} x(i)$ and $\mathrm{LO}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x(j)$. One would not expect to encounter these functions in real-world optimisation. However, they serve as a good starting point to analyse the algorithms. We cannot expect good performance from an algorithm which performs poorly on these simple functions. We also consider Funnel which was proposed by Dang et al. [9], It is a bi-modal function with sparse local optima and a dense fitness valley which belongs to the problem class $\mathrm{SparseLocalOpt}_{\alpha,\varepsilon}$ [10]. The parameters u, v, w in the Funnel function describe the sparsity of the deceptive region and the density of the fitness valley. Dang et al. [9] proved that the $(\mu + \lambda)$ EA and the (μ, λ) EA are inefficient on Funnel if $v - u = \Omega(n)$ and $w - v = \Omega(n)$, while the 3-tournament EA with the mutation rate $\chi/n = 1.09812/n$ can find the optimum in polynomial runtime. In the experiments, we test the Funnel function with the parameters $u = 0.5n$ $v = 0.6n$ and $w = 0.7n$ which satisfy the restrictions above.

For each problem, we independently run each algorithm 30 times for each problem size n from 100 to 200 with step size 10 and record the runtimes. For fair comparison, we set sufficiently large population sizes $\lambda = 10^4 \ln(n)$ for the MOSA-EA, the 3-tournament EA, the (μ, λ) EA, the UMDA and the SA-EA.

4.3 Combinatorial optimisation problems

We consider two NP-hard problems, the random NK-Landscape problem and the random k-Sat problem, which feature many local optima [28, 38]. We compare the performance of the MOSA-EA with other popular randomised search heuristics in a fitness evaluation budget. To further investigate the MOSA-EA, we compare it with the problem-specific algorithm Open-WBO [35] which is a MaxSat solver in a fixed CPU time.

For fair comparisons, we set the population sizes $\lambda = 20000$ for the MOSA-EA, the 3-tournament EA, the (μ, λ) EA, the UMDA and the SA-EA. We also apply Wilcoxon rank-sum tests [44] between the results of each algorithm and the MOSA-EA.

Random NK-Landscape problems The NK-Landscape problem [28] can be described as: given $n, k \in \mathbb{N}$ satisfying $k \leq n$, and a set of sub-functions $f_i : \{0,1\}^k \to \mathbb{R}$ for $i \in [n]$, to maximise $\text{NK-Landscape}(x) := \sum_{i=1}^{n} f_i\left(\Pi\left(x, i\right)\right)$, where the function $\Pi : \{0,1\}^n, [n] \to \{0,1\}^k$ returns a bit-string containing k right side neighbours of the i-th bit of x, i.e., $x_i, \ldots, x_{(i+k-1) \bmod n}$. Typically, each sub-function is defined by a lookup table with 2^{k+1} entries, each in the interval $(0, 1)$. The "difficulty" of instance can be varied by changing k [39]. Generally, the problem instances are considered to become harder for larger k. We generate 100 random NK-Landscape instances with $n = 100$ for each $k \in \{5, 10, 15, 20, 25\}$ by uniformly sampling values between 0 and 1 in the lookup table. We run each algorithm once on each instance and record the highest fitness value achieved in the fitness evaluation budget of 10^8.

Random k-Sat problems The k-Sat problem is an optimisation problem that aims to find an assignment in the search space $\{0,1\}^n$ which maximises the number of satisfied clauses of a given Boolean formula in conjunctive normal form [1,8,23]. For each random k-Sat instance, each of m clauses have k literals which are sampled uniformly at random from $[n]$ without replacement. We first generate 100 random k-Sat instances with $k = 5$, $n = 100$ and $m \in \{100i \mid i \in [30]\}$. Similarly with the NK-Landscape experiments, we run each algorithm on these random instances and record the smallest number of unsatisfied clauses during runs of 10^8 fitness function evaluations. Additionally, we run Open-WBO and the MOSA-EA on the same machine in one hour CPU time. The MOSA-EA is implemented in OCaml, while OpenWBO is implemented in C++, which generally leads to faster-compiled code than OCaml.

4.4 Noisy optimisation

We consider the one-bit noise model (q) which are widely studied [**?**, 18, 20, 21, 31, 40]. Let $f^n(x)$ denote the noisy fitness value. Then the one-bit noise model (q) can be described as: given a probability $q \in [0, 1]$, i.e., noise level, and a solution $x \in \{0,1\}^n$, then $f^n(x) = \begin{cases} f(x) & \text{with probability } 1-q \\ f(x') & \text{with probability } q \end{cases}$, where x' is a uniformly sampled Hamming neighbour of x.

