Self-adaptation Can Improve the Noise-tolerance of Evolutionary Algorithms

Lehre, Per Kristian; Qin, Xiaoyu

DOI:
10.1145/3594805.3607128

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Lehre, PK & Qin, X 2023, Self-adaptation Can Improve the Noise-tolerance of Evolutionary Algorithms. in *FOGA '23: Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms.* Association for Computing Machinery (ACM), pp. 105-116, Foundations of Genetic Algorithms XVII, Potsdam, Germany, 30/08/23. https://doi.org/10.1145/3594805.3607128

Self-adaptation Can Improve the Noise-tolerance of Evolutionary Algorithms

Per Kristian Lehre*
University of Birmingham
Birmingham, United Kingdom
p.k.lehre@cs.bham.ac.uk

Xiaoyu Qin*
University of Birmingham
Birmingham, United Kingdom
xxq896@cs.bham.ac.uk

ABSTRACT

Real-world optimisation often involves uncertainty. Previous studies proved that evolutionary algorithms (EAs) can be robust to noise when using proper parameter settings, including the mutation rate. However, finding the appropriate mutation rate is challenging if the occurrence of noise (or noise level) is unknown. Self-adaptation is a parameter control mechanism which adjusts mutation rates by encoding mutation rates in the genomes of individuals and evolving them. It has been proven to be effective in optimising unknown-structure and multi-modal problems. Despite this, a rigorous study of self-adaptation in noisy optimisation is missing. This paper mathematically analyses the runtimes of 2-tournament EAs with self-adapting two mutation rates, fixed mutation rates and uniformly chosen mutation rate from two given rates on LeadingOnes with and without symmetric noise. Results show that using self-adaptation achieves the lowest runtime regardless of the presence of symmetric noise. In supplemental experiments, we extend analyses to other types of noise, i.e., one-bit and bit-wise noise. We also consider another self-adaptation mechanism, which adapts the mutation rate from a given interval. Self-adaptive EAs adapt their mutation rate to the noise level and outperform static EAs in these experiments. Overall, self-adaptation can improve the noise-tolerance of EAs in the noise-models studied here.

CCS CONCEPTS

• **Theory of computation → Evolutionary algorithms**; • **Computing methodologies → Discrete space search**.

KEYWORDS

Evolutionary algorithms, self-adaptation, noisy optimisation

ACM Reference Format:
Per Kristian Lehre and Xiaoyu Qin. 2023. Self-adaptation Can Improve the Noise-tolerance of Evolutionary Algorithms. In *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA '23), August 30-September 1, 2023, Potsdam, Germany.* ACM, New York, NY, USA, 26 pages. https://doi.org/10.1145/3594805.3607128

*Authors are listed in alphabetical order.

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

ACM ISBN 979-8-4007-0202-0/23/08.
https://doi.org/10.1145/3594805.3607128

1 INTRODUCTION

Real-world optimisation often involves uncertainty, such as noise. The exact fitness value of a search point may not be determined due to noise. Evolutionary algorithms (EAs), as a type of general optimisers, are widely used to optimise noisy functions [18]. The community of evolutionary computation has considered several noise models for discrete search spaces. For example, one-bit noise [13] flips one bit uniformly with probability q before evaluation; bit-wise noise [25] flips each bit bit-wisely with probability p before evaluation; and symmetric noise [22, 26] returns the changed fitness value $C - f(x)$ instead of $f(x)$ with probability q.

Runtime analysis is a popular theoretical method to mathematically evaluate the performance of EAs, which estimates the expected number of evaluations until the optimum is achieved [11]. In the previous runtime analyses, even a single individual EA is shown robust to low-level noise on pseudo-Boolean functions, but can fail under high-level noise [6, 15, 31]. Gießen and Kötzing [15], Sudholt [31] proved that the runtime of the (1+1) EA on LeadingOnes is exponential if the noise levels satisfies $q = \Omega(1)$ and $p \in \Omega(1/n^2)$ in the one-bit and bit-wise noise models, respectively. A naive method to reduce the effect of noise is averaging multiple noisy evaluations of a solution (resampling strategy). The (1+1) EA with a proper resampling size m with respect to noise level is proved to find the optimum of LeadingOnes in a polynomial runtime under one-bit noise with any level $q \in [0, 1]$ or bit-wise noise with level $p \in O(\log(n)/n)$ [25]. Nevertheless, this strategy is proved to fail in the symmetric noise model regardless of the resampling size used [26]. Using a population is an alternative way to improve robustness in many noise models but requires to set the parameter currently, such as population size and mutation rate [22, 26]. For example, the 2-tournament EA can guarantee $O(n^2)$ expected runtime on LeadingOnes under symmetric noise with noise level $q \in [0, 1/2)$ when the mutation rate is less than $\ln(2 - 2q)/n$; otherwise, its runtime is exponential [22]. Thus, we need to know the exact noise level to set the "right" mutation rate.

It is challenging to find the appropriate parameter setting if the occurrence of noise is unpredictable (or the noise level is unknown). Self-adaptation as a parameter control mechanism can help to configure the mutation rate in the noise-unknown environments, which encodes parameters in the genomes of individuals and evolves them together with the solutions. Self-adaptive EAs have been proved efficient on escaping local optima [5, 23], and solving unknown-structure problems [1]. A recent empirical analysis also demonstrated that the MOSA-EA [28], which self-adapts its mutation rate via multi-objectivisation, can beat a large number of randomised search heuristics on complicated combinatorial optimisation problems and the the noisy LeadingOnes function.

Table 1: Theoretical results of EAs on LeadingOnes under symmetric noise (C, q) ($C \in \mathbb{R}$, constant $0 < \chi_{\text{high}} < \ln(2)$, $\chi_{\text{low}} = a/n$, $\lambda = c\log(n)$ where $a, c > 0$ are constants, $p_c \in o(1) \cap \Omega(1/n)$ in 2-tour' EA with SA-2mr)

Algorithm	Noise-free	Under Noise
(1+1) EA	$O(n^2)$ [13]	$e^{\Omega(n)}$ † (Thm. 2.3)
2-tour' EA with $\frac{\chi_{\text{high}}}{n}$	$O(n^2)$ [2]	$e^{\Omega(n)}$ ‡ (Thm. 2.1)
2-tour' EA with $\frac{\chi_{\text{low}}}{n}$	$\Omega(n^2\log(n))$ (Cor. 3.1)	$O(n^3)$ § (Thm. 2.2)
2-tour' EA with UM-2mr	$O(n^2)$ (Thm. 4.1)	$e^{\Omega(n)}$ ‡ (Thm. 4.2)
2-tour' EA with SA-2mr	$O(n^2)$ (Thm. 5.1)	$O(n^3)$ § (Thm. 5.2)

However, a rigorous analysis of self-adaptation to noise is missing. Runtime analysis of non-elitist population-based EAs can be challenging. Clearly, including self-adaptation and noise makes the analysis even harder.

The main contribution of this paper is the first theoretical analysis of self-adaptive EAs in noisy environments. The rigorous runtime analysis on the LeadingOnes problem shows that the 2-tournament EA with self-adaptation from high/low mutation rates (SA-2mr) can guarantee the lowest expected runtime among the fixed high/low mutation rates and the uniformly chosen mutation rate from high/low rates (uniformly mixing mutation rate, UM-2mr), regardless of the presence of symmetric noise. The results are summarised in Table 1. In addition, we extend to more types of noise, one-bit and bit-wise noise, and a more natural self-adaptation mechanism that adapts the mutation rate from a given interval $(0, 1/2]$ (SA) in the empirical study. The experimental results show that self-adaptive EAs can adapt mutation rates to noise levels and outperform static EAs. The paper is organised as follows: Section 2 introduces algorithms, noise models, related theorems and runtime analysis tools. Sections 3-4 show limitations of using high/low and uniformly mixing mutation rates under symmetric noise. Section 5 analyses the runtime of the 2-tournament EA with self-adapting mutation rates with/without noise. Section 6 shows empirical results. The paper concludes in Section 7.

