Self-adaptation via multi-objectivisation

Lehre, Per Kristian; Qin, Xiaoyu

DOI:
10.1145/3512290.3528836

Self-adaptation via Multi-objectivisation: A Theoretical Study

Per Kristian Lehre*
University of Birmingham
Birmingham, United Kingdom

Xiaoyu Qin*
University of Birmingham
Birmingham, United Kingdom

ABSTRACT

The exploration vs exploitation dilemma is to balance exploring new but potentially less fit regions of the fitness landscape while also focusing on regions near the fittest individuals. For the tunable problem class SparseLocalOpt, a non-elitist EA with tournament selection can limit the percentage of "sparse" local optimal individuals in the population using a sufficiently high mutation rate (Dang et al., 2021). However, the performance of the EA depends critically on choosing the "right" mutation rate, which is problem instance-specific. A promising approach is self-adaptation, where parameter settings are encoded in chromosomes and evolved.

We propose a new self-adaptive EA for single-objective optimisation, which treats parameter control from the perspective of multi-objective optimisation: The algorithm simultaneously maximises the fitness and the mutation rates. Since individuals in "dense" fitness valleys survive high mutation rates, and individuals on "sparse" local optima only survive with lower mutation rates, they can co-exist on a non-dominated Pareto front.

Runtime analyses show that this new algorithm (MOSA-EA) can efficiently escape a local optimum with unknown sparsity, where some fixed mutation rate EAs become trapped. Complementary experimental results show that the MOSA-EA outperforms a range of EAs on random NK-Landscape and k-Sat instances.

CCS CONCEPTS

• **Theory of computation → Optimisation with randomised search heuristics**;

KEYWORDS

Evolutionary algorithms, self-adaptation, multi-modal functions

ACM Reference Format:
Per Kristian Lehre and Xiaoyu Qin. 2022. Self-adaptation via Multi-objectivisation: A Theoretical Study. In *Genetic and Evolutionary Computation Conference (GECCO '22), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3512290.3528836

1 INTRODUCTION

Evolutionary algorithms (EAs) can be good solvers for multi-modal optimisation problems if they balance exploring new but potential less fit regions of the fitness landscape while also focusing on the regions near the fittest individuals [28]. In the past decade, several studies in the area of runtime analysis investigated how EAs can cope with *local optima*. In addition to mechanisms like crossover [2, 10], stagnation detection [3, 26] and adapting population size [19], a large mutation rate was shown to help some mutation-EA only escaping certain local optima. For example, the (1+1) EA can be sped up on $\textsc{Jump}_k$ by using the larger mutation rate k/n rather than $1/n$ [3]. However, a too large mutation rate may lead to failed optimisation. For non-elitist EAs, there are *error thresholds* of mutation rate values [21], where if the mutation rate is above the error threshold, the runtime of the algorithm is exponential.

Non-elitist EAs can "jump" a large Hamming distance. But they can potentially also maintain less fit individuals in the population, allowing the population to cross a fitness valley. They might keep some currently low but potentially high fitness individuals in the population and optimise them "smoothly". Recently, a tunable problem class SparseLocalOpt was proposed to describe a kind of fitness landscapes with *sparse deceptive regions* (local optima) and *dense fitness valleys* [9]. Informally, every search point in a dense set has many neighbours in that set, and every search point in a sparse set has few members in any direction. Dang et al. [9] show that EAs with a non-linear selection and a sufficiently high mutation rate, i.e., close to the error threshold, can cope with sparse local optima. Non-linear selection is a type of non-elitist selection, in which the probability of each individual to be selected is based on its rank in the population, e.g., tournament and linear ranking selections [23]. Typically, the fitter individual has a higher probability to be selected, but the worse individual still has some chance to be chosen. From their analysis, non-linear selections and sufficiently high mutation rates can limit the percentage of "sparse" local optimal individuals in the population by choosing a sufficiently high mutation rate. The reason is that the sparse local optimal individuals can have a higher chance to be selected but can only survive a small percentage of such individuals after mutation, while the dense fitness valley individuals may have less chance of being selected but can have higher chance of surviving mutation. However, the performance of the EA depends critically on choosing the "right" mutation rate, which should be sufficiently high but below the error threshold. Moreover, finding such a mutation rate might be difficult or infeasible for some problem instances with not too sparse local optima.

The self-adaptive parameter control mechanism is a promising method to configure parameters. It encodes the parameters in each individual and evolves the parameters together with its solution through variation operators. There exist a few theoretical studies of self-adaptation in the decade. Dang and Lehre [12] first present that a self-adaptive population using two mutation rates can perform well on the PeakedLO function which is a simple artificial two-peak function. As a comparison, they also proved that the algorithm using neither fixed mutation rate nor uniformly selected

*Authors are listed in alphabetical order.

GECCO '22, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9237-2/22/07…$15.00
https://doi.org/10.1145/3512290.3528836

Table 1: Theoretical results of EAs on $\textsc{PeakedLO}_{m,k}$ (for some constant $c, \delta > 0$)

Algorithm	$\textsc{PeakedLO}_{m,k}$	Runtime T	Theorem
$(\mu+\lambda)$ EA	Any $k \le n$ and $k, m \in \Omega(n)$	$\Pr(T \le e^{cn}) \le e^{-\Omega(n)}$	Theorem 2
(μ,λ) EA	Any $k \le n$ and $k, m \in \Omega(n)$	$\Pr(T \le e^{cn}) \le e^{-\Omega(n)}$	Theorem 3
2-tour. EA	Any $k < (\ln(3/2)-\delta)n$ and $k, m \in \Omega(n)$	$\Pr(T \le e^{cn}) \le e^{-\Omega(\lambda)}$	Theorem 4
(μ,λ) MOSA-EA	Any $n \ge k \in \Omega(n)$, $\lceil m \rceil < 2A(1+\ln(p_{\text{inc}})/\ln(\alpha_0) - o(1))k$ [2]	$E[T] = O\left(n^2 \log(n)\right)$	Theorem 5

mutation rate will fail on this problem. Doerr et al. [17] rigorously analysed a self-adaptation mechanism on $(1,\lambda)$ EA, in which the mutation rate r/n is mutated multiplying a constant or dividing a constant (uniform random selection) before mutating the solution. They prove that the algorithm optimises OneMax in expected time $O(n\log(n))$ runtime, which is the best possible results for evolutionary algorithms. Another recent rigorous analysis from Case and Lehre [5] showed that the self-adaptation of mutation rate over a continuous interval can be effective on the unknown structure version of LeadingOnes function. Their analysis divides the search space of solution and mutation rate into two-dimensional levels. They then use the level-based theorem [7] to obtain the runtime, which is asymptotically optimal among all unary unbiased black-box algorithms.

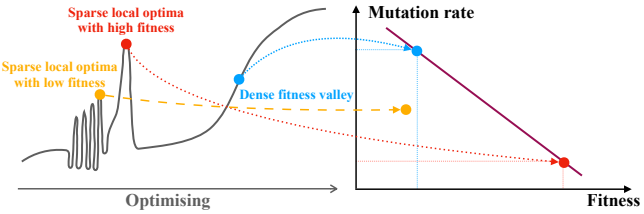

Figure 1: Intuition of MOSA-EA

Case and Lehre [5] also showed that the self-adaptation can find the (nearly) maximal "right" mutation rate for each search point on some benchmark functions. We say the "right" mutation rate for a search point is below a threshold, where the expected number of non-worse offsprings of an individual with a mutation rate below this threshold is greater than 1. For SparseLocalOpt, the maximal "right" mutation rate for dense fitness valleys can be higher, while the maximal "right" mutation rate for sparse local optima may be lower. In self-adaptation, individuals can be configured to different mutation rates. If we maximise the mutation rates for both sparse local optima and dense fitness valleys, then the sparse local optimal individual and the dense fitness valley individual could potentially co-exist in a non-dominated Pareto front (Figure 1). Therefore, we can treat parameter control from the perspective of multi-objective optimisation to find the optimal solution and the maximal "right" mutation rates for each search point simultaneously.