From the previous studies [11, 32], non-elitist EAs can cope with the higher-level noise by reducing the mutation rate. However, we need to know the exact noise level to set a proper mutation rate. Our aim with the noisy optimisation experiments is to investigate whether the mutation rate self-adapts to the noise level when using the multi-objective self-adaptation mechanism. Therefore, we set the mutation rate of the (μ, λ) EA as $\ln(\lambda/\mu)/(2n)$ instead of the value close to the error threshold [29, 34]. For a fair comparison, we set population sizes $\lambda = 10^4 \ln(n)$ and $\mu = \lambda/16$ for both the MOSA-EA and the (μ, λ) EA. The difference between the two EAs is only the parameter control method.

We test the algorithms on LeadingOnes in the one-bit noise model with noise levels $q = \{0.2, 0.6, 0.8, 0.9\}$. For each problem size $n = 100$ to 200 with step size 10, we execute 100 independent runs for each algorithm and record the runtimes. To track the behaviours of the MOSA-EA under different levels of noise, we also record the mutation rate of the individual with the highest real fitness value during the run.

5 Results and discussion

5.1 Classical functions

Figs. 3(a), 3(b) and 3(c) show the runtimes of the MOSA-EA and nine other heuristic algorithms on OneMax, LeadingOnes and Funnel over 30 independent runs, respectively. Based on theoretical results [25], the expected runtimes of the (1+1) EA are $O(n \log(n))$ and $O(n^2)$ on OneMax and LeadingOnes, respectively. We thus normalise the y-axis of Figs. 3(a) and (b) by $n \ln(n)$ and n^2, respectively. We also use the log-scaled y-axis for Figs. 3(a) and 3(b). The runtime of the 3-tournament EA with a mutation rate $\chi/n = 1.09812/n$ and a population size $c \log(n)$ for a sufficiently large constant c on Funnel is $O(n^2 \log(n))$ [9]. We thus normalise the y-axis of Fig. 3(c) by $n^2 \ln(n)$. Note that (1+1) EA, RLS, (μ, λ) EA, cGA, FastGA, $(1 + (\lambda, \lambda))$ GA, SA-EA and

SA-$(1,\lambda)$ EA cannot achieve the optimum of the FUNNEL function in 10^9 fitness evaluations. It is known that no elitist black-box algorithm can optimise FUNNEL in polynomial time with high probability [9,10].

Although the MOSA-EA is slower than EDAs and elitist EAs on the unimodal functions ONEMAX and LEADINGONES, it has comparable performance with the other non-elitist EAs and the SA-EA. Recall theoretical results on FUNNEL [9,10], elitist EAs and the (μ,λ) EA fail to find the optimum, while the 3-tournament EA is efficient. The results in Fig. 3 (c) are consistent with the theoretical results. In this paper, the (μ,λ) EA, the SA-EA and the MOSA-EA use the (μ,λ) selection. Compared with the (μ,λ) EA and the SA-EA, self-adaptation via multi-objectivisation can cope with sparse local optima and even achieve a better performance than the 3-tournament EA.

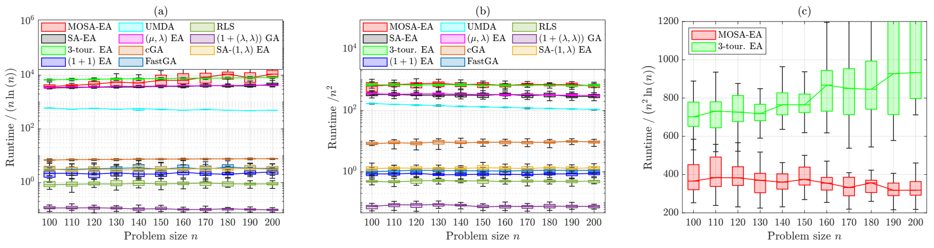

Fig. 3: Runtimes of nine algorithms on the (a) ONEMAX, (b) LEADINGONES, (c) FUNNEL ($u = 0.5n, v = 0.6n, w = 0.7n$) functions over 30 independent runs. The y-axis in sub-figures (a) and (b) are log-scaled. (1+1) EA, RLS, (μ,λ) EA, cGA, FastGA, $(1+(\lambda,\lambda))$ GA, SA-EA and SA-$(1,\lambda)$ EA cannot find the optimum of the FUNNEL function in 10^9 fitness evaluations.

5.2 Combinatorial optimisation problems

Random NK-Landscape problems Fig. 4 illustrates the experimental results of eleven algorithms on random NK-LANDSCAPE problems. From Wilcoxon rank-sum tests, the highest fitness values achieved by the MOSA-EA are statistically significantly higher than all other algorithms with significance level $\alpha = 0.05$ for all NK-LANDSCAPE with $k \in \{10, 15, 20, 25\}$. Furthermore, the advantage of the MOSA-EA is more significant for the harder problem instances.