2 PRELIMINARIES

We first introduce algorithms, noise models, related theorems and runtime analysis tools. We now define notation used later. For any natural numbers $n_2 > n_1$, we define $[n_1, n_2] := \{n_1, \dots, n_2\}$. For any natural number $n \geq 1$, we define $[n] := [1, n]$. The natural logarithm is denoted by $\ln(\cdot)$. 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. We consider two well-known pseudo-Boolean functions OneMax and LeadingOnes, which are defined as $\text{OM}(x) := \sum_{i=1}^{n} x_i$ and $\text{LO}(x) := \sum_{i=1}^{n}\prod_{j=1}^{i} x_j$, respectively.

2.1 Algorithms

We first describe the 2-tournament EA, a non-elitist EA that is robust to noise when using a proper parameter setting [22]. In this paper, we consider three different mutation rate strategies: static, uniformly mixing (UM-2mr) and self-adaptive (SA-2mr and SA).

†For any constant noise levels $q \in (0.127107, 1/2)$.
‡For some constant noise levels $q \in (0, 1/2)$.
§For all constant noise levels $q \in (0, 1/2)$.

Algorithm 1 Static 2-tournament EA

Require: Fitness function $f : \mathcal{X} \to \mathbb{R}$.
Require: Population size $\lambda \in \mathbb{N}$ where $\lambda \geq 2$.
Require: Mutation param. $\chi \in (0, n/2]$. Initial pop. $P_0 \in \mathcal{X}^\lambda$.
for $t = 0, 1, 2, \dots$ until termination condition met **do**
  for $i = 1$ to λ **do**
    $x_1 \leftarrow P_t(i_1)$ where $i_1 \backsim \text{Uniform}([\lambda])$.
    $x_2 \leftarrow P_t(i_2)$ where $i_2 \backsim \text{Uniform}([\lambda])$.
    if $f(x_1) \geq f(x_2)$ **then** $z \leftarrow x_1$ **else** $z \leftarrow x_2$
    $P_{t+1}(i) \leftarrow y$, where y is created by mutating z with mutation rate χ/n.

Algorithm 2 2-tournament EA with self-adaptation

Require: Fitness function $f : \mathcal{X} \to \mathbb{R}$.
Require: Population size $\lambda \in \mathbb{N}$ where $\lambda \geq 2$.
Require: Self-adapting mutation rate strategy $D_{\text{mut}} : \mathbb{R} \to \mathbb{R}$.
Require: Initial population $P_0 \in \mathcal{Y}^\lambda$.
for $t = 0, 1, 2, \dots$ until termination condition met **do**
  for $i = 1$ to λ **do**
    $(x_1, \chi_1/n) \leftarrow P_t(i_1)$ where $i_1 \backsim \text{Uniform}([\lambda])$.
    $(x_2, \chi_2/n) \leftarrow P_t(i_2)$ where $i_2 \backsim \text{Uniform}([\lambda])$.
    if $(x_1, \chi_1/n) \geq (x_2, \chi_2/n)$ **then**
      $(z, \chi/n) \leftarrow (x_1, \chi_1/n)$,
    else
      $(z, \chi/n) \leftarrow (x_2, \chi_2/n)$.
    Sample $\chi'/n \sim D_{\text{mut}}(\chi/n)$.
    $P_{t+1}(i) \leftarrow (y, \chi'/n)$ where y created by mutating z with mutation rate χ'/n.

Algorithm 3 Self-adapting mutation rate strategy (SA) [23]

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

Algorithm 2 shows the static 2-tournament EA. In generation t, we obtain a new individual by selection and mutation. We uniformly select two potential parents, x_1 and x_2, from the population P_t at random (with replacement). Subsequently, we select the individual with the better fitness value to serve as the parent z. Lastly, we employ bit-wise mutation to generate the offspring y. For the 2-tournament with UM-2mr [5], a mutation rate χ/n uniformly sampled from two rates is used in each mutation (see Algorithm 6).

2.2 Self-adaptation

In self-adaptive EAs, each individual not only has its solution but also encodes its own mutation rate, so we define a self-adaptive population $P_t \in \mathcal{Y}^\lambda$, where $\mathcal{Y} = \{0,1\}^n \times (0, 1/2]$. Lehre and Qin [23] summarised a framework of self-adaptive EAs. Similar to static EAs, each individual in the next population P_{t+1} is produced via selection and mutation. The selection based on a *ranking rule* is with respect to fitness function f and individuals' mutation rates.

Algorithm 4 Self-adapting two mutation rates (SA-2mr) [5]

Require: $\chi_{\text{high}} > \chi_{\text{low}} > 0$, $p_c \in (0,1)$.
Require: Mutation rate χ/n.
1: Set $\chi' := \begin{cases} \chi_{\text{high}} \text{ if } \chi = \chi_{\text{low}}, \chi_{\text{low}} \text{ if } \chi = \chi_{\text{high}} & \text{with prob. } p_c \\ \chi & \text{otherwise.} \end{cases}$
2: Return new mutation rate χ'/n.

Then, the selected individual changes its mutation rate based on a *self-adapting mutation rate strategy*, and its solution is mutated by this new mutation rate. This paper applies the lexicographic order from [1, 23], sorting first by increasing fitness value, then by increasing mutation rates:

$$(x,\chi) \succeq (x',\chi') \Leftrightarrow f(x) > f(x') \vee (f(x) = f(x') \wedge \chi \geq \chi'). \tag{1}$$

To self-adapt a mutation rate, we can use the same strategy D_{mut} from [23] (Algorithm 3), where the mutation rate is multiplied by $A > 1$ with probability p_{inc} or divided by A otherwise. We describe the 2-tournament EA with SA as Algorithm 2 using this strategy (Algorithm 3). Although some studies exist on self-adapting mutation rates from the given interval $(0, 1/2]$ [1, 10, 23], noise can make the analysis harder. Therefore, we consider a simplified version in the runtime analysis, the 2-tournament EA with SA-2mr, which only self-adapts two mutation rates $\{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$. In contrast, we illustrate that the fixed high/low mutation rates and the uniformly mixing mutation rate cannot be fast or efficient in the noisy setting. For the sake of analysis, we re-define the self-adaptive population in this algorithm: $P_t \in \mathcal{Y}^\lambda$, where $\mathcal{Y} = \{0,1\}^n \times \{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$. We apply a simple self-adapting mutation rate strategy D_{mut}, in which the mutation rate switches to the other value with a probability $p_c \in (0,1)$ (self-adaptation parameter) (Algorithm 4). Thus, the 2-tournament EA with SA-2mr can be described as Algorithm 2 using Algorithm 4. Note that a similar two-rate self-adaptive EA was studied in [5]. However, the authors employed a ranking rule based solely on fitness values, which can constrain the optimisation speed, as low mutation rate individuals may dominate the population preventing the necessary exploration for faster convergence.

2.3 Noise Models

In the theoretical study, we focus on the symmetric noise model, a well-established noise model extensively investigated in previous research [21, 26]. This choice is motivated by existing runtime analyses of EAs that demonstrate the ineffectiveness of resampling strategies for successful optimisation in the symmetric noise model. In contrast, employing a population has been shown to enhance robustness in this context. Nevertheless, for non-elitist EAs, attaining successful noisy optimisation requires precise tuning of the mutation rate in relation to the noise level. Expanding upon previous research, our theoretical study investigates the potential for further enhancing robustness through the self-adapting mutation rates in the scenario where the presence of symmetric noise is unknown.