In this paper, we propose the *multi-objective self-adaptive EA* (MOSA-EA) for single-objective optimisation, which treats parameter control from the perspective of multi-objective optimisation: The algorithm simultaneously maximises the fitness and the mutation rates. To present advantages of the MOSA-EA, we first propose the $\textsc{PeakedLO}_{m,k}$ function, which is a version of PeakedLO [12] with tunable *sparsity*, where the fitness value is m if a local optimal individual $x = 0^k *^{n-k}$ and is LeadingOnes value otherwise, where $*$ represents an arbitrary bit. The sparsity of the local optima can be tuned by the sparse parameter $k \in [n]$. With the $\textsc{PeakedLO}_{m,k}$ function, we simulate a situation in that algorithms have already been trapped into unknown sparse local optima and estimate the expected time to reach the global optimum. The runtime analyses show that the MOSA-EA can cope with local optima for any $k \in \Omega(n)$, while the elitist EA, the (μ,λ) EA, and the tournament EA fails for some $k \in \Omega(n)$. Table 1 demonstrates the theoretical results obtained in this study. The used parameters for the algorithms can be found in the corresponding theorems. The experimental results show that the MOSA-EA can outperform the (1+1) EA, the non-elitist EAs and the UMDA on two challenging combinatorial optimisation problems, random NK-Landscape and random k-Sat instances.

[2] For some constants $p_{\text{inc}} < 2/5$, $\alpha \ge 4$, $A > 1$ based on restrictions in Theorem 5.

2 PRELIMINARIES

We first define some notations which are used later. For any $n, m \in \mathbb{N}$ satisfying $n \ge m > 0$, we define $[n] := \{1,\dots,n\}$, $[0..n] := \{0\} \cup [n]$, $[-1..n] := \{-1\} \cup [0..n]$ and $[m..n] := [0..n] \setminus [0..m-1]$. We use $H(\cdot,\cdot)$ to denote the Hamming distance. The natural logarithm and the logarithm to the base 2 are denoted by $\ln(\cdot)$ and $\log(\cdot)$, respectively. 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. Define $\textsc{LeadingOnes}(x) :=: \text{LO}(x) := \sum_{i=1}^{n} \prod_{j}^{i} x_j$. For self-adapting mutation rate, we are interested in an extended search space of $\mathcal{Y} := \mathcal{X} \times [\varepsilon, 1/2]$ which includes $\mathcal{X}$ and the space of mutation rates. Appendices mentioned can be found in the supplementary material.

2.1 $\textsc{PeakedLO}_{m,k}$ problems

Dang and Lehre [12] proposed a two-modal function $\textsc{PeakedLO}_m$ where the fitness value is m if a peak individual $x = 0^n$ and $\text{LO}(x)$ otherwise. The "density" of the fitness valley can be tuned by m. In this section, we introduce a more general version of PeakedLO, where the "sparsity" of the peak individual can be tuned by a parameter k. Let $*$ represent an arbitrary bit. Then the problem class is defined as $\textsc{PeakedLO}_{m,k}(x) := \begin{cases} m & \text{if } x = 0^k *^{n-k}, \\ \text{LO}(x) & \text{otherwise,} \end{cases}$ for any $k \in [n]$ and any $1 < m \in \mathbb{R}$. The $\textsc{PeakedLO}_m$ function [12] is a special case of $\textsc{PeakedLO}_{m,k}$ where $k = n$. Similarly to [12], we assume that the algorithm is initiated from the search points $0^k *^{n-k}$. It is unlikely that the algorithm will end up on this particular local optima if the population is initialised uniformly at random. However, here, we are interested in the capability of an algorithm to escape a local optimum, wherever it is encountered. Our assumption is therefore reasonable for the purposes of analysis.

2.2 Non-elitist EAs

Non-elitism means that the fitness individual in a generation is not always copied to the next generation. There exist many studies on non-elitist selection [8, 9, 11, 15, 22]. In this paper, we show that fixed mutation rate non-elitist EAs are inefficient when optimising $\textsc{PeakedLO}_{m,k}$ when initialised from $0^k *^{n-k}$. We take the (μ,λ) EA

and the 2-tournament EA as examples to show this, which can be described as Algorithm 6 (shown in Appendix C) with "plugging" Algorithms 8 and 9 (shown in Appendix C) into Line 4, respectively. The non-elitist selection mechanisms can also be applied to the self-adaptive EAs which is introduced in the next section.

2.3 Level-based theorem

The *level-based theorem* [7, 16] is a runtime analysis tool used to obtain upper bounds on the runtime of population-based EAs on many problems [9, 13, 22]. The theorem applies to algorithms that follow the scheme of Algorithm 7 (shown in Appendix C). Let D be some mapping from the set of all possible populations $\mathcal{Z}^\lambda$ into the space of probability distributions over of $\mathcal{Z}$, where $\mathcal{Z}$ is a finite state space. Thus, we can consider that each individual in P_{t+1} is sampled from a distribution $D(P_t)$ which responds to Lines 4-5 of Algorithm 6.

Theorem 1 ([7]). *Given a partition $(A_1, A_2, \ldots, A_m)$ of a finite state space $\mathcal{Z}$, let $T := \min\{t\lambda || P_t \cap A_m| > 0\}$ be the first point in time that the elements of A_m appear in P_t of Algorithm 7. If there exist $z_1, \ldots, z_{m-1}$, $\delta \in (0, 1]$, and $\gamma_0 \in (0, 1)$ such that for any population $P \in \mathcal{Z}^\lambda$,*

(G1) for each level $j \in [m-1]$, if $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ then $\Pr_{y\sim D(P)}\left(y \in A_{\geq j+1}\right) \geq z_j$,

(G2) for each level $j \in [m-2]$, and all $\gamma \in (0, \gamma_0]$, if $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ and $|P \cap A_{\geq j+1}| \geq \gamma\lambda$ then $\Pr_{y\sim D(P)}\left(y \in A_{\geq j+1}\right) \geq (1+\delta)\gamma$,

(G3) and the population size $\lambda \in \mathbb{N}$ satisfies $\lambda \geq \left(\frac{4}{\gamma_0\delta^2}\right)\ln\left(\frac{128m}{z_*\delta^2}\right)$, where $z_* := \min_{j\in[m-1]}\{z_j\}$,

then $E[T] \leq \left(\frac{8}{\delta^2}\right)\sum_{j=1}^{m-1}\left(\lambda \ln\left(\frac{6\delta\lambda}{4+z_j\delta\lambda}\right) + \frac{1}{z_j}\right)$.

3 MULTI-OBJECTIVE SELF-ADAPTIVE EA

This section will introduce the multi-objective self-adaptive evolutionary algorithm (MOSA-EA). The basic idea of the MOSA-EA is to treat the parameter optimisation task as another objective, i.e., maximising fitness mutation rate simultaneously. Compared to existing self-adaptive EAs, the MOSA-EA is characterised by the population sorting method. Previous self-adaptive EAs [5, 12, 17] sort the population by considering the fitness of individuals first, then the mutation rate. In contrast, the MOSA-EA sorts the population by a non-dominated sorting (Algorithm 2) where we have adapted from two famous multi-objective EAs [14, 27].

3.1 Framework of self-adaptive EAs

There exist three self-adapting mutation rate non-elitist EAs in the theory community. They are the 2-tournament self-adaptive EA using two mutation rates [12], the (μ, λ) self-adaptive EA [5] and the $(1, \lambda)$ self-adaptive EA [17]. In each generation t, they all essentially first sort the population P_t by some sorting mechanism based on fitness function f and mutation rate $\frac{\chi}{n}$. Then, each individual in the next population P_{t+1} is produced via selection and mutation. The selection mechanism is based on the order of the sorted population. Then, the selected individual changes its mutation rate based on the same self-adapting mutation rate strategy and is bit-wisely flipped with the probability of a new mutation rate. To describe the above processes, we summarise a framework of self-adaptive EAs (Algorithm 1). In this framework, we can customise the sorting mechanism $Sort$, the selection mechanism P_{sel} and the self-adapting mutation rate strategy D_{mut}. The MOSA-EA can also be based on this framework. Similarly to non-elitist EAs, we can also consider that each individual in P_{t+1} is sampled from a distribution $D(P_t)$ in Theorem 1 which corresponds to Lines 4-6 of Algorithm 1.