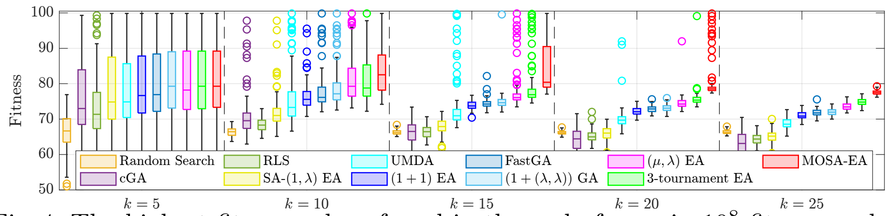

Fig. 4: The highest fitness values found in the end of runs in 10^8 fitness evaluations on 100 random NK-LANDSCAPE instances with different k ($n = 100$).

Fig. 5 illustrates the highest fitness values found during the optimisation process on one random NK-LANDSCAPE instance ($k = 20$, $n = 100$). Note that the non-elitist algorithms, i.e., EDAs, (μ,λ) EA, 3-tournament EA, SA-$(1,\lambda)$ EA and MOSA-EA, do not always keep the best solution found. Therefore, the corresponding lines might fluctuate. In contrast, the elitist EAs, e.g., (1+1) EA, increase the fitness value monotonically during the whole run.

The elitist EAs converge quickly to solutions of medium quality, then stagnate. In contrast, the 3-tournament EAs, the (μ, λ) EA and the MOSA-EA improve the solution steadily. Most noticeably, the MOSA-EA improves the solution even after 10^7 fitness evaluations.

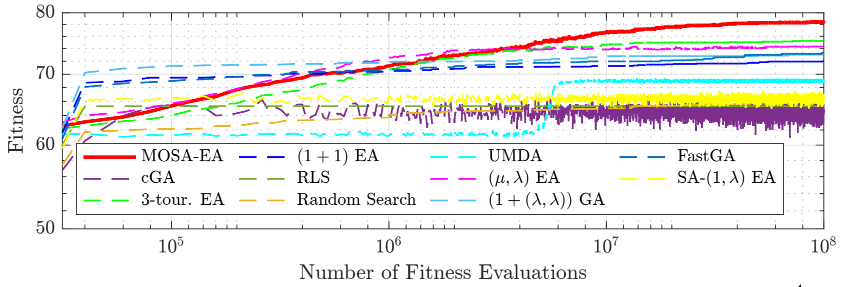

Fig. 5: The median of the highest fitness values found in every 2×10^4 fitness evaluations over 30 independent runs on one random NK-LANDSCAPE instance ($k = 20$, $n = 100$). The x-axis is log-scaled.

Random k-Sat problems Fig. 6 illustrates the medians of the smallest number of unsatisfied clauses found in the 10^8 fitness evaluations budget among eleven algorithms on 100 random k-SAT instances ($k = 5$, $n = 100$) with different total number of clauses m. Coja-Oghlan [8] proved that the probability of generating a satisfiable instance drops from nearly 1 to nearly 0, if the ratio of the number of clauses m and the problem size n is greater than a threshold, $r_{k-\text{SAT}} = 2^k \ln(2) - \frac{1}{2}(1 + \ln(2)) + o_k(1)$, where $o_k(1)$ signifies a term that tends to 0 in the limit of large k. In this case, $r_{k-\text{SAT}}$ is roughly 2133 if we ignore the $o_k(1)$ term. We therefore call an instance with $m \geq 2133$ hard. The MOSA-EA is statistically significantly better than the other ten algorithms with significance level $\alpha = 0.05$ on hard instances from Wilcoxon rank-sum tests.

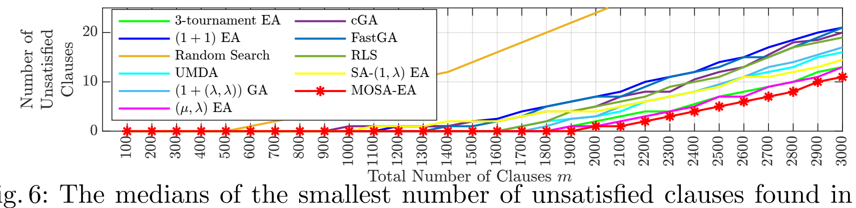

Fig. 6: The medians of the smallest number of unsatisfied clauses found in 10^8 fitness evaluations on 100 random k-SAT instances with different total numbers of clauses m ($k = 5$, $n = 100$).

Fig. 7 illustrates the smallest number of unsatisfied clauses of the best solution found during the optimisation process on one random k-SAT instance ($k = 5$, $n = 100$, $m = 2500$). From Fig. 7, we come to similar conclusions with the experiments on NK-LANDSCAPE (Fig. 5).