To explain this model, we start by defining $f^n(x)$ as the noisy fitness function and $f(x)$ as the noise-free fitness function. The symmetric noise model, as presented by [22], can be formulated as follows: given a probability $q \in [0,1]$ (representing the noise level), an arbitrary constant $C \in \mathbb{R}$, the noisy fitness function f^n is defined for any solution $x \in \{0,1\}^n$ by

$$f^n(x) = \begin{cases} f(x) & \text{with probability } 1-q, \\ C - f(x) & \text{with probability } q. \end{cases}$$

We now present some earlier results related to the symmetric noise model. For static 2-tournament EAs, two theorems related to the LeadingOnes problem in the symmetric noise model have been established [21]. Theorem 2.1 identifies the mutation rate which leads to inefficient optimisation for a given noise level, while Theorem 2.2 reveals the appropriate mutation rate for efficient optimisation. Additionally, for the sake of completeness in our research, we introduce Theorem 2.3, which demonstrates that the (1+1) EA is inefficient under high-level symmetric noise (with proof in Appendix B.1, adapted from Theorem 20 in [15]).

Theorem 2.1 (Theroem 9 in [22]). *For any constant $q \in [0, 1/2)$ and some constant $\delta \in (0,1)$, the probability that the 2-tournament EA with any population size $\lambda \in \text{poly}(n)$ and mutation rate $\chi/n > (\ln(2(1-q)) + \delta)/n$, optimises LeadingOnes in the symmetric noise model (C,q) within e^{cn} generations is $e^{-\Omega(n)}$, for some constant $c > 0$.*

Theorem 2.2 (Theorem 11 in [22]). *For any constant $q \in [0, 1/2)$, and $C \in \mathbb{R}$ and any $\chi \in (0, \ln(2(1-q)))$, the 2-tournament EA with mutation rate χ/n and population size $\lambda > c\log(n)$ for a sufficiently large constant c achieves the optimum on LeadingOnes in the symmetric noise model (C,q) in expected time $O\left(n\lambda \log(n/\chi) + n^2/\chi\right)$.*

Theorem 2.3. *For any $C \in \mathbb{R}$ and any constant $q \in (0.127107, 1/2)$, the probability of the (1+1) EA optimising LeadingOnes in the symmetric noise model (C,q) in $e^{\Omega(n)}$ runtime is $e^{-\Omega(n)}$.*

In the empirical study, we extend the analysis to one-bit noise [15, 25, 31] and bit-wise noise [12, 14, 15, 20, 25, 31], which are defined, respectively, as $f^n(x) = \begin{cases} f(x) & \text{with probability } 1-q, \\ f(x') & \text{with probability } q, \end{cases}$ where x' is a uniformly sampled Hamming neighbour of x, and $f^n(x) = f^n(x')$ where x' is obtained by bit-wisely flipping x with probability p. We refer the reader to [22] for a comprehensive summary of runtime analyses on EAs under noise.

Note that in this study, we apply the reevaluation strategy [4, 14, 15, 21, 22, 25–27] which means the noisy fitness value of an individual is reevaluated every time the individual enters a tournament.

2.4 Runtime Analysis Tools

This subsection introduces the runtime analysis tools for obtaining the lower and upper bounds of the runtime used in this paper. Theorem 2.4 [30] indicates the lower bound of the runtime for any mutation only EAs with respect to the mutation rate. The *level-based theorems* [2, 3, 9] give upper bounds of the runtime for non-elitist population-based algorithms.

Theorem 2.4 (Theorem 11 in [30]). *The expected runtime of every mutation-based EA using mutation rate χ/n on every function with a unique optimum is at least $\left(\frac{\ln(n)-\ln(\ln(n))-3}{\chi(1-\chi/n)^n}\right) n$, if $2^{-n/3} n \leq \chi \leq 1$.*

We use two level-based theorems in this paper: the level-based theorem [2] (Theorem 2.5) and the new level-based theorem [3] (Theorem 2.6). The theorems are applied to algorithms that follow

Algorithm 5 Population-based algorithm

Require: Finite state space $\mathcal{X}$ and population size $\lambda \in \mathbb{N}$; Map D from $\mathcal{X}^\lambda$ to the space of probability distributions over $\mathcal{X}$.
Require: Initial population $P_0 \in \mathcal{X}^\lambda$.
1: **for** $t = 0, 1, 2, \ldots$ until termination condition met **do**
2: **for** $i = 1$ to λ **do**
3: Sample $P_{t+1}(i) \sim D(P_t)$.

the scheme of Algorithm 5. Assume that the search space $\mathcal{X}$ is partitioned into ordered disjoint subsets (called levels) $A_1, \cdots, A_m$. Let $A_{\geq j} := \cup_{k=j}^{m} A_k$ be the search points in level j and higher, and let D be some mapping from the set of all possible populations $\mathcal{X}^\lambda$ into the space of probability distributions of $\mathcal{X}$. Given any subset $A \subseteq \mathcal{X}$, we define $|P_t \cap A| := |\{i \mid P_t(i) \in A\}|$, i.e., the number of individuals in P_t that belong to A. To estimate an upper bound on the runtime using the level-based theorem (Theorem 2.5), three conditions must typically be satisfied: (G1) requires the probability of level "upgrading", i.e., creating an individual in higher levels; (G2) requires the probability of the number of individuals in higher levels "growing"; (G3) requires a sufficient population size. Besides these three conditions, the new level-based theorem (Theorem 2.6) has been proposed to address the issue of "deceptive" regions B that contains individuals with a higher selection probability but at a lower level. The theorem includes an additional condition (G0) that requires a decreasing probability of producing "deceptive" individuals, if there are too many such individuals in the population.

Theorem 2.5 (Level-based theorem [2]). *Given a partition $(A_1, \ldots, A_m)$ of a finite state space $\mathcal{X}$, let $T := \min\{t\lambda \mid |P_t \cap A_m| > 0\}$ be the first point in time that the elements of A_m appear in P_t of Algorithm 5. If there exist $z_1, \ldots, z_{m-1}$, $\delta \in (0, 1]$, and $\gamma_0 \in (0, 1)$ such that for any population $P \in \mathcal{X}^\lambda$,*
(G1) for all $j \in [m-1]$, 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 all $j \in [m-2]$, 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

$$\lambda \geq 4/(\gamma_0\delta^2) \ln\left(128m/(z_*\delta^2)\right), \text{ where } z_* := \min\left\{z_j\right\},$$

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

Theorem 2.6 (New level-based theorem [3]). *Given a partition $(A_1, \ldots, A_m)$ of a finite state space $\mathcal{X}$ and a subset $B \subset \mathcal{X}$, let $T := \min\{t\lambda \mid |P_t \cap A_m| > 0\}$ be the first point in time that the elements of A_m appear in P_t of Algorithm 5. If there exist $z_1, \ldots, z_{m-1}$, $\delta \in (0, 1]$, and $\gamma_0, \psi_0 \in (0, 1)$ such that for any population $P \in \mathcal{X}^\lambda$,*
(G0) for all $\psi \in [\psi_0, 1]$, if $|P \cap B| \leq \psi\lambda$ then $\Pr_{y \sim D(P)}(y \in B) \leq (1-\delta)\psi$,
(G1) for all $j \in [m-1]$, if $|P \cap B| \leq \psi_0\lambda$ and $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ then $\Pr_{y \sim D(P)}\left(y \in A_{\geq j+1}\right) \geq z_j$,
(G2) for all $j \in [m-2]$, 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

$$\lambda \geq 12/(\gamma_0\delta^2) \ln\left(300m/(z_*\delta^2)\right), \text{ where } z_* := \min\left\{z_j\right\},$$

then $E[T] \leq \frac{12\lambda}{\delta} + \frac{96}{\delta^2}\sum_{j=1}^{m-1}\left(\lambda \ln\left(\frac{6\delta\lambda}{4+z_j\delta\lambda}\right) + \frac{1}{z_j}\right)$.