Algorithm 1 Framework of self-adaptive EAs

Require: Fitness function f. Population sizes $\lambda \in \mathbb{N}$. Sorting mechanism $Sort$. Selection mechanism P_{sel}. Self-adapting mutation rate strategy D_{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 χ'/n.
7: Set $P_{t+1}(i) := (x', \chi'/n)$.

3.2 Sorting mechanism

The MOSA-EA aims to maximise fitness and mutation rates simultaneously. To achieve this goal, we apply a multi-objective sorting mechanism to sort the population before producing the next population, as shown in Algorithm 2. Specifically, the multi-objective sorting mechanism uses the strict non-dominated sorting (Algorithm 3) based on maximising two objective functions, i.e., the fitness value of the solution $f_1(x, \chi) := f(x)$ and the mutation rate of the individual $f_2(x, \chi) := \chi$.

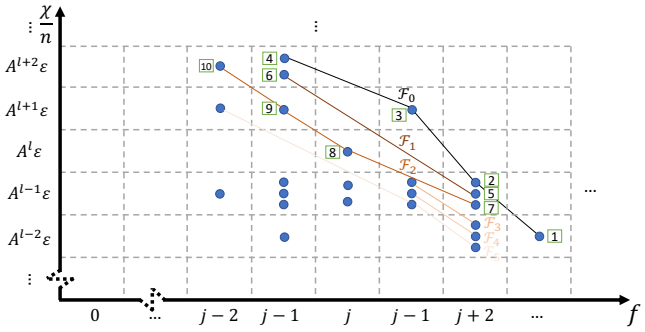

Figure 2: Illustration of multi-objective sorting

After multi-objective sorting described in Algorithm 2, we can get a series of strict non-dominated fronts $\mathcal{F}_0^t, \mathcal{F}_1^t, \ldots$ of P_t. In each strict non-dominated Pareto front, Algorithm 3 guarantees that there do not exist two individuals with the same fitness value and mutation rate, and individuals are put in a fitness increasing sequence. Note that we assume that the aim is to maximise all objectives in Algorithm 3, and we say an individual a dominates another individual b, written as $a \prec b$ if (1) the objective values of a are no lower than b for all functions, and (2) at least one objective value of a is strictly higher than that objective value in b. Figure 2 illustrates an example of the order of a population after multi-objective sorting. Note that the points in the same cell have the same fitness value and the same mutation rate.

The main difference between the fast non-dominated sorting algorithm used in NSGA-II [14] and the strict non-dominated sorting algorithm shown in Algorithm 3 is that the latter can avoid too many similar or copies of the same individual, i.e., same objective values in all functions, in one front, by Lines 9-11. This characteristic can avoid the situation that some individuals dominate the population, i.e., occupying the top positions of the sorted population. In other words, the strict non-dominated sorting algorithm can increase the diversity of the population.

After calculating the strict non-dominated fronts, we sort the individuals in each front based on the fitness function f (Line 3 in Algorithm 2), since the priority of optimisation is to find better solutions. Then we get a sorted population, in which any individual a is ranked before an individual b if a has a higher fitness value or a higher mutation rate than b.

Algorithm 2 Multi-objective sorting mechanism

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 $\mathcal{F}_0^t, \mathcal{F}_1^t, \ldots$ based on $f_1(x, \chi) := f(x)$ and $f_2(x, \chi) := \chi$
2: **for** $\mathcal{F} = \mathcal{F}_0^t, \mathcal{F}_1^t, \ldots$ **do**
3: Sort $\mathcal{F}$ such that $f_1(\mathcal{F}(1)) > f_1(\mathcal{F}(2)) > \ldots$.
4: $P_t := \left(\mathcal{F}_0^t, \mathcal{F}_1^t, \ldots\right)$.
5: **return** P_t.

Algorithm 3 Strict non-dominated sorting

Require: Population sizes $\lambda \in \mathbb{N}$. Population $P \in \mathcal{Z}^\lambda$, where $\mathcal{Z}$ is a finite state space. Objective functions $f_1, f_2, \ldots : \mathcal{Z} \to \mathbb{R}$ (assume to maximise all objective functions).

1: **for** each individual $P(i)$ **do**
2: Set $S_i := \emptyset$ and $n_i := 0$
3: **for** $i = 1, \ldots, \lambda$ **do**
4: **for** $j = 1, \ldots, \lambda$ **do**
5: **if** $P(i) \prec P(j)$ based on $f_1, f_2, \ldots$ **then**
6: $S_i := S_i \cup \{P(i)\}$,
7: **else if** $P(j) \prec P(i)$ based on $f_1, f_2, \ldots$ **then**
8: $n_i := n_i + 1$,
9: **else if** $f_\ell(P(i)) = f_\ell(P(j))$ where $\ell = 1, 2, \ldots$ **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** $\mathcal{F}_0, \mathcal{F}_1, \ldots$.

Algorithm 4 Self-adapting mutation rate strategy

Require: Parameters $A > 1$, $\varepsilon > 0$ and $p_{\text{inc}} \in (0, 1)$. Mutation parameter χ.

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** χ'.

Algorithm 5 Self-adapting mutation rate strategy (theoretical)

Require: Parameters $A > 1$, $\varepsilon > 0$, $p_{\text{inc2}} > 0$, $p_{\text{inc}} > 0$ and $p_{\text{inc}} + p_{\text{inc2}} < 1$. Mutation parameter χ.

1: $\chi' = \begin{cases} \min(A\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor}) \text{ with probability } p_{\text{inc}}, \\ \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor} \text{ with probability } p_{\text{inc2}}, \\ \max(\chi/A, \varepsilon n) \text{ otherwise.} \end{cases}$
2: **return** χ'.

3.3 Self-adapting mutation rate

Self-adapting mutation rate is the key step of self-adaptive EAs. Case and Lehre [5] and Doerr et al.[17] use a strategy to increase the mutation rate by a multiplicative factor $A > 1$ with some probability p_{inc}, decrease otherwise. We apply a similar strategy in Algorithm 4 on the MOSA-EA in experiments (Section 6), where adapting mutation rate from ε to nearly $1/2$. To make theoretical analysis easier, we adopt Algorithm 5 on the MOSA-EA in Theorem 5. The difference is that Algorithm 5 has an additional tiny probability p_{inc2} to increase the mutation rate to the highest of possible mutation rates.

Finally, we say the MOSA-EA is Algorithm 1 using the multi-objective sorting mechanism (Algorithm 2). In this paper, we only consider the (μ, λ) MOSA-EA with using comma selection (Algorithm 8). For self-adapting mutation rate strategy, we apply Algorithms 4 and 5 in Sections 6 and 5, respectively.

4 INEFFICIENCY OF FIXED MUTATION RATE

The $(\mu + \lambda)$ EA and the (μ, λ) EA are proved inefficient when optimising the BBFunnel function which belongs to the SparseLocalOpt problem class [8, 9]. We use similar proof ideas to prove the inefficiency on PeakedLO$_{m,k}$ for $(\mu + \lambda)$ EA and (μ, λ) EA, which is shown in Theorems 2 and 3, respectively. Furthermore, the tournament EA can solve SparseLocalOpt problems with very "sparse" local optima [8, 9]. In Theorem 4, we show the 2-tournament EA with any fixed mutation rate cannot handle a too sparse local optimum on PeakedLO$_{m,k}$. Due to the page limit, the proofs in this section are omitted and can be found in Appendix D.