Fig. 8 illustrates the medians of the smallest number of unsatisfied clauses found in one hour CPU-time budget of Open-WBO and the MOSA-EA on 100

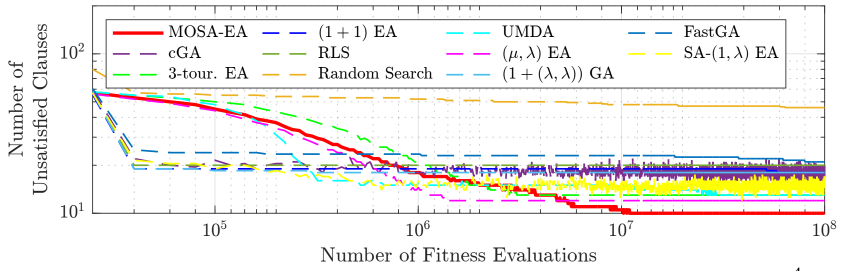

Fig. 7: The smallest number of unsatisfied clauses found over 2×10^4 fitness evaluations over 30 independent runs on one random k-Sat instance ($k = 5$, $n = 100$, $m = 2500$). The axis are log-scaled.

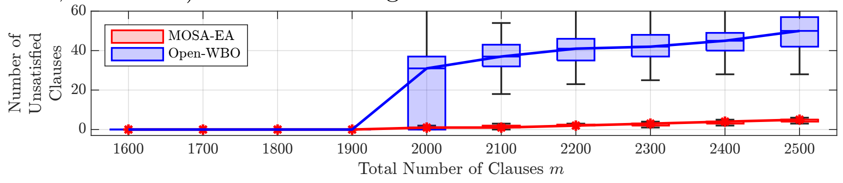

Fig. 8: The smallest number of unsatisfied clauses found in one hour CPU-time on 100 random k-Sat instances with different total numbers of clauses m ($k = 5$, $n = 100$).

random k-Sat instances ($k = 5$, $n = 200$) with different total number of clauses m. For the instances with small numbers of clauses, i.e., $m \leq 1900$, Open-WBO returns all satisfied assignments within a few minutes, while the MOSA-EA takes up to one hour to find all satisfied assignments. However, the performance of the MOSA-EA is statistically significantly better than Open-WBO on hard instances in one hour of CPU time.

5.3 Noisy optimisation

Figs. 9 show the runtimes of the MOSA-EA and the (μ, λ) EA on LeadingOnes in the one-bit noise model. With a fixed mutation rate $\chi = \ln(\lambda/\mu)/(2n) = \ln(16)/(2n) \approx 1.386/n$, the runtimes of the (μ, λ) EA could be in $O(n^2)$ for low-level noise, while the runtimes rise sharply as problem size growing if the noise levels are $q = 0.9$. Based on the theoretical results [32], we could furthermore cope with the higher-level noise by using a lower mutation rate.

However, the exact noise level in real-world optimisation is usually unknown. Self-adaptation might help to configure the proper mutation rate automatically. From Fig. 9, the MOSA-EA handles the highest levels of one-bit noise, where the (μ, λ) EA encounters a problem. Fig. 10 illustrates the relationships between mutation rates and real fitness values of the MOSA-EA. We observe a decrease in the mutation rate when the noise level increases. In particular, the MOSA-EA automatically adapts the mutation rate to below $1.386/n$, when using the (μ, λ) EA close to the optimum under the highest noise level. The lower mutation rate could be the reason for successful optimisation under noise.

6 Conclusion

EAs applied to noisy or multi-modal problems can benefit from non-elitism. However, it is non-trivial to set the parameters of non-elitist EAs appropriately.

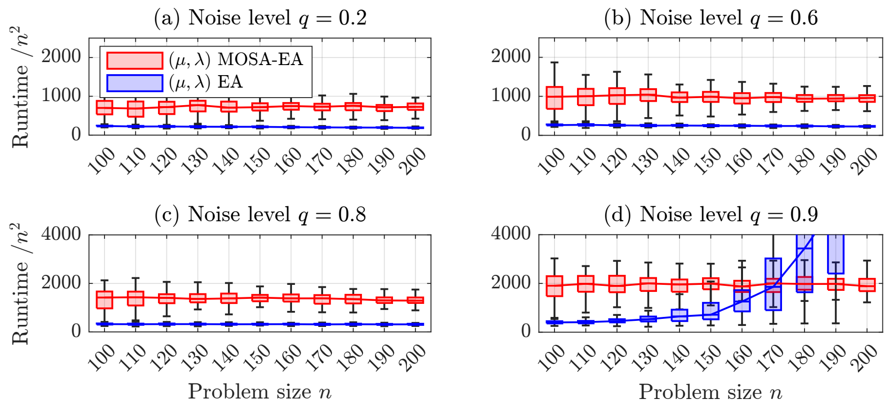

Fig. 9: Runtimes of the MOSA-EA and the (μ, λ) EA with the fixed mutation rate $\chi/n = 1.386/n$ on LeadingOnes under one-bit noise with different noise levels q. ($\lambda = 10^4 \ln(n)$, $\mu = \lambda/16$)