3 HIGH/LOW MUTATION RATES LEAD TO FAILED OPTIMISATION UNDER NOISE OR OR SLOW OPTIMISATION WITHOUT NOISE

The non-elitist EAs should reduce the mutation rate to handle noise [22]. However, too low mutation rates lead to a slow optimisation in the noise-free environment. In this section, we show that the static 2-tournament EAs using the high or low mutation rate cannot be efficient if the presence of noise is unknown. We say the high mutation rate is $\frac{\chi_{\text{high}}}{n}$ where the mutation parameter $\chi_{\text{high}} > 0$ is a constant, and the lower mutation rate is $\frac{\chi_{\text{low}}}{n}$ where $\chi_{\text{low}} = a/n$ for some constant $a > 0$.

It is well-known that the 2-tournament EA with mutation rate χ/n with $\chi < \ln(2)$ is a constant, and population size $\lambda = c\log(n)$ for a sufficiently large constant c, achieves the optimum of LeadingOnes without noise in expected runtime $O(n^2)$ [2]. However, the algorithm can fail in noisy environments if using a constant mutation parameter, i.e., χ_{high}. From Theorem 2.1, we know that for any constant mutation parameter $\chi > 0$, we can find some constant noise level $q \in [0, 1/2)$ such that the 2-tournament EA using any population size $\lambda = \text{poly}(n)$ optimises LeadingOnes under symmetric noise within e^{cn} generations with probability $e^{-\Omega(n)}$ where $c > 0$ is a constant.

We can use a sufficiently low mutation rate against noise, e.g., $\frac{\chi_{\text{low}}}{n}$. From Theorem 2.2, we know that the expected runtime of the 2-tournament EA with mutation rate χ_{low}/n and population size $\lambda = c\log(n)$ for a sufficiently large constant c on optimising LeadingOnes under symmetric noise for any constant noise level $q \in [0, 1/2)$ is $O\left(n^3\right)$. However, such a low mutation rate slows down the noise-free optimisation by a small but super-constant factor, i.e., $\Omega(n^2\log(n))$ runtime instead of $O(n^2)$ guaranteed by using $\frac{\chi_{\text{high}}}{n}$, which is indicated by Corollary 3.1 via Theorem 2.4.

Corollary 3.1. *The expected runtime of the 2-tournament EA using mutation parameter satisfying $\chi \geq 2^{-n/3}n$ and $\chi \in O(1/n)$ on LeadingOnes is $\Omega(n^2\log(n))$.*

4 UNIFORMLY MIXING MUTATION RATES DO NOT HELP UNDER NOISE

In this section, we show runtime analysis results on the 2-tournament EA with uniformly mixing high/low mutation rates (UM-2mr) under noise. Theorems 4.1-4.2 present that using UM-2mr can optimise the noise-free LeadingOnes function in expected runtime $O(n^2)$, but can fail under symmetric noise with a high probability. The proofs are in Appendices B.2-B.3.

Theorem 4.1. *For any constants $\chi_{\text{high}}, a > 0$ and $\chi_{\text{low}} = a/n$, the expected runtime of the 2-tournament EA with UM-2mr from $\{\chi_{\text{high}}/n, \chi_{\text{low}}/n\}$ and population size $\lambda > c\log(n)$ for a sufficiently large constant c on optimising LeadingOnes is $O\left(n\lambda\log(n) + n^2\right)$.*

Theorem 4.2. *For any constant $q \in (0, 1/2)$ and any constant $\delta \in (0, q)$, the probability that the 2-tournament EA with UM-2mr from*

$\{\chi_{\text{high}}/n, \chi_{\text{low}}/n\}$ where constants $\chi_{\text{high}} \geq \ln\left(\frac{1-q}{q-\delta}\right) > \chi_{\text{low}} > 0$ with any population size $\lambda \in \text{poly}(n)$ optimises LeadingOnes in the symmetric noise model (C, q), where $C \in \mathbb{R}$, within time e^{cn} is $e^{-\Omega(n)}$, for some constant $c > 0$.

5 SELF-ADAPTING MUTATION RATES GUARANTEE EFFICIENCY UNDER NOISE

We now analyse the self-adaptive EA using level-based theorems to show their efficiency in noisy and noise-free environments. Theorem 5.1 shows that the 2-tournament EA using SA-2mr achieves a comparable performance to using a high mutation rate $\frac{\chi_{\text{high}}}{n}$, i.e., $O(n^2)$ runtime, on the noise-free LeadingOnes function. The proof of Theorem 5.1 is conducted by the level-based theorem (Theorem 2.5). (shown in Appendix B.4). Theorem 5.2 shows that the self-adaptive EA also efficiently optimises under symmetric noise.

Theorem 5.1. *For any constant $\chi_{\text{high}} \in (0, \ln(2(1-\delta))]$ where $\delta \in (0, 1/2)$ is any constant, $\chi_{\text{low}} = a/n$ where $a > 0$ is any constant, and any $p_c \in o(1) \cap \Omega(1/n)$, the expected runtime of the 2-tournament EA using SA-2mr from $\{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ with self-adaptation parameter p_c and population size $\lambda > c\log(n)$ for a sufficiently large constant c on optimising LeadingOnes without noise is $O\left(n\lambda \log(n) + n^2\right)$.*

Theorem 5.2. *For any constant $\chi_{\text{high}} > 0$, $\chi_{\text{low}} = a/n$ where $a > 0$ is any constant, an arbitrary constant $q \in [0, 1/2)$ and $p_c \in o(1) \cap \Omega(1/n)$, the expected runtime of the 2-tournament EA using SA-2mr from $\{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ with self-adaptation parameter p_c and population size $\lambda > c\log(n)$ for a sufficiently large constant c on optimising LeadingOnes in the symmetric noise model (C, q), where $C \in \mathbb{R}$, is $O\left(n\lambda \log(n) + n^3\right)$.*

Theorem 5.2 is the most important result of this paper. To prove it, we consider the two cases based on the noise level q. If the noise level is small enough compared to the high mutation rate $\frac{\chi_{\text{high}}}{n}$, we use a similar approach of Theorem 5.1 to complete the proof. Otherwise, we use a different level partition and the new level-based theorem (Theorem 2.6) to prove it. Precisely, we define a value $\ell \in \mathbb{N}$ such that for any constant $\delta \in \left(0, (1/2-q)^3\right]$,

$$\left(1 - \frac{\chi_{\text{high}}}{n}\right)^{\ell-1} > \frac{1+\delta}{2(1-q)} \geq \left(1 - \frac{\chi_{\text{high}}}{n}\right)^{\ell}, \tag{2}$$

and distinguish between two cases: (A) $\ell \leq n-2$ and (B) $\ell \geq n-1$.

For case (A) $\ell \leq n-2$, we use the level partition defined in Definition 5.1. Figure 1 illustrates this level definition. The state space $\mathcal{Y}$ is divided into $n+1$ levels $A_{j\in[0..n]}$, with respect to $\text{LO}(x)$. Each of the first ℓ levels (the red/I region) is divided into two sub-levels, $A_{(j,1)}$ and $A_{(j,2)}$, representing the low and high mutation rates, respectively. Levels $A_{j\in[\ell+1..n-1]}$ (the green/III region) are defined as having only one sub-level $A_{(j,1)}$, which represents the low mutation rate. The final level (the optimal level, the white region), $A_n := A_{(n,1)}$, contains both high and low mutation rates. The sub-level $A_{(\ell,2)}$ (the cyan/II region) is extended to the rest of the state space , where $f(x) \geq \ell$ and with high mutation rate.