Theorem 2. *The expected runtime of the $(\mu + \lambda)$ EA with $\lambda, \mu \in \text{poly}(n)$, $\lambda/\mu \geq 1$, initial population $P_0 = \{0^k *^{n-k}\}^\mu$, and mutation parameter $\chi \in O(1)$ on PeakedLO$_{m,k}$ with any $k \leq n$ and $k, m \in \Omega(n)$ satisfies $\Pr(T \leq e^{cn^d}) \leq e^{-\Omega(n)}$ for some constants $c, d > 0$.*

Theorem 3. *For any constant $\delta > 0$, the (μ, λ) EA with $\lambda, \mu \in \text{poly}(n)$, $\lambda/\mu = \alpha_0$ where $\alpha_0 > 1$ is a constant, mutation parameter $0 < \chi \notin [\ln(\lambda/\mu) - \delta, \ln(\lambda/\mu) + \delta]$, and initial population $P_0 = \{0^k *^{n-k}\}^\lambda$ has runtime $\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$ on PeakedLO$_{m,k}$ with any $k \leq n$ and $k, m \in \Omega(n)$ for some constant $c > 0$.*

Theorem 4. *For some constant $\xi \in (0, 2/3)$, any $k \leq$ an where $a = \ln(3(1-\xi)/2)$ is a constant and $m \in \Omega(n)$, the runtime of the 2-tournament EA with population size $\lambda \in \text{poly}(n)$ on PeakedLO$_{m,k}$ with the initial population $P_0 = \{0^k *^{n-k}\}^\lambda$ and any fixed mutation rate satisfies $\Pr(T \leq e^{cn}) = e^{-\Omega(\lambda)}$ for a constant $c > 0$.*

5 EFFICIENCY OF MOSA-EA

We now analyse the runtime of the MOSA-EA on PeakedLO$_{m,k}$ (Theorem 5). In a previous study, Case and Lehre [5] derived an upper bound on the runtime of the (μ, λ) self-adaptive EA on an unknown structure version of LeadingOnes function via the level-based theorem [7]. They first divide the search space $\mathcal{Y}$ into a two-dimensional level partition including fitness levels and mutation rate sub-levels. Then they define two threshold values $\theta_1(j)$ and $\theta_2(j)$ which is an ideal range of mutation rate to improve the solution for each fitness level. Informally, they count the number of generations to increase the mutation rate to enter this ideal mutation rate range, and the number of generations to improve the solution to the next fitness level if with an ideal mutation rate.

Theorem 5. *For some constant $\delta \in (0, 1)$, Algorithm 1 using the multi-objective sorting mechanism (Algorithm 2), self-adapting mutation rate strategy described in Algorithm 5 and the (μ, λ) selection (Algorithm 8) with $\frac{\lambda}{\mu} = \alpha_0 \geq 4$ and $c \log^2(n) \leq \lambda \in \text{poly}(n)$ where α_0 is a constant and c is a large enough constant, has expected runtime $O\left(n\lambda \log(n) \log(\log(n)) + n^2 \log(n)\right)$ on PeakedLO$_{m,k}$ started with any initial population, if satisfying*

- *$p_{\text{inc}} \in \left(\frac{1+\delta}{\alpha_0}, \frac{2}{5}\right)$, $A > (1 + \sqrt{1/(\alpha_0(1-p_{\text{inc}}))})$ are constants,*
- *$\varepsilon = \frac{b}{n}$, $p_{\text{inc}2} = \frac{d}{n}$ for any small constants $b, d > 0$,*
- *$\lceil m \rceil < \left(\ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right)\right) / \left(2A \ln\left(\frac{\alpha_0}{1-\delta}\right)\right) k - \ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right) + 1$, where $k =$ an for any constant $a \in (0, 1]$.*

Although the MOSA-EA analysed in Theorem 5 is significantly different from the (μ, λ) self-adaptive EA, e.g., different sorting mechanisms and different self-adapting mutation rate strategies, we can use a similar proof idea for these two varieties of LeadingOnes. We first divide the search space $\mathcal{Y}$ into regions based on k and m of PeakedLO$_{m,k}$. Then we partition each region into fitness levels and mutation rate sub-levels. We formally define these in Section 5.1. Section 5.2 defines some threshold values and functions, similaly to [5], and we also introduce some useful lemmas. Finally, in Section 5.3, we apply the level-based theorem (Theorem 1) to the level partition to get an upper bound of runtime on PeakedLO$_{m,k}$.

5.1 Partitioning the search space into levels

We partition the two-dimensional search space $\mathcal{Y} = \mathcal{X} \times [\varepsilon, 1/2]$ into "levels". We first divide the search space $\mathcal{Y}$ into three parts A', A and B_k based on fitness value and mutation rate, which are coloured by yellow, blue and grey in Figure 3, respectively. Note that Figure 3 is an informal illustration since the formal definitions are complicated. Formally, we define the regions with the PeakedLO$_{m,k}$ parameters k, m and a threshold value θ_2' as

- $A'_j := \{(x, \chi) \mid \text{LO}(x) = j \text{ and } x \neq 0^k *^{n-k} \text{ and } \chi < \theta_2'\}$ for $j \in [0..\lceil m \rceil - 1]$;
- $A'_{\lceil m \rceil} := \{(x, \chi) \mid x = 0^k *^{n-k} \text{ and } \chi < \theta_2'\}$;
- $A_j := \{(x, \chi) \in \mathcal{X} \mid \text{LO}(x) = j \text{ and } x \neq 0^k *^{n-k} \text{ and } \chi \geq \theta_2'\}$ for $j \in [0..\lceil m \rceil - 1]$;
- $A_j := \{(x, \chi) \mid \text{LO}(x) = j\}$, for $j \in [\lceil m \rceil..n]$;
- $B_k := \{(x, \chi) \mid x = 0^k *^{n-k} \text{ and } \chi \geq \theta_2'\}$.

We will define θ_2' in the next subsection. Note that B_k contains no level which is applied in the level-based theorem, and we will prove

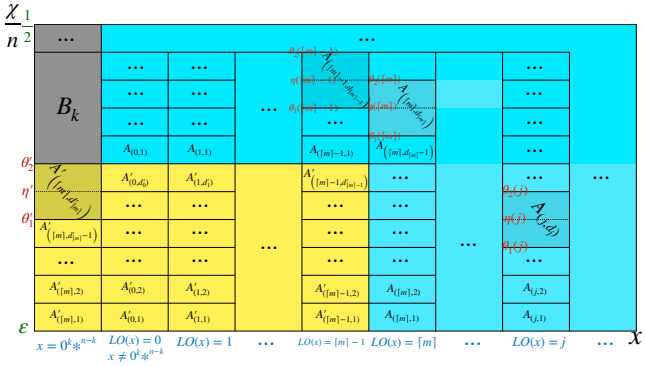

Figure 3: Informal illustration of the level partition on PeakedLO$_{m,k}$ function (Regions A', A, B_k are coloured by yellow, blue, grey, respectively)

To describe the ranking of sub-levels, we define that any level in A is higher than all levels in A', any sub-level in A_j is higher than all sub-levels in A_{j-1}, and any sub-level in A'_j is higher than all sub-levels in A'_{j-1} for $j \geq 1$.

Now, we define sub-levels in regions A' and A. For region A', we first define the *depth* d'_j of levels A'_j:

- For $j \in [0..\lceil m \rceil - 1]$, $d'_j := \min\{\ell \in \mathbb{N} \mid \varepsilon A^\ell \geq \theta_2'\}$;
- For $j = \lceil m \rceil$, $d'_{\lceil m \rceil} := \min\{\ell \in \mathbb{N} \mid \varepsilon A^\ell \geq \theta_1'\}$.

The depth of level in A' implies the number of sub-levels to enter higher region, i.e., A. Then,

- For $j \in [0..\lceil m \rceil - 1]$, we define the sub-levels for $\ell \in [d'_j]$ as $A'_{(j,\ell)} := A'_j \times [\varepsilon A^{\ell-1}, \min(\varepsilon A^\ell, \theta_2'))$,
- For $j = \lceil m \rceil$, we define the *low levels* for $\ell \in [d'_j - 1]$ as $A'_{(\lceil m \rceil, \ell)} := A'_{\lceil m \rceil} \times [\varepsilon A^{\ell-1}, \min(\varepsilon A^\ell, \theta_1'))$, and we define the *edge level* as $A'_{(\lceil m \rceil, d'_{-1})} := A'_{\lceil m \rceil} \times [\theta_1', \min(\frac{1}{2}, \theta_2'))$.

To define sub-levels in region A_j, we use two threshold values $\theta_1(j)$ and $\theta_2(j)$ which will be defined in the following subsection. Similarly to [5], $\theta_1(j)$ and $\theta_2(j)$ imply the ideal range of mutation rate to mutate the solution to level A_j. We also begin at defining the *depth* d_j of levels A_j,

- For $j \in [0..\lceil m \rceil - 1]$, $d_j := \min\{\ell \in \mathbb{N} \mid \theta_2' A^\ell \geq \theta_1(j)\}$;
- For $j \in [\lceil m \rceil..n - 1]$, $d_j := \min\{\ell \in \mathbb{N} \mid \varepsilon A^\ell \geq \theta_1(j)\}$.