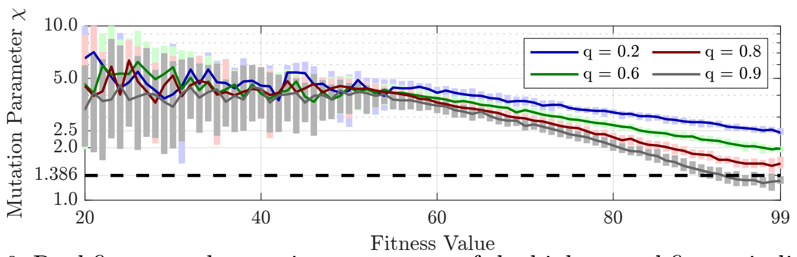

Fig. 10: Real fitness and mutation parameter of the highest real fitness individual per generation of the MOSA-EA on LeadingOnes under one-bit noise with different noise levels q. Lines show median value of mutation parameter χ. The corresponding shadows indicate the IQRs. The y-axis is log-scaled. ($n = 100$, $\lambda = 10^4 \ln(n)$, $\mu = \lambda/16$)

Self-adaptation via multi-objectivisation, a parameter control method, is proved to be efficient in escaping local optima with unknown sparsity [33]. This paper continues the study of MOSA-EA through an empirical study of its performance on a wide range of combinatorial optimisation problems. We first empirically study the MOSA-EA on theoretical benchmark problems. The performance of the MOSA-EA is comparable with other non-elitist EAs on unimodal functions, i.e., OneMax and LeadingOnes. Self-adaption via multi-objectivisation can also help to cope with sparse local optima. For the NP-hard combinatorial optimisation problems, random NK-Landscape and k-Sat, the MOSA-EA is significantly better than the other nine heuristic algorithms. In particular, the MOSA-EA can beat a state-of-the-art MaxSat solver on some hard random k-Sat instances in a fixed CPU time. We then experimentally analyse the MOSA-EA in noisy environments. The results also demonstrate that self-adaptation can adapt mutation rates to given noise levels. In conclusion, the MOSA-EA outperforms a range of optimisation algorithms on several multi-modal and noisy optimisation problems.

Acknowledgements This work was supported by a Turing AI Fellowship (EPSRC grant ref EP/V025562/1). The computations were performed using University of Birmingham's BlueBEAR and Baskerville HPC service.

Appendix 1 Omitted algorithms in this paper

Algorithm 3 Multi-objective sorting mechanism [33]

```
Require: Population sizes $\lambda \in \mathbb{N}$. Population $P_t \in \mathcal{Y}^\lambda$. Fitness function $f$.
 1: Sort $P_t$ into strict non-dominated fronts $\mathcal{F}_0^t, \mathcal{F}_1^t, \dots$ based on $f_1(x, \chi) := f(x)$ and
    $f_2(x, \chi) := \chi$.
 2: for $\mathcal{F} = \mathcal{F}_0^t, \mathcal{F}_1^t, \dots$ do
 3:     Sort $\mathcal{F}$ such that $f_1(\mathcal{F}(1)) > f_1(\mathcal{F}(2)) > \dots$.
 4: $P_t := (\mathcal{F}_0^t, \mathcal{F}_1^t, \dots)$.
 5: return $P_t$.
```

Algorithm 4 Strict non-dominated sorting [33]

```
Require: Population sizes $\lambda \in \mathbb{N}$. Population $P \in \mathcal{Z}^\lambda$, where $\mathcal{Z}$ is a finite state
    space. Objective functions $f_1, f_2, \dots : \mathcal{Z} \to \mathbb{R}$ (assume to maximise all objective
    functions).
 1: for each individual $P(i)$ do
 2:     Set $S_i := \emptyset$ and $n_i := 0$.
 3: for $i = 1, \dots, \lambda$ do
 4:     for $j = 1, \dots, \lambda$ do
 5:         if $P(i) \prec P(j)$ based on $f_1, f_2, \dots$ then
 6:             $S_i := S_i \cup \{P(i)\}$,
 7:         else if $P(j) \prec P(i)$ based on $f_1, f_2, \dots$ then
 8:             $n_i := n_i + 1$,
 9:         else if $f_\ell(P(i)) = f_\ell(P(j))$ where $\ell = 1, 2, \dots$ then
10:             if $P(i) \notin S_j$ then $S_i := S_i \cup \{P(i)\}$ else $n_i := n_i + 1$.
11:     if $n_i = 0$ then $\mathcal{F}_0 = \mathcal{F}_0 \cup \{P(i)\}$.
12: Set $k := 0$.
13: while $\mathcal{F}_k \neq \emptyset$ do
14:     $Q := \emptyset$.
15:     for each individual $P(i) \in \mathcal{F}_k$ and $P(j) \in S_i$ do
16:         Set $n_j := n_j - 1$.
17:         if $n_j = 0$ then $Q := Q \cup \{P(j)\}$.
18:     Set $k := k + 1$, $\mathcal{F}_k := Q$.
19: return $\mathcal{F}_0, \mathcal{F}_1, \dots$.
```