Definition 5.1. *For any $\ell \in [0, n-2]$, we define*

$$A_{(j,1)} := \begin{cases} \{(x, \frac{\chi_{\text{low}}}{n}) \mid LO(x) = j\} & \text{if } 0 \leq j \leq n-1, \\ \{(x, \frac{\chi_{\text{high}}}{n}), (x, \frac{\chi_{\text{low}}}{n}) \mid LO(x) = n\} & \text{if } j = n; \text{ and} \end{cases}$$

$$A_{(j,2)} := \begin{cases} \{(x, \frac{\chi_{\text{high}}}{n}) \mid LO(x) = j\} & \text{if } 0 \leq j \leq \ell-1 \\ \{(x, \frac{\chi_{\text{high}}}{n}) \mid LO(x) \geq \ell\} & \text{if } j = \ell. \end{cases}$$

To compare the levels $A_{(j,i)}$ and $A_{(j',i')}$, we define $(j,i) > (j',i')$ if either ($j = j'$ and $i > i'$) or ($j > j'$). To simplify the notation, we also define $(j,1)+1 = (j,2)$ and $(j,2)+1 = (j+1,1)$ for $j \leq \ell$, and $(j,1)+1 = (j+1,1)$ for $j \geq \ell+1$.

To apply Theorem 2.6, we must estimate the "upgrading" probability that is sampling an offspring in the higher level (condition (G1)), and the "growing" probability that sampling an offspring in at least the same level (condition (G2)). The individuals in the green/III region might not have a sufficiently large probability of being selected if there are too many individuals with high fitness and high mutation rate (the cyan/II region). The high mutation rate can fail the optimisation under noise. Therefore, it is crucial to verify condition (G0), which ensures that there are not too many individuals in the cyan/II region. Finally, we gain an upper runtime bound by calculating the required population size (condition (G3)).

We first introduce some lemmas for the proof. The presence of noise essentially affects the selection [21], so we compute the probability of selecting a high-level individual in noisy environments in Lemma 5.3. Lemmas 5.4-5.5 are used to verify conditions (G0) and (G2) of Theorem 2.6, respectively.

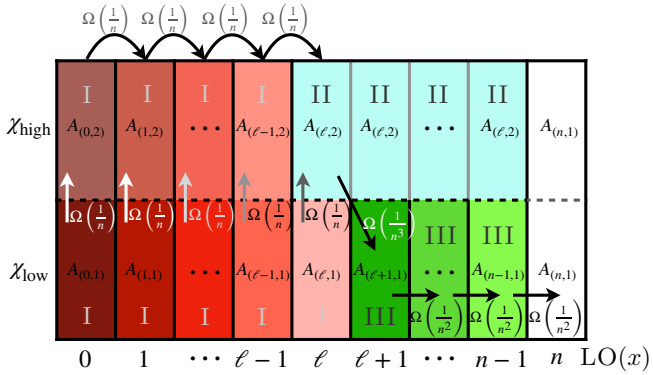

Figure 1: Illustration of the level partition defined in Definition 5.1. The notions on arrows indicate the "upgrading" probabilities for levels in the proof of Theorem 5.2.

Lemma 5.3. *Assume that we have a population $P_t \in \mathcal{Y}^\lambda$ where $\mathcal{Y} := \{0,1\}^n \times \{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ and $\chi_{\text{high}} > \chi_{\text{low}} > 0$, that is sorted such that $P_t(1) \geq \cdots \geq P_t(\lambda)$ on the noise-free version of LeadingOnes function. For any $\gamma \in (0,1)$, any $C \in \mathbb{R}$ and any $q \in [0, 1/2)$, if $(x_1, \chi_1/n)$ and $(x_2, \chi_2/n)$ are two individuals which are uniformly at random selected from P_t, and $(z, \chi'/n)$ is created by steps 5-8 in Algorithm 2 on the LeadingOnes function in the symmetric noise model (C, q), then the probability of $(z, \chi'/n) \geq P_t(\lfloor\gamma\lambda\rfloor)$ where $\lambda = |P_t|$ is $\gamma(2(1-q) - (1-2q)\gamma) \geq 2\gamma(1-q)(1-\gamma)$.*

Proof. There are two events in which we select an "advanced individual", i.e., $(z, \chi'/n) \geq P_t(\lfloor\gamma\lambda\rfloor)$:

(a) the algorithm selects two individuals $(x_1, \chi_1/n)$ and $(x_2, \chi_2/n)$ both from the top $\lfloor\gamma\lambda\rfloor$ individuals of sorted P_t. This happens with probability γ^2.

(b) either $(x_1, \chi_1/n)$ or $(x_2, \chi_2/n)$ from the top $\lfloor \gamma\lambda \rfloor$ individuals and selects such individual even if the noise occurs.

For event (2), we assume without loss of generality that $(x_1, \chi_1/n) \succeq (x_2, \chi_2/n)$. Then the probability of a successful comparison S, i.e, $(x_1, \chi_1/n)$ is exactly selected, is $\Pr(S) = \Pr(f^n(x_1) > f^n(x_2)) + \frac{1}{2}\Pr(f^n(x_1) = f^n(x_2)) = (1-q)$, where the last equation is from Lemma 1 (f) in [22]. For completeness, we now repeat the arguments from that lemma here. We assume that $f(x_1) = a$ and $f(x_2) = b$ where $a > b$. Then we say that x_1 "wins" if event S happens, and we distinguish between three cases:

(1) If $a + b > C$, then x_1 wins if and only if there is noise in x_2, i.e, $\Pr(E) = (1-q)^2 + (1-q)q$.
(2) If $a+b = C$, then x_1 wins if and only if there is no noise in both x_1 and x_2, or there is noise in either x_1 and x_2 (same fitness values, so with half chance), i.e., $\Pr(E) = (1-q)^2+(1-q)q/2+q(1-q)/2$.
(3) If $a + b < C$, then x_1 wins if and only if there is no noise in x_2, i.e., $\Pr(E) = (1-q)^2 + q(1-q)$.

Therefore, we obtain $\Pr(S) = (1-q)^2 + (1-q)q = \frac{1}{2} + \frac{1}{2} - q$. Thus, the probability of selecting an individual in the top $\lfloor \gamma\lambda \rfloor$ individuals of sorted P_t is

$$\begin{aligned}\gamma^2 + 2\gamma(1-\gamma)\Pr(S) &= \gamma\left(2\left(1-q\right) - \left(1-2q\right)\gamma\right)\\ &\geq 2\gamma(1-q)(1-\gamma).\end{aligned}$$ □

The following lemma ensures that there are not too many individuals in the cyan/II region.