The depth of level implies the number of sub-levels to tune the mutation rate from the lowest to an ideal mutation rate.

- For $j \in [0..\lceil m \rceil - 1]$, we define the *low levels* as $A_{(j,\ell)} := A_j \times [\theta_2' A^{\ell-1}, \min(\theta_2' A^\ell, \theta_1(j)))$, for $\ell \in [d_j - 1]$, and define the *edge levels* as $A_{(j,d_j)} := A_j \times [\theta_1(j), \min(\frac{1}{2}, \theta_2(j))] \cup A_{>j} \times (\min(\frac{1}{2}, \theta_2(j+1)), \min(\frac{1}{2}, \theta_2(j))]$.
- For $j \in [\lceil m \rceil..n - 1]$, we define the *low levels* as $A_{(j,\ell)} := A_j \times [\varepsilon A^{\ell-1}, \min(\varepsilon A^\ell, \theta_1(j)))$, and we define the *edge levels* as $A_{(j,d_j)} := A_j \times [\theta_1(j), \min(\frac{1}{2}, \theta_2(j))] \cup A_{>j} \times (\min(\frac{1}{2}, \theta_2(j+1)), \min(\frac{1}{2}, \theta_2(j))]$.

- We let A_n contain only one sub-level $A_{(n,1)} := A_j \times [\varepsilon, 1/2]$.

5.2 Definitions and useful lemmas

We now define the functions θ_1, η and θ_2 for levels A_j where $j \in [0..n-1]$, and the values θ_1', η' and θ_2' for levels $A'_{\lceil m \rceil}$, which use in the levels definitions above.

- For A_j where $j \in [n-1]$, let

$$\eta(j) := \frac{1}{2A}\left(1 - \left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right)^{\frac{1}{j}}\right), \theta_1(j) := \frac{\eta(j)}{A}, \theta_2(j) := 1 - q^{\frac{1}{j}}, \text{ where}$$

$$q := \frac{1-\zeta}{\alpha_0},\ r_0 := \frac{1+\delta}{\alpha_0(1-p_{\text{inc}}-p_{\text{inc2}})},\ \zeta := 1-\alpha_0(r_0)^{1+\sqrt{r_0}}$$

- For A_0, let $\eta(0)$ be any function such that $\eta(0) = \frac{\eta(1)}{A} - o(1)$, and we define $\theta_1(0) := \frac{\eta(0)}{A}$, and $\theta_2 := \frac{\theta_2(1)}{A}$.
- For $A'_{\lceil m \rceil}$, we define $\eta' := \eta(k)$, $\theta_1' := \theta_1(k)$ and $\theta_2' := \theta_2(k)$.

For convenience, let $x' \sim p_{\text{mut}}(x, \chi)$ denote that individual x' is sampled by independently flipping each bit of x with probability χ/n, which is another expression of Line 6 in Algorithm 1. Then, for all individuals (x, χ) in A_j where $j \in [0..n-1]$ and $\chi \in [\varepsilon n, n/2]$, we define the *survival probability* as $r(j, \chi) := \min_{x \in A_j} \Pr_{x' \sim p_{\text{mut}}(x,\chi)}(x' \in A_{\geq j})$, and for all individuals (x, χ) in A'_j, where $j \in [0..\lceil m \rceil]$ and $\chi \in [\varepsilon n, n/2]$, we define the *survival probability* as $r'(j, \chi) := \min_{x \in A'_j} \Pr_{x' \sim p_{\text{mut}}(x,\chi)}(x' \in A'_{\geq j})$.

We then explain that condition $\lceil m \rceil < \left(\ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right)\right)/\left(2A\ln\left(\frac{\alpha_0}{1-\delta}\right)\right)k - \ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right) + 1$ in Theorem 5 essentially implies $\theta_2' < \theta_1(j)$ for all $j \in [0..\lceil m \rceil - 1]$ by Lemma 1 (the proof can be found in Appendix D.4). Informally, it means that the local optima is "sparse" and the fitness valley is "dense".

Lemma 1. *Assume that the parameters A, α_0 and p_{inc} satisfy the constraints in Theorem 5. For any constant $\delta \in (0,1)$, if $\lceil m \rceil < \left(\ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right)\right)/\left(2A\ln\left(\frac{\alpha_0}{1-\delta}\right)\right)k - \ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right) + 1$, where $k = an$ for any constant $a \in (0,1]$, then $\theta_2' < \theta_1(j)$ for all $j \in [0..\lceil m \rceil - 1]$.*

Then we introduce three lemmas to support the proofs for conditions (G2) and (G1) of Theorem 1. Due to the page limit, the lemmas and their proofs can be found in Appendices D.5, D.6 and D.7. Lemma 6 presents some useful inequalities, and Lemmas 7 and 8 can be used to prove conditions (G2) and (G1), respectively. The lemmas and the proofs may look similar to [5], but they are separate statements since different algorithms and level partitions are considered in Theorem 5 and [5].

Too many individuals with high fitness but incorrect mutation rate would ruin the progress of the population. We therefore need to prove that there are not too many such "bad" individuals in the population. We first define a "bad" region $B \subset \mathcal{Y}$ containing search points with a mutation rate that is too high, and we say an individual $(x, \chi/n) \in B$ has too high mutation rate. For the constant $\zeta \in (0,1)$, let

$$B := \Big\{(x, \chi/n) \in A_j \times [\varepsilon, 1/2] \mid$$

$$(j \in [0..n-1] \ \wedge\ \forall y \in \mathcal{X} \backslash \{0^k * ^{n-k}\} \Pr_{x' \sim p_{\text{mut}}(y,\chi)}(x' \in A_{\geq j}) < \frac{1-\zeta}{\alpha_0})$$

$$\vee\ (\forall y = 0^k *^{n-k} \Pr_{x' \sim p_{\text{mut}}(y,\chi)}(x' = 0^k *^{n-k}) < \frac{1-\zeta}{\alpha_0})\Big\}. \quad (1)$$

By $\theta_2(j)$, θ_2' and B_k, the region B can also be expressed as

$$B = \cup_{j=0}^{n-1} A_{>j} \times (\min(1/2, \theta_2(j+1)), \min(1/2, \theta_2(j))] \cup B_k. \quad (2)$$

We show that too many such individuals in the population is rare by Lemma 9 (shown in Appendix D.8).

5.3 Applying the level-based theorem

Now, we use Lemmas 6, 7, 8 and 9 to prove Theorem 5 via Theorem 1.

Proof (Theorem 5). We say that a generation t is "failed" if the population P_t contains more than $(1-\zeta/2)\mu$ individuals in region B. We will optimistically assume that no generation fails. Under this assumption, we will prove that the conditions of Theorem 1 hold, leading to an upper bound on the expected number of function evaluations $t_0(n)$ until a search point in $A_{(n,1)}$ is created. In the end we will use a restart argument to account for failed generations.

Each strict non-dominated front has at most $\lceil \log_A(1/2\varepsilon) \rceil = c' \log(n)$ individuals where $c' > 0$ is some constant, so there are at least $\lambda/(c'\log(n)) \in \Omega(\log(n))$ fronts. Let $c' := \lceil \log_A\left(\frac{n}{2\varepsilon}\right) \rceil / \log(n)$, where $c'\log(n)$ is the maximal number of individuals in a front. Let $\gamma_0 := b'/\log(n)$ for some constant $0 < b' < \zeta/(2\alpha_0 c')$. By the assumption, the number of individuals are not in region B and have chance to be selected is at least $\zeta\mu/2$. Since $\gamma_0\lambda = b'\lambda/\log(n) < \zeta\lambda/(2\alpha_0 c'\log(n)) = \zeta\mu/(2\log(n)) < \zeta\mu/2$, therefore any individual ranked in the first $\gamma_0\lambda$ positions will be selected with probability $\frac{1}{\mu}$ by the (μ, λ) selection.

By the ranking mechanism, if an individual (x, χ) ranked in the first $\gamma\lambda$ positions where $\gamma \in (0, \gamma_0]$, then any individual (x', χ') where either $f(x') > f(x)$ or $\chi' > \chi$ will be ranked in the first $\gamma\lambda$ positions, which guarantees individuals in $A'_{>(j,\ell)}$ are ranked in the first $\gamma\lambda$ positions if there are at least $\gamma\lambda$ individuals in $A'_{\geq(j,\ell)}$, and guarantees individuals in $A_{>(j,\ell)}$ are ranked in the first $\gamma\lambda$ positions if there are at least $\gamma\lambda$ individuals in $A_{\geq(j,\ell)}$.