Algorithm 5 Multi-objective sorting mechanism (alternative)

```
Require: Population sizes $\lambda \in \mathbb{N}$. Population $P_t \in \mathcal{Y}^\lambda$. Fitness function $f$.
 1: Sort $P_t$ into $P_t^1, P_t^1, \dots$ where $P_t^1$ containing all individuals with the highest fitness
    $f$, $P_t^2$ containing all individuals with the 2nd highest fitness $f$, $\dots$.
 2: for $i = 1, \dots, \lambda$ do
 3:     Set $\hat{\chi} := -\infty$.
 4:     for $Q = P_t^1, P_t^1, \dots$ do
 5:         Find $(x', \chi')$ which is the element with the highest $\chi$ in $Q$.
 6:         if $Q \neq \emptyset$ and $\chi' > \hat{\chi}$ then
 7:             $P_t(i) := (x', \chi')$ and $\hat{\chi} := \chi'$.
 8:             Pop $(x', \chi')$ from $Q$.
 9:             Break.
10: return $P_t$.
```

Algorithm 6 (μ, λ) selection

```
Require: Population size $\lambda \in \mathbb{N}$. Parameter $\mu \in [\lambda]^3$.
 1: $I_t \sim \text{Unif}([\mu])$.
 2: return $I_t$.
```

Algorithm 7 Fitness-first sorting mechanism [7]

```
Require: Population sizes $\lambda \in \mathbb{N}$. Population $P_t \in \mathcal{Y}^\lambda$. Fitness function $f$.
 1: Sort $P_t$ such that $P_t(1) \succeq \cdots \succeq P_t(\lambda)$, according to
 2:       $(x, \chi) \succeq (x', \chi') \Leftrightarrow f(x) > f(x') \vee (f(x) = f(x') \wedge \chi \geq \chi')$.
 3: return $P_t$.
```

[3] For any $n \in \mathbb{N}$, we define $[n] := \{1, \dots, n\}$

Appendix 2 Omitted statistical results of experiments

Table 2: Statistical results of experiments on random NK-Landscape problems. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and MOSA-EA.

k	**Stat.**	**RS**	**cGA**	**UMDA**	**RLS**	**SA-**$(1,\lambda)$**EA**	$(1+1)$**EA**	**FastGA**	$(1+(\lambda,\lambda))$**GA**	(μ,λ)**EA**	**3-tour.EA**	**MOSA-EA**
5	Median	66.6591	72.9964	74.8631	71.3547	74.8418	76.6613	76.9230	79.2846	78.2089	79.2846	**79.2846**
	p-value	2.1e-22	2.3e-04	0.0213	6.5e-08	0.0226	0.2668	0.4215	0.9299	0.7985	0.8805	-
10	Median	66.4442	69.5499	73.2968	68.3100	71.0248	75.5792	76.1340	77.1520	79.2680	78.7832	**82.5270**
	p-value	2.6e-34	1.5e-26	2.0e-15	2.6e-34	3.5e-34	5.0e-18	1.1e-12	2.2e-09	0.0030	0.0063	-
15	Median	66.2055	66.5517	70.9576	66.4446	67.8968	73.7253	74.2253	74.6407	76.0777	76.9053	**80.4417**
	p-value	2.6e-34	2.6e-34	5.5e-22	2.6e-34	2.6e-34	2.6e-34	1.8e-33	5.2e-33	1.3e-20	1.1e-17	-
20	Median	66.1233	64.4191	69.6786	64.9865	66.0533	72.8025	72.8783	73.0882	74.2580	75.3662	**78.5247**
	p-value	2.6e-34	2.6e-34	7.0e-31	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	4.0e-33	1.2e-31	-
25	Median	66.2207	63.1222	68.5683	64.3685	65.1886	70.8648	71.7564	71.9623	73.4398	74.8115	**77.5024**
	p-value	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	1.4e-33	-

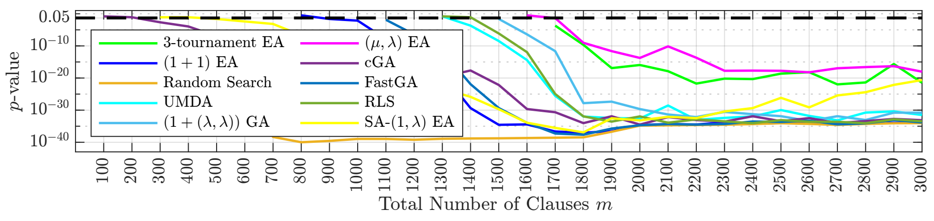

Fig. 11: The p-values of Wilcoxon rank-sum tests between the algorithms and the MOSA-EA on 100 random k-Sat instances. The y-axis is log-scaled.