Lemma 5.4 (Condition (G0)). *Given any subset $B \subset \mathcal{Y}$ where $\mathcal{Y} := \{0,1\}^n \times \{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ and $\chi_{\text{high}} > \chi_{\text{low}} > 0$, let $Y_t := |P_t \cap B|$ be the number of individuals in population P_t of the 2-tournament EA with SA-2mr from $\{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ and $p_c \in o(1) \cap \Omega(1/n)$ that belong to subset B. Consider the symmetric noise model (C, q), where $C \in \mathbb{R}$ and constant $q \in [0, 1/2)$. If there exist three parameters $\rho, \varepsilon, \sigma \in (0,1)$ such that $\Pr((y,\chi'/n) \in B \mid (z,\chi/n) \in B) \leq \rho$, and $\Pr((y,\chi'/n) \in B \mid (z,\chi/n) \notin B) \leq \sigma\psi - \varepsilon$ for $\psi \in [\psi_0, 1]$, where $\psi_0 = \frac{2(1-q)-(1-\sigma)/\rho}{1-2q}$ and $\psi_0 \in (0,1)$, then*

$$\Pr\left((y,\chi'/n) \in B \mid |P_t \cap B| \leq \psi\lambda\right) \leq \psi(1-\varepsilon).$$

This lemma is very similar to Lemma 2 in [5].

Proof. Let $\psi := Y_t/\lambda$. For the upper bound, we assume that all search points in B have higher fitness and higher mutation rate than search points in $\mathcal{X}\backslash B$. Then,

$$\begin{aligned}&\Pr\left((z,\chi/n) \in B \wedge (y,\chi'/n) \in B\right)\\ &= \Pr\left((z,\chi/n) \in B\right)\Pr\left((y,\chi'/n) \in B \mid (z,\chi/n) \in B\right)\end{aligned}$$

by Lemma 5.3,

$$\leq \psi\left(2\left(1-q\right) - \left(1-2q\right)\psi\right)\rho.$$

Let $g(\psi) = \psi\left(2\left(1-q\right) - \left(1-2q\right)\psi\right)$ which is monotone increasing when $\psi \in (0,1)$ by Lemma A.3 (1), such that

$$\leq g\left(\max(\psi_0, \psi)\right)\rho$$

then by $\psi \geq \psi_0$ and the value of ψ_0,

$$\leq \psi\left(2\left(1-q\right) - \left(1-2q\right)\psi_0\right)\rho = \psi(1-\sigma).$$

Thus, the probability of producing an individuals in B is

$$\begin{aligned}&\Pr\left((y,\chi'/n) \in B \mid |P_t \cap B| \leq \psi\lambda\right)\\ &= \Pr\left((z,\chi/n) \in B \wedge (y,\chi'/n) \in B \mid |P_t \cap B| \leq \psi\lambda\right)\\ &\quad + \Pr\left((z,\chi/n) \notin B \wedge (y,\chi'/n) \in B \mid |P_t \cap B| \leq \psi\lambda\right)\\ &\leq \psi(1-\sigma) + (\sigma\psi - \varepsilon)\\ &\leq \psi - \varepsilon < \psi(1-\varepsilon) \leq \psi(1-\varepsilon).\end{aligned}$$ □

The following lemma gives the "expanding" probability that sampling an offspring in at least the same level.

Lemma 5.5 (Condition (G2)). *Assume that $0 < \delta \leq (\frac{1}{2} - q)^3$ and $q \in [0, 1/2)$ are any constants, $\psi_0 := \frac{1-q}{\frac{1}{2}-q}\left(1 - \frac{1-(1/2)(1/2-q)^2}{1+\delta}\right)$ and the state space $\mathcal{Y} := \{0,1\}^n \times \{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ is divided according to Definition 5.1. Consider LeadingOnes in the symmetric noise model (C, q) where $C \in \mathbb{R}$. There exist constants $\Delta \in (0, (-q/5+1/10)/2]$ and $\gamma_0 \in (0, -q/90+1/180]$, for any population P_t of the 2-tournament EA with SA-2mr from $\{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ and $p_c \in o(1) \cap \Omega(1/n)$, any $\gamma \in (0, \gamma_0]$ and $(j,i) \geq (\ell, 2)$, if $|P_t \cap A_{(\ell,2)}| \leq \psi_0\lambda$, $|P_t \cap A_{\geq(j,i)}| \geq \gamma_0\lambda$ and $|P_t \cap A_{\geq(j,i)+1}| \geq \gamma\lambda$, then $\Pr\left((y,\chi'/n) \in A_{\geq(j,i)+1}\right) \geq \gamma(1+\Delta)$.*

Proof. We assume that $|P_t \cap A_{(\ell,2)}| \leq \psi_0\lambda$. By the definition of ψ_0, and $0 < \delta \leq (1/2-q)^3$, we get upper and lower bounds of ψ_0 for later use. By replacing for the upper on δ in the definition of ψ_0,

$$\begin{aligned}\psi_0 &\leq \frac{2(1-q) - \frac{2\left(1-\frac{1}{2}\left(\frac{1}{2}-q\right)^2\right)(1-q)}{1+(1/2-q)^3}}{1-2q}\\ &= \frac{4 - 16q + 20q^2 - 8q^3}{9 - 6q + 12q^2 - 8q^3}.\end{aligned} \tag{3}$$

By replacing for the lower on δ in the definition of ψ_0,

$$\begin{aligned}\psi_0 &> \frac{2(1-q) - 2\left(1 - \frac{1}{2}\left(\frac{1}{2}-q\right)^2\right)(1-q)}{1-2q}\\ &= \left(\frac{1}{2}-q\right)\left(\frac{1}{2}-\frac{q}{2}\right).\end{aligned} \tag{4}$$

We now derive a lower bound p_0 on the probability that given an individual $(z,\chi/n)$ in level $A_{\geq(j,i)+1}$, the mutation operator produces an individual $(y,\chi'/n)$ in $A_{\geq(j,i)+1}$, for $(j,i) \geq (\ell,2)$. The levels higher than $A_{(\ell,2)}$ are all with mutation rate $\frac{\chi_{\text{low}}}{n}$ except the optimal level $A_{(n,1)}$, thus

$$\begin{aligned}&\Pr\left((y,\chi'/n) \in A_{\geq(j,i)+1} \mid (z,\chi/n) \in A_{\geq(j,i)+1}\right)\\ &\geq \left(1 - \frac{\chi_{\text{low}}}{n}\right)^n (1-p_c)\end{aligned}$$

by Lemma A.2,

$$\geq e^{-\chi_{\text{low}}}\left(1 - \frac{{\chi_{\text{low}}}^2}{n}\right)(1-p_c) =: p_0 = (1-o(1)),$$

since $p_c \in o(1)$ and $\chi_{\text{low}} \in o(1)$.

Based on the level partition, the individuals in levels $A_{\geq(j,i)+1}$ are fitter than any other individual not in $A_{(\ell,2)}$, but could be less fit than individuals in $A_{(\ell,2)}$. Therefore, an individual in $A_{\geq(j,i)+1}$ can be produced by the all events happened: Event (a) no individual

is selected from $A_{(\ell,2)}$, Event (b) at least one individual is selected from $A_{\geq(j,i)+1}$ and it wins the tournament (step 5 in Algorithm 1), and Event (c) with probability at least p_0, the mutated individual z is in $A_{\geq j+1}$. The probability of Event (a) occurring is given by $\Pr(S) = 1-q$, as demonstrated in Lemma 5.3. Thus, the joint probability of these events is at least

$$2\gamma(1-\psi_0-\gamma)(1-q)p_0 = 2\gamma(1-q)(1-\psi_0-\gamma)(1-o(1))$$

by Eq. (3),

$$\geq 2\gamma(1-q)\left(1-\frac{4-16q+20q^2-8q^3}{9-6q+12q^2-8q^3}-\gamma\right)(1-o(1))$$
$$= 2\gamma(1-q)\left(\frac{5+10q-8q^2}{9-6q+12q^2-8q^3}-\gamma\right)(1-o(1))$$