Now we consider (G1)-(G2) of Theorem 1 in regions A' and A.

Region A': For $\gamma \in (0, \gamma_0]$, if there are $\gamma\lambda$ individuals in level $A'_{>(j,\ell)}$ for some $j \in [0..\lceil m \rceil]$ and $\ell \in [d_j']$, then the probability of selecting an individual from $A'_{>(j,\ell)}$ is $\gamma\alpha_0$. Since $|P_t \cap B| \leq (1-\zeta/2)\mu$, it follows that all $\gamma\lambda < (\zeta/2)\mu$ individuals of $A'_{\geq(j,\ell)}$ are among the μ fittest in the population. Therefore, the probability of selecting an individual from $A'_{>(j,\ell)}$ indeed is $\gamma\lambda/\mu = \gamma\alpha_0$. We assume that the current population has at least $\gamma_0\lambda$ individuals in levels $A'_{\geq(j,\ell)}$. To verify condition (G2), we must estimate the probability of producing an offspring in levels $A'_{>(j,\ell)}$, assuming that there are at least $\gamma\lambda$ individuals in levels $A'_{>(j,\ell)}$, for any $\gamma \in (0, \gamma_0]$. We distinguish five cases:

- Case 1: $j \in [0..\lceil m \rceil - 1]$ and $\ell \in [d_j' - 1]$. Assuming that the parent $(x, \chi/n)$ is in $A'_{(u,v)} \subseteq A'_{\geq(j,\ell+1)}$. Since $\chi/n < \theta_2' < \theta_1(j) < \eta(j)$ by θ^2, η and Lemma 1, it is "safe" to increase the mutation rate. For a lower bound, we therefore pessimistically only account for offspring where the mutation parameter is increased to $\min(\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor})$. The probability of producing an offspring in levels $A'_{\geq(j,\ell+1)}$ is at least $p_{\text{inc}}(1-A\chi/n)^j > p_{\text{inc}}(1-A\theta_2')^j > p_{\text{inc}}(1-A\eta(j))^j \geq (1+\delta)/\alpha_0$.

- Case 2: $j \in [0..\lceil m \rceil - 1]$ and $\ell = d'_j$. We use a stronger assumption that there are at least $\gamma\lambda$ individuals in $A_{\geq(j,1)}$ for any $\gamma \in (0, \gamma_0]$, which is a subset of $A'_{>(j,d'_j)}$. To produce an offspring in levels $A_{\geq(j,1)}$, it suffices to first select a parent from $A_{\geq(j,1)}$, and secondly create an offspring in levels $A_{\geq(j,1)}$. The probability of selecting such a parent is at least $\gamma\alpha_0$. By Lemma 7, the probability of producing an offspring in levels $A_{\geq(j,1)}$ is at least $(1+\delta)/\alpha_0$.
- Case 3: $j = \lceil m \rceil$ and $\ell \in [d'_j - 2]$. Similarly to Case 1, it is "safe" to increase the mutation rate, since $\chi/n < \eta'$. For a lower bound, we therefore pessimistically only account for offspring where the mutation parameter is increased to $\min(A\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor})$. The probability of producing an offspring in levels $A'_{\geq(\lceil m \rceil, \ell+1)}$ is at least $1 + \delta/\alpha_0$.
- Case 4: $j = \lceil m \rceil$ and $\ell = d'_j - 1$. By Lemma 7, the probability of producing an offspring in levels $A'_{\geq(\lceil m \rceil, d'_{\lceil m \rceil})}$ is at least $\frac{1+\delta}{\alpha_0}$.
- Case 5: $j = \lceil m \rceil$ and $\ell = d'_{\lceil m \rceil}$. We use a stronger assumption that there are at least $\gamma\lambda$ individuals in $A_{\geq(0,1)}$ for any $\gamma \in (0, \gamma_0]$, which is a subset of $A'_{>(\lceil m \rceil, d'_{\lceil m \rceil})}$. The probability of producing an offspring in levels $A_{\geq(0,1)}$ is at least $(1+\delta)/\alpha_0$.

To produce an offspring in levels $A'_{>(j,\ell)}$, it suffices to firstly select a parent $(x, \chi/n)$ from $A'_{>(j,\ell)}$, and then create an offspring $(x', \chi'/n)$ in levels $A'_{>(j,\ell)}$. The probability of selecting such a parent is at least $\gamma\alpha_0$. Thus, the probability that the offspring $(x', \chi'/n)$ in levels $A'_{>(j,\ell)}$, is at least $\gamma\alpha_0 \frac{1+\delta}{\alpha_0} = \gamma(1+\delta)$, which (G2) is satisfied.

For condition (G1) of Theorem 1, we assume that there are $\gamma_0\lambda$ individuals in P_t in $A'_{\geq(j,\ell)}$ where $j \in [0..\lceil m \rceil]$ and $\ell \in [d'_j]$. To verify condition (G1), we assume that the parent $(x, \chi/n)$ is in $A'_{(j,\ell)}$, and we distinguish four cases:

- Case 1: $j \in [0..\lceil m \rceil - 1]$ and $\ell \in [d'_j - 1]$. The probability of the offspring in levels $A'_{\geq(j,\ell+1)}$ is at least $p_{\text{inc}}(1-\chi/n)^{\lceil m \rceil - 1} = \Omega(1)$.
- Case 2: $j \in [0..\lceil m \rceil - 1]$ and $\ell = d'_j$. The probability of the offspring in region A, i.e., $A_{\geq(j,1)}$, is at least $p_{\text{inc}}(1-\chi/n)^{\lceil m \rceil - 1} = \Omega(1)$.
- Case 3: $j = \lceil m \rceil$ and $\ell \in [d'_j - 1]$. The probability of the offspring in levels $A'_{\geq(\lceil m \rceil, \ell+1)}$ is at least $p_{\text{inc}}(1-\chi/n)^{\lceil m \rceil - 1} = \Omega(1)$.
- Case 4: $j = \lceil m \rceil$ and $\ell = d'_{\lceil m \rceil}$. The probability of the offspring in levels $A_{\geq(0,d_0)}$ is at least $p_{\text{inc2}}(1-(1-1/2)^{\lceil m \rceil - 1}) = \Omega(1/n)$, where the mutation parameter is changed to $\min(\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor})$ and at least one 0-bit of first k bit-position is flipped.

Thus, a lower bound of the probability of producing an offspring in higher levels, i.e., $A'_{>(j,\ell)}$, is $\gamma_0\alpha_0\Omega(1/n) = \Omega(1/n\log(n)) =: z'(\lceil m \rceil, d'_{\lceil m \rceil})$ and $\gamma_0\alpha_0\Omega(1) = \Omega(1/\log(n)) =: z'(j,\ell)$ except the case of $j = \lceil m \rceil$ and $\ell = d'_{\lceil m \rceil}$.

Region A: To verify condition (G2), we distinguish two cases for all $j \in [0..n-1]$ and $\ell \in [d_j]$:

- Case 1: $\ell \in [d_j - 1]$. Firstly, we need to estimate the probability of producing an offspring in levels $A_{\geq(j,\ell+1)}$, assuming that there are at least $\gamma\lambda$ individuals in levels $A_{\geq(j,\ell+1)}$, for any $\gamma \in (0, \gamma_0]$. To produce an offspring in levels $A_{\geq(j,\ell+1)}$, it suffices to first select a parent $(x, \chi/n)$ from $A_{\geq(j,\ell+1)}$, and secondly create an offspring $(x', \chi'/n)$ in levels $A_{\geq(j,\ell+1)}$. The probability of selecting such a parent is at least $\gamma\alpha_0$. Assuming that the parent is in level $A_{(u,v)} \subseteq A_{\geq(j,\ell+1)}$, and applying Lemma 7 to level $A_{(u,v)}$, the probability that the offspring $(x', \chi'/n)$ is in levels $A_{(u,v)} \subseteq A_{\geq(j,\ell+1)}$ is $(1+\delta)/\alpha_0$ for some $\delta \in (0,1)$. Thus the probability of selecting a parent in levels $A_{\geq(j,\ell+1)}$, then producing an offspring in levels $A_{\geq(j,\ell+1)}$, is at least $\gamma\alpha_0(1+\delta)/\alpha_0 = \gamma(1+\delta)$, so condition (G2) is satisfied.
- Case 2: $\ell = d_j$. We assume that there are at least $\gamma\lambda$ individuals in levels $A_{\geq(j+1,1)}$, for $\gamma \in (0, \gamma_0]$. We again apply Lemma 7 to show the probability of selecting an individual from $A_{\geq(j+1,1)}$ and producing a new individual also in $A_{\geq(j+1,1)}$ is at least $\gamma(1+\delta)$, showing condition (G2) is satisfied.