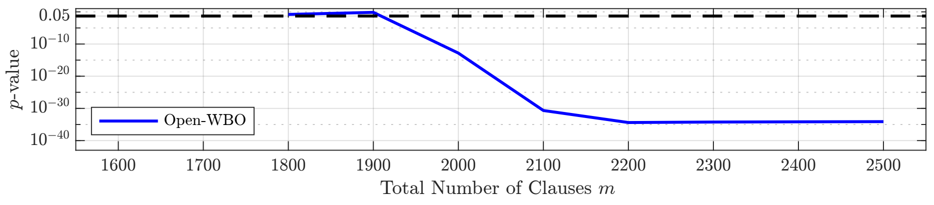

Fig. 12: The p-value of Wilcoxon rank-sum test between Open-WBO and the MOSA-EA on 100 random k-Sat instances. The y-axis is log-scaled.

References

1. Achlioptas, D., Moore, C.: Random kSAT: Two Moments Suffice to Cross a Sharp Threshold. SIAM Journal on Computing **36**(3), 740–762 (Jan 2006)
2. Ansótegui, C., Bacchus, F., Järvisalo, M., Martins, R.: MaxSAT Evaluation 2017. SAT (2017)
3. Antipov, D., Doerr, B., Karavaev, V.: A Rigorous Runtime Analysis of the $(1 + (\lambda, \lambda))$ GA on Jump Functions. Algorithmica (Jan 2022)
4. Bäck, T.: Self-Adaptation in Genetic Algorithms. In: Self Adaptation in Genetic Algorithms. pp. 263–271. MIT Press (1992)
5. Bartz-Beielstein, T., Doerr, C., Berg, D.v.d., Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., La Cava, W., Lopez-Ibanez, M., Malan, K.M., Moore, J.H., Naujoks, B., Orzechowski, P., Volz, V., Wagner, M., Weise, T.: Benchmarking in Optimization: Best Practice and Open Issues. arXiv:2007.03488 [cs, math, stat] (Dec 2020), arXiv: 2007.03488
6. Buzdalov, M., Doerr, B.: Runtime analysis of the $(1 + (\lambda, \lambda))$ genetic algorithm on random satisfiable 3-CNF formulas. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1343–1350. ACM, Berlin Germany (Jul 2017)
7. Case, B., Lehre, P.K.: Self-adaptation in non-Elitist Evolutionary Algorithms on Discrete Problems with Unknown Structure. IEEE Transactions on Evolutionary Computation **24**(4), 650–663 (2020)
8. Coja-Oghlan, A.: The asymptotic k-SAT threshold. In: Proceedings of the forty-sixth annual ACM symposium on Theory of computing. pp. 804–813. ACM, New York New York (May 2014)
9. Dang, D.C., Eremeev, A., Lehre, P.K.: Escaping Local Optima with Non-Elitist Evolutionary Algorithms. In: Proceedings of AAAI 2021. AAAI Press, Palo Alto, California USA (2020)
10. Dang, D.C., Eremeev, A., Lehre, P.K.: Non-elitist Evolutionary Algorithms Excel in Fitness Landscapes with Sparse Deceptive Regions and Dense Valleys. In: Proceedings of the Genetic and Evolutionary Computation Conference. ACM, Lille, France (2021)
11. Dang, D.C., Lehre, P.K.: Efficient Optimisation of Noisy Fitness Functions with Population-based Evolutionary Algorithms. In: Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII - FOGA '15. pp. 62–68. ACM Press, Aberystwyth, United Kingdom (2015)
12. Dang, D.C., Lehre, P.K.: Self-adaptation of Mutation Rates in Non-elitist Populations. In: Parallel Problem Solving from Nature PPSN XIV. vol. 9921, pp. 803–813. Springer International Publishing, Cham (2016)
13. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (Apr 2002)
14. Doerr, B.: Does Comma Selection Help to Cope with Local Optima? Algorithmica (Jan 2022)
15. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. Theoretical Computer Science **567**, 87–104 (Feb 2015)
16. Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 777–784. ACM, Berlin Germany (Jul 2017)