by $\gamma_0 \in (-q/90+1/180]$, then for all $\gamma \in (0,\gamma_0)$,

$$\geq 2\gamma(1-q)\left(\frac{5+10q-8q^2}{9-6q+12q^2-8q^3}-\gamma_0\right)(1-o(1))$$
$$= 2\gamma(1-q)\left(\frac{5+10q-8q^2}{9-6q+12q^2-8q^3}-\frac{1}{180}+\frac{q}{90}\right)(1-o(1))$$
$$= \gamma\left(1+\frac{81+1473q-4368q^2+2216q^3-48q^4+16q^5}{90(9-6q+12q^2-8q^3)}\right)(1-o(1))$$

by Lemma A.3 (2), we know $90(9-6q+12q^2-8q^3) < 810$, then

$$> \gamma\left(1+\frac{81+1473q-4368q^2+2216q^3-48q^4+16q^5}{810}\right)(1-o(1))$$
$$= \gamma\left(1+\left(\frac{1}{10}+\frac{491q}{270}-\frac{728q^2}{135}+\frac{1108q^3}{405}-\frac{8q^4}{135}+\frac{8q^5}{405}\right)\right)(1-o(1))$$

by $\Delta \in (0,(-q/5+1/10)/2]$ which is a constant and Lemma A.3 (3), we know $\frac{1}{10}+\frac{491q}{270}-\frac{728q^2}{135}+\frac{1108q^3}{405}-\frac{8q^4}{135}+\frac{8q^5}{405} \geq 2\Delta$, then

$$\geq \gamma(1+2\Delta)(1-o(1)) \geq \gamma(1+\Delta).$$ □

We now prove Theorem 5.2 using Lemmas 5.3 to 5.5.

Proof of Theorem 5.2. Recall Eq. (2), we first consider case (A) $\ell \leq n-2$ which refers to the level partition defined by Definition 5.1. We apply the new level-based theorem (Theorem 2.6) with respect to this level partitioning. Prior to proving the theorem, we introduce several constants that will be used in the subsequent calculations: δ is any constant that satisfies $0 < \delta \leq (1/2-q)^3$, $\rho := \frac{1+\delta}{2(1-q)}$, $\sigma := (1/2-q)^2/2$, $\psi_0 := \frac{2(1-q)-(1-\sigma)/\rho}{1-2q}$, $\zeta := \frac{1}{40}\left(\frac{1}{2}-q\right)^3$, $\varepsilon := (1/2-q)^3/10 > 0$, $\gamma_0 := \min\left\{\frac{\delta}{4(1+\delta)}, -q/90+1/180\right\}$ and $\Delta := \min\{(-q/5+1/10)/2, \delta/2\}$.

We now show that the condition (G0) of Theorem 2.6 is satisfied. We consider $A_{(\ell,2)}$ as the B subset in Theorem 2.6 for all $(j,i) \geq (\ell,2)$ (the cyan/II region illustrated in Figure 1). Assume $(z,\chi/n) \in A_{(\ell,2)}$, then, to produce $(y,\chi'/n) \in A_{(\ell,2)}$, it is necessary not to change the mutation rate and not flip the first ℓ bits. Using Eq. (2) and $p_c \in (0,1)$, the probability of this event is

$$\Pr\left((y,\chi'/n) \in A_{(\ell,2)} \mid (z,\chi/n) \in A_{(\ell,2)}\right) = (1-p_c)\left(1-\frac{\chi_{\text{high}}}{n}\right)^\ell$$
$$< \frac{1+\delta}{2(1-q)} = \rho.$$

When applying Lemma 5.4, we use the parameter ψ_0 which has been defined in terms of ρ and σ, as

$$\psi_0 = \frac{2(1-q)-(1-\sigma)/\rho}{1-2q} = \frac{1-q}{1/2-q}\left(1-\frac{1-(1/2)(1/2-q)^2}{1+\delta}\right)$$

by Eq. (4),

$$> \left(\frac{1}{2}-q\right)\left(\frac{1}{2}-\frac{q}{2}\right).$$

If $(z,\chi/n)$ is not in $A_{(\ell,2)}$, it is necessary to flip at least one specific bit-position, or change the mutation rate, an event which occurs with probability

$$\Pr\left((y,\chi'/n) \in A_{(\ell,2)} \mid (z,\chi/n) \notin A_{(\ell,2)}\right) < \max\left\{p_c, \frac{\chi_{\text{high}}}{n}\right\}$$

which is by constants $p_c, \frac{\chi_{\text{high}}}{n} \in o(1)$, $\psi_0, \sigma > 0$ and $\varepsilon = (1/2-q)^3/10 > 0$,

$$< \psi_0\sigma - \varepsilon \leq \psi\sigma - \varepsilon.$$

$$\psi_0\sigma - \varepsilon > \left(\frac{1}{2}-q\right)\left(\frac{1}{2}-\frac{q}{2}\right)\frac{(1/2-q)^2}{2} - \frac{(1/2-q)^3}{10} = \frac{1}{20}\left(\frac{1}{2}-q\right)^3(3-5q)$$

since $q < 1/2$, we have $3-5q > 1/2$, then

$$> \frac{1}{40}\left(\frac{1}{2}-q\right)^3 =: \zeta > 0,$$

where ζ is a constant which is larger than $\max\left\{p_c, \frac{\chi_{\text{high}}}{n}\right\} = o(1)$. Lemma 5.4 now implies that condition (G0) satisfied with $\psi_0 = \frac{1-q}{1/2-q}\left(1-\frac{1-(1/2)(1/2-q)^2}{1+\delta}\right)$.

We then verify condition (G2) of Theorem 2.6. We first consider the case of $(j,i) \leq (\ell,1)$ (the red/I region illustrated in Figure 1). Recall the definition of γ_0 and define $A_+ := A_{\geq(j,i)+1}$. Assume that $|P_t \cap A_+| = \gamma\lambda$ for $\gamma \in (0,\gamma_0]$. To produce an individual $(y,\chi'/n) \in A_+$, it suffices to select an individual $(z,\chi/n) \in A_+$, do not change the mutation rate and do not flip first $j+1$ bits, then the lower bound of this probability is,

$$\Pr\left((y,\frac{\chi'}{n}) \in A_+\right) = \Pr\left((z,\frac{\chi}{n}) \in A_+\right)\Pr\left((y,\frac{\chi'}{n}) \in A_+ \mid (z,\frac{\chi}{n}) \in A_+\right)$$
$$= 2\gamma(1-q)(1-\gamma)(1-p_c)\left(1-\frac{\chi'}{n}\right)^j$$

since $j \leq \ell$ in this case,

$$\geq 2\gamma(1-q)(1-\gamma)(1-p_c)\left(1-\chi'/n\right)^\ell$$
$$\geq 2\gamma(1-q)(1-\gamma)(1-p_c)\left(1-\chi_{\text{high}}/n\right)^\ell$$

by the definition of ℓ in Eq. (2),

$$\begin{aligned}
&\geq 2\gamma(1-q)(1-\gamma)(1-p_c)\frac{1+\delta}{2(1-q)}\left(1-\frac{\chi_{\text{high}}}{n}\right)\\
&= \gamma(1-\gamma)(1-p_c)(1+\delta)(1-o(1))\\
&\geq \gamma(1-\gamma_0)(1-p_c)(1+\delta)(1-o(1))\\
&\geq \gamma\left(1+3\delta/4\right)(1-p_c)(1-o(1))
\end{aligned}$$

then since $\Delta = \delta/2$ and $p_c \in o(1)$,

$$\geq \gamma(1+\Delta).$$

We also know condition (G2) is satisfied if $(j,i) \geq (\ell,2)$ from Lemma 5.5 for constants γ_0 and Δ.