To verify condition (G1), we assume that there are $\gamma_0\lambda$ individuals in P_t in $A_{\geq(j,\ell)}$ where $j \in [0..n-1]$ and $\ell \in [d_j]$. If the parent $(x, \chi/n)$ is in $A_{\geq(j,\ell)}$, then

- Case 1: $\ell \in [d'_j - 1]$. The probability of the offspring in levels $A_{\geq(j,\ell+1)}$ is at least $(1+\delta)/\alpha_0 = \Omega(1)$.
- Case 2: $\ell = d'_j$. By Lemma 8, the probability of the offspring in $A_{\geq(j+1,1)}$ is at least $\Omega(1/j)$.

Thus, a lower bound of the probability of producing an offspring in higher levels, i.e., $A_{(j,d_j)}$ is $\gamma_0\alpha_0\Omega(1) = \Omega(1/\log(n)) =: z(j, d_j)$ for $j \in [0..\lceil m \rceil]$, and $\gamma_0\alpha_0\Omega(1/j) = \Omega(1/j\log(n)) =: z(j,\ell)$ for $j \in [0..\lceil m \rceil]$ and $\ell \in [d_j - 1]$.

To verify that $\lambda \geq c\log^2(n)$ is large enough to satisfy condition (G3), we first must calculate the number of total sub-levels m'. The depth of each level j is no more than $d'_j < \lceil \log_A(\frac{n}{2\varepsilon}) \rceil = O(\log(n))$ for all $j \in [0..\lceil m \rceil]$ and $d_j < \lceil \log_A(\frac{n}{2\varepsilon}) \rceil = O(\log(n))$ for all $j \in [0..n-1]$. Therefore, by $\gamma_0 = \Omega(1/\log(n))$, $m' = O(n\log(n))$ and $z_{\min} = \Omega(1/n\log(n))$, we know that $\lambda \geq c\log^2(n)$ satisfies condition (G3) for a large enough $c > 1$.

Overall, assuming no failure, the expected time to reach the last level can be calculated by considering regions A' and A separately, which is no more than

$$
\begin{aligned}
t_0(n) \leq \frac{8}{\delta^2}\Bigg(& \sum_{j=0}^{\lceil m \rceil - 1} \sum_{\ell=1}^{d'_j} \left(\lambda \log\left(\frac{6\delta\lambda}{4 + z'_{(j,\ell)}\delta\lambda} \right) + \frac{1}{z'_{(j,\ell)}} \right) \\
& + \sum_{\ell=1}^{d'_j - 1} \left(\lambda \log\left(\frac{6\delta\lambda}{4 + z'_{(\lceil m \rceil,\ell)}\delta\lambda} \right) + \frac{1}{z'_{(\lceil m \rceil,\ell)}} \right) \\
& + \lambda \log\left(\frac{6\delta\lambda}{4 + z'_{(\lceil m \rceil, d'_j)}\delta\lambda} \right) + \frac{1}{z'_{(\lceil m \rceil, d'_j)}} \\
& + \sum_{j=0}^{n-1} \sum_{\ell=1}^{d_j - 1} O\left(\lambda \log\left(\frac{1}{z_{(j,\ell)}} \right) + \frac{1}{z_{(j,\ell)}} \right) \\
& + \sum_{j=0}^{n-1} O\left(\lambda \log(\lambda) + \frac{1}{z_{(j,d_j)}} \right) \Bigg) \\
= & O(n\lambda \log(n) \log(\log(n)) + n^2 \log(n)).
\end{aligned}
$$

Finally, we account for "failed" generations where our assumption that there are less than $(1-\zeta/2)\mu$ individuals in region B does not hold. We refer to a sequence of $2t_0(n)/\lambda$ generations as a *phase*, and call a phase *good* if for $2t_0(n)/\lambda$ consecutive generations there are fewer than $(1-\zeta/2)\mu$ individuals in region B. By Lemma 9 and a union bound, a phase is good with probability $1 - 2t_0(n)/\lambda e^{-\Omega(\mu)} = \Omega(1)$, for $\mu = \Omega(\log(n))$. By Markov's inequality, the probability of reaching a global optimum in a good phase is

at least 1/2. Hence, the expected number of phases required, each costing $2t_0(n)$ function evaluations, is $O(1)$. □

6 EXPERIMENTS

The theoretical analysis considers an idealised scenario, and it is interesting to evaluate its performance on problems with more complex structure. In this section, we empirically analyse the performance of the MOSA-EA on two well-known combinatorial optimisation problems, random NK-Landscape and k-Sat instances. We also compare the performances of the MOSA-EA with other popular randomised search heuristics.

6.1 Problems

The NK-Landscape problem [20] 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 every $i \in [n]$, to maximise NK-Landscape$(x) := \sum_{i=1}^{n} f_i(\Pi(x,i))$, 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., $x_i, \ldots, x_{(i+k-1) \mod n}$. Typically, each sub-function is defined as a lookup table with 2^{k+1} values between $(0,1)$. We generate 100 random NK-Landscape instances with $n = 100$ for each $k \in \{5, 10, 15, 20, 25\}$ by uniformly sampling values in the lookup table. We run each algorithm once on each instance, and record the highest fitness value achieved in the evaluation budget.

The k-Sat problem is an optimisation problem that aims to find an assignment $\{0,1\}^n$ which maximises the number of satisfied clauses of a given Boolean formula in the conjunctive normal form [1, 6, 18]. For each random k-Sat instance, all m clauses have the same size $k \in [n]$ and are sampled k elements from $[n]$ without replacement. For $k \geq 3$, Coja-Oghlan [4] gives a *threshold for satisfiability* $r_{k\text{-Sat}} = 2^k \ln 2 - \frac{1}{2}(1+\ln 2) + o_k(1)$, where the probability to sample a satisfiable instance is nearly 0 if $\frac{m}{n}$ is close to this threshold. The notation $o_k(1)$ signifies a term that tends to 0 in the limit of large k. We firstly generate 100 random k-Sat instances with $k = 5$ and $m = \lceil (2^k \ln(2) - (1 + \ln(2))/2)n \rceil \approx nr_{k\text{-Sat}}$ and for each $n \in \{100, 200, 300\}$. Similarly to experiments on NK-Landscape, we then run each algorithm among these random instances, and record the minimal numbers of unsatisfied clauses during the algorithm running in the fitness evaluation budget.

6.2 Algorithms

We compare the MOSA-EA with other EAs on the NK-Landscape and k-Sat problems. For the MOSA-EA used in experiments, we use the (μ, λ) selection described in Algorithm 8 and the self-adapting mutation rate strategy shown in Algorithm 4. For the parameter setting, we use $\mu = \lambda/8$, the minimal mutation rate $\varepsilon = 1.0/n$, the self-adapting mutation rate parameters $A = 1.01$ and $p_{\text{inc}} = 0.4$. The initial mutation rates of each individual is set as εA^{ℓ} where $\ell \sim \text{Unif}\left(\left[0..\left\lfloor \log_A(1/2\varepsilon) \right\rfloor\right]\right)$.

The baseline algorithm (1+1) EA can be implemented by the $(\mu + \lambda)$ EA with $\lambda = \mu = 1$. We use the standard mutation rate $\chi/n = 1/n$. For the (μ, λ) EA, we set the population parameter $\mu = \lambda/8$ and the mutation rate a little bit below the error threshold, i.e., $\chi/n = 2.07/n < \ln(\lambda/\mu)$. The UMDA [24] has only two parameters, population parameters λ and μ. Consistently, we set $\mu = \lambda/8$ in experiments. We configure the mutation rate of the 3-tournament EA as $\chi/n = 1.09812/n$ following [8, 9]. For fair comparisons, we use the same population size $\lambda = 20000$ for all population-based algorithms and the same budget of fitness evaluations 10^8.