17. Doerr, B., Witt, C., Yang, J.: Runtime Analysis for Self-adaptive Mutation Rates. Algorithmica **83**(4), 1012–1053 (Apr 2021)
18. Droste, S.: Analysis of the (1+1) EA for a Noisy OneMax. In: Genetic and Evolutionary Computation GECCO 2004, vol. 3102, pp. 1088–1099. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
19. Friedrich, T., Kotzing, T., Krejca, M.S., Sutton, A.M.: The Compact Genetic Algorithm is Efficient under Extreme Gaussian Noise. IEEE Transactions on Evolutionary Computation pp. 1–1 (2016)
20. Friedrich, T., Ktzing, T., Krejca, M.S., Sutton, A.M.: Robustness of Ant Colony Optimization to Noise. Evolutionary Computation **24**(2), 237–254 (2016), publisher: MIT Press
21. Gieen, C., Ktzing, T.: Robustness of Populations in Stochastic Environments. Algorithmica **75**(3), 462–489 (Jul 2016)
22. Goldberg, D.E., Deb, K.: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In: Foundations of Genetic Algorithms, vol. 1, pp. 69–93. Elsevier, - (1991)
23. Gottlieb, J., Marchiori, E., Rossi, C.: Evolutionary Algorithms for the Satisfiability Problem. Evolutionary Computation **10**(1), 35–50 (Mar 2002)
24. Harik, G., Lobo, F., Goldberg, D.: The compact genetic algorithm. IEEE Transactions on Evolutionary Computation **3**(4), 287–297 (Nov 1999)
25. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Natural Computing **3**(1), 21–35 (2004)
26. Hevia Fajardo, M.A., Sudholt, D.: Self-adjusting offspring population sizes outperform fixed parameters on the cliff function. In: Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms. pp. 1–15. ACM, Virtual Event Austria (Sep 2021)
27. Hevia Fajardo, M.A.H., Sudholt, D.: Self-adjusting population sizes for non-elitist evolutionary algorithms: why success rates matter. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1151–1159. ACM, Lille France (Jun 2021)
28. Kauffman, S.A., Weinberger, E.D.: The NK model of rugged fitness landscapes and its application to maturation of the immune response. Journal of theoretical biology **141**(2), 211–245 (1989), publisher: Elsevier
29. Lehre, P.K.: Negative Drift in Populations. In: Parallel Problem Solving from Nature, PPSN XI. pp. 244–253. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
30. Lehre, P.K.: Fitness-levels for non-elitist populations. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11. p. 2075. ACM Press, Dublin, Ireland (2011)
31. Lehre, P.K., Nguyen, P.T.H.: Runtime Analyses of the Population-Based Univariate Estimation of Distribution Algorithms on LeadingOnes. Algorithmica **83**(10), 3238–3280 (Oct 2021)
32. Lehre, P.K., Qin, X.: More Precise Runtime Analyses of Non-elitist EAs in Uncertain Environments. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1160–1168. ACM, Lille, France (2021)
33. Lehre, P.K., Qin, X.: Self-adaptation via Multi-objectivisation: A Theoretical Study. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1417–1425. ACM, Boston USA (2022)
34. Lehre, P.K., Yao, X.: On the Impact of Mutation-Selection Balance on the Runtime of Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation **16**(2), 225–241 (Apr 2012)

35. Martins, R., Manquinho, V., Lynce, I.: Open-WBO: A Modular MaxSAT Solver. In: Theory and Applications of Satisfiability Testing SAT 2014, vol. 8561, pp. 438–445. Springer International Publishing, Cham (2014)
36. Meyer-Nieberg, S.: Self-adaptation in evolution strategies. PhD Thesis, Dortmund University of Technology (2007)
37. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: International conference on parallel problem solving from nature. pp. 178–187. Springer (1996)
38. Ochoa, G., Chicano, F.: Local optima network analysis for MAX-SAT. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 1430–1437. ACM, Prague Czech Republic (Jul 2019)
39. Ochoa, G., Tomassini, M., Vrel, S., Darabos, C.: A study of NK landscapes' basins and local optima networks. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation - GECCO '08. p. 555. ACM Press, Atlanta, GA, USA (2008)
40. Qian, C., Bian, C., Jiang, W., Tang, K.: Running Time Analysis of the (1+1)-EA for OneMax and LeadingOnes Under Bit-Wise Noise. Algorithmica **81**(2), 749–795 (Feb 2019)
41. Smith, J.E.: Self-Adaptation in Evolutionary Algorithms for Combinatorial Optimisation. In: Adaptive and Multilevel Metaheuristics, pp. 31–57. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
42. Srinivas, N., Deb, K.: Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation **2**(3), 221–248 (Sep 1994)
43. Sudholt, D., Witt, C.: Update Strength in EDAs and ACO: How to Avoid Genetic Drift. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016. pp. 61–68. ACM, Denver Colorado USA (Jul 2016)
44. Wilcoxon, F.: Individual comparisons by ranking methods. In: Breakthroughs in statistics, pp. 196–202. Springer (1992)