We now verify condition (G1) of Theorem 2.6. Assume that the size of $P_t \cap A_{\geq(j,i)}$ is at least $\gamma_0\lambda$, i.e., $\gamma_0\lambda \leq |P_t \cap A_{\geq(j,i)}|$. The lower bound of selecting an individual $(z,\chi/n) \in A_{\geq(j,i)}$ is $\Pr\left((z,\chi/n) \in A_{\geq(j,i)}\right) \geq \gamma_0^2 = \Omega(1)$. We distinguish levels into four groups:

- For levels $A_{(j,1)\leq(\ell,1)}$ (the red/I region with χ_{low} in Figure 1), to produce an individual $(y,\chi'/n) \in A_{\geq(j,1)+1}$ it suffices to select an individual $(z,\chi/n) \in A_{\geq(j,1)}$ and change its mutation rate from $\frac{\chi_{\text{low}}}{n}$ to $\frac{\chi_{\text{high}}}{n}$, then this probability is at least

$$\begin{aligned}
\Pr\left((y,\chi'/n) \in A_{\geq(j,1)+1}\right) &\geq \Pr\left((z,\chi/n) \in A_{\geq(j,1)}\right)p_c(1-q)\\
&=: z_{(j,1)} \in \Omega\left(\frac{1}{n}\right).
\end{aligned}$$

- For levels $A_{(j,2)\leq(\ell-1,2)}$ (the red/I region with $\frac{\chi_{\text{high}}}{n}$ in Figure 1), to produce an individual $(y,\chi'/n) \in A_{\geq(j,2)+1}$ it suffices to select an individual $(z,\chi/n) \in A_{\geq(j,2)}$, do not change the mutation rate and only flip the $(j+1)$-th bit, then the lower bound of this probability is

$$\begin{aligned}
&\Pr\left((y,\chi'/n) \in A_{\geq(j,2)+1}\right)\\
&\quad\geq \Pr\left((z,\chi/n) \in A_{\geq(j,2)}\right)(1-p_c)(1-q)\left(1-\frac{\chi_{\text{high}}}{n}\right)^{n-1}\frac{\chi_{\text{high}}}{n}\\
&\quad=: z_{(j,2)} \in \Omega\left(\frac{1}{n}\right).
\end{aligned}$$

- For level $A_{(\ell,2)}$ (the cyan/II region in Figure 1), to produce an individual $(y,\chi'/n) \in A_{\geq(\ell+1,1)}$ it suffices to select an individual $(z,\chi/n) \in A_{\geq(\ell,2)}$, change its mutation rate from $\frac{\chi_{\text{high}}}{n}$ to $\frac{\chi_{\text{low}}}{n}$ and only flip the $\ell+1$-th bit, then the lower bound of this probability is

$$\begin{aligned}
&\Pr\left((y,\chi'/n) \in A_{\geq(\ell+1,1)}\right)\\
&\quad\geq \Pr\left((z,\chi/n) \in A_{\geq(\ell,2)}\right)(1-q)p_c\left(1-\frac{\chi_{\text{low}}}{n}\right)^{n-1}\frac{\chi_{\text{low}}}{n}\\
&\quad=: z_{(\ell,2)} \in \Omega\left(\frac{1}{n^3}\right).
\end{aligned}$$

- For levels $A_{(j,1)\geq(\ell+1,1)}$ (the green/III region in Figure 1), to produce an individual $(y,\chi'/n) \in A_{\geq(j+1,1)}$ it suffices to select an individual $(z,\chi/n) \in A_{\geq(j,1)}$, do not change the mutation rate and only flip the $(j+1)$-th bit, then the lower bound of this probability is

$$\begin{aligned}
&\Pr\left((y,\chi'/n) \in A_{\geq(j+1,1)}\right)\\
&\quad\geq \Pr\left((z,\chi/n) \in A_{\geq(j,1)}\right)(1-q)(1-p_c)\left(1-\frac{\chi_{\text{low}}}{n}\right)^{n-1}\frac{\chi_{\text{low}}}{n}\\
&\quad=: z_{(j,1)} \in \Omega\left(\frac{1}{n^2}\right).
\end{aligned}$$

Then we compute the population size required by condition (G3). Since $\gamma_0, \Delta > 0$ are some constants and $m = 2\ell + (n-\ell) \leq 2n$, then $\lambda > \frac{12}{\gamma_0\Delta^2}\ln\left(\frac{300m}{\min\{z_{(j,i)}\}\Delta^2}\right) = O(\log(n))$. Condition (G3) is satisfied by $\lambda \geq c\log(n)$ for a sufficiently large constant c.

Finally, all conditions of Theorem 2.6 hold, and the expected runtime is no more than

$$\begin{aligned}
E[T] \leq\ & \frac{12\lambda}{\Delta} + \frac{96}{\Delta^2}\sum_{j=0}^{\ell}\left(\lambda\ln\left(\frac{6\Delta\lambda}{4+z_{(j,1)}\Delta\lambda}\right)+\frac{1}{z_{(j,1)}}\right)\\
&+\frac{96}{\Delta^2}\sum_{j=0}^{\ell-1}\left(\lambda\ln\left(\frac{6\Delta\lambda}{4+z_{(j,2)}\Delta\lambda}\right)+\frac{1}{z_{(j,2)}}\right)\\
&+\frac{96}{\Delta^2}\left(\lambda\ln\left(\frac{6\Delta\lambda}{4+z_{(\ell,2)}\Delta\lambda}\right)+\frac{1}{z_{(\ell,2)}}\right)\\
&+\frac{96}{\Delta^2}\sum_{j=\ell+1}^{n-1}\left(\lambda\ln\left(\frac{6\Delta\lambda}{4+z_{(j,1)}\Delta\lambda}\right)+\frac{1}{z_{(j,1)}}\right)\\
=\ & O\Big(\lambda+(\ell+1)\left(\lambda\log(n)+n\right)+\ell\left(\lambda\log(n)+n\right)\\
&+\left(\lambda\log(n)+n^3\right)+(n-\ell-2)\left(\lambda\log(n)+n^2\right)\Big)\\
=\ & O(n\lambda\log(n)+n^3).
\end{aligned}$$

For case (B) $\ell \geq n-1$, we know that $\frac{1+\delta}{2(1-q)} \leq \left(1-\frac{\chi_{\text{high}}}{n}\right)^{n-1} = \left(1-\frac{\chi_{\text{high}}}{n}\right)^{n}/(1-o(1))$. We use the level-based theorem (Theorem 2.5) on the level partition applied in the proof of Theorem 5.1. Condition (G2) can be verified by definitions of Δ and γ_0:

$$\begin{aligned}
&\Pr\left((y,\chi'/n) \in A_{\geq(j,i)+1}\right)\\
&= \Pr\left((z,\chi/n) \in A_{\geq(j,i)+1}\right)\\
&\qquad\cdot\Pr\left((y,\chi'/n) \in A_{\geq(j,i)+1} \mid (z,\chi/n) \in A_{\geq(j,i)+1}\right)\\
&\geq (\gamma^2+2\gamma(1-\gamma))(1-q)(1-p_c)\left(1-\frac{\chi'}{n}\right)^{n}\\
&\geq 2\gamma(1-\gamma)(1-q)(1-p_c)\left(1-\frac{\chi_{\text{high}}}{n}\right)^{n} \geq \gamma(1+\Delta).
\end{aligned}$$

Condition (G1) is similar with the proof of Theorem 5.1. Then, we know that the runtime is $O(n\lambda\log(n)+n^2)$ if using population size $\lambda > c\log(n)$ for a sufficiently large constant c.

Therefore, the overall runtime is $O(n\lambda\log(n)+n^3)$. □

6 EXPERIMENTS

As a complement to our theoretical analysis, we expand our investigation to include both the symmetric, one-bit and bit-wise