6.3 Results

Figures 4 and 5 illustrate the experimental results on random NK-Landscape and k-Sat instances, respectively. Each box shows the distribution of one algorithm among 100 random instances on one problem setting. Tables 2 and 3 (shown in Appendix E) show statistics for results on NK-Landscape and k-Sat, respectively. Figure 4 and Table 2 indicate that 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 $k \in \{10, 15, 20, 25\}$. Figure 5 and Table 3 indicate that the number of unsatisfied clauses achieved by the MOSA-EA are statistically significantly less than all other algorithms with significance level $\alpha = 0.05$ for all tested k-Sat problems.

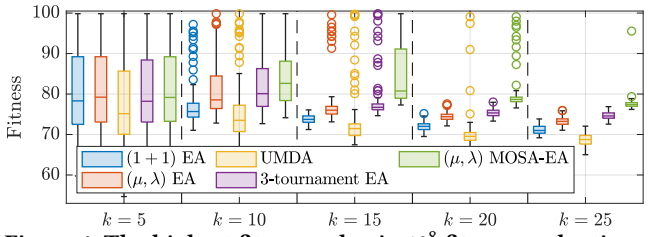

Figure 4: The highest fitness value in 10^8 fitness evaluations on random NK-Landscape instances ($n = 100$)

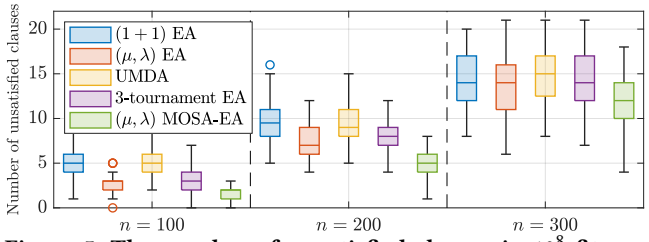

Figure 5: The number of unsatisfied clauses in 10^8 fitness evaluations on random k-Sat instances ($k = 5$)

7 CONCLUSION

In this paper, we first introduce the MOSA-EA for single-objective optimisation, which treats parameter control from the perspective of multi-objective optimisation: The algorithm simultaneously maximises the fitness and the mutation rates. To achieve this, we propose the multi-objective sorting mechanism, which sorts the population based on the strict non-dominated fronts. Theoretically, we demonstrate that the MOSA-EA can escape from sparse local optima of PeakedLO$_{m,k}$ for any $k \in \Omega(n)$, where fixed mutation rate EAs may be trapped. Empirically, we show that the MOSA-EA can outperform some popular random search heuristics on complex combinatorial optimisation problems, i.e., random NK-Landscape and k-Sat problems.

ACKNOWLEDGMENTS

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.

REFERENCES

[1] Dimitris Achlioptas and Cristopher Moore. 2006. Random k-SAT: Two Moments Suffice to Cross a Sharp Threshold. *SIAM J. Comput.* 36, 3 (Jan. 2006), 740–762.

[2] Denis Antipov, Benjamin Doerr, and Vitalii Karavaev. 2022. A Rigorous Runtime Analysis of the $(1 + (\lambda, \lambda))$ GA on Jump Functions. *Algorithmica* (Jan. 2022).

[3] Henry Bambury, Antoine Bultel, and Benjamin Doerr. 2021. Generalized jump functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, Lille France, 1124–1132.

[4] Guy Bresler and Brice Huang. 2021. The Algorithmic Phase Transition of Random k-SAT for Low Degree Polynomials. *arXiv:2106.02129 [math-ph, stat]* (Oct. 2021). arXiv: 2106.02129.

[5] Brendan Case and Per Kristian Lehre. 2020. Self-adaptation in non-Elitist Evolutionary Algorithms on Discrete Problems with Unknown Structure. *IEEE Transactions on Evolutionary Computation* (2020), 1–1.

[6] Amin Coja-Oghlan. 2014. The asymptotic k-SAT threshold. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. ACM, New York New York, 804–813.

[7] Dogan Corus, Duc-Cuong Dang, Anton V. Eremeev, and Per Kristian Lehre. 2018. Level-Based Analysis of Genetic Algorithms and Other Search Processes. *IEEE Transactions on Evolutionary Computation* 22, 5 (Oct. 2018), 707–719.

[8] Duc-Cuong Dang, Anton Eremeev, and Per Kristian Lehre. 2020. Escaping Local Optima with Non-Elitist Evolutionary Algorithms. In *Proceedings of AAAI 2021*. AAAI Press.

[9] Duc-Cuong Dang, Anton Eremeev, and Per Kristian Lehre. 2021. 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.

[10] Duc-Cuong Dang, Tobias Friedrich, Timo Kotzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. 2018. Escaping Local Optima Using Crossover With Emergent Diversity. *IEEE Transactions on Evolutionary Computation* 22, 3 (June 2018), 484–497.

[11] Duc-Cuong Dang and Per Kristian Lehre. 2016. Runtime Analysis of Non-elitist Populations: From Classical Optimisation to Partial Information. *Algorithmica* 75, 3 (July 2016), 428–461.

[12] Duc-Cuong Dang and Per Kristian Lehre. 2016. Self-adaptation of Mutation Rates in Non-elitist Populations. In *Parallel Problem Solving from Nature – PPSN XIV*. Vol. 9921. Springer International Publishing, Cham, 803–813.

[13] Duc-Cuong Dang, Per Kristian Lehre, and Phan Trung Hai Nguyen. 2019. Level-Based Analysis of the Univariate Marginal Distribution Algorithm. *Algorithmica* 81, 2 (Feb. 2019), 668–702.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (April 2002), 182–197.

[15] Benjamin Doerr. 2022. Does Comma Selection Help to Cope with Local Optima? *Algorithmica* (Jan. 2022).

[16] Benjamin Doerr and Timo Kötzing. 2020. Multiplicative Up-Drift. *Algorithmica* (Oct. 2020).

[17] Benjamin Doerr, Carsten Witt, and Jing Yang. 2021. Runtime Analysis for Self-adaptive Mutation Rates. *Algorithmica* 83, 4 (April 2021), 1012–1053.

[18] Jens Gottlieb, Elena Marchiori, and Claudio Rossi. 2002. Evolutionary Algorithms for the Satisfiability Problem. *Evolutionary Computation* 10, 1 (March 2002), 35–50.

[19] Mario Alejandro Hevia Fajardo and Dirk Sudholt. 2021. 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*. ACM, Virtual Event Austria, 1–15.

[20] Stuart A Kauffman and Edward D Weinberger. 1989. The NK model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of theoretical biology* 141, 2 (1989), 211–245. Publisher: Elsevier.

[21] Per Kristian Lehre. 2010. Negative Drift in Populations. In *Parallel Problem Solving from Nature, PPSN XI*. Springer Berlin Heidelberg, Berlin, Heidelberg, 244–253.

[22] Per Kristian Lehre and Xiaoyu Qin. 2021. More Precise Runtime Analyses of Non-elitist EAs in Uncertain Environments. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, Lille, France, 9.

[23] Per Kristian Lehre and Xin Yao. 2012. On the Impact of Mutation-Selection Balance on the Runtime of Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 16, 2 (April 2012), 225–241.

[24] Heinz Mühlenbein and Gerhard Paaß. 1996. From recombination of genes to the estimation of distributions I. Binary parameters. In *International conference on parallel problem solving from nature*. Springer, 178–187.

[25] Constantin P. Niculescu and Andrei Vernescu. 2004. A two sided estimate of $e^x - (1 + x/n)^n$. *Journal of Inequalities in Pure and Applied Mathematics* 5, 3 (2004).

[26] Amirhossein Rajabi and Carsten Witt. 2021. Stagnation detection in highly multimodal fitness landscapes. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, Lille France, 1178–1186.

[27] N. Srinivas and Kalyanmoy Deb. 1994. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation* 2, 3 (Sept. 1994), 221–248.

[28] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. 2013. Exploration and exploitation in evolutionary algorithms: A survey. *Comput. Surveys* 45, 3 (June 2013), 1–33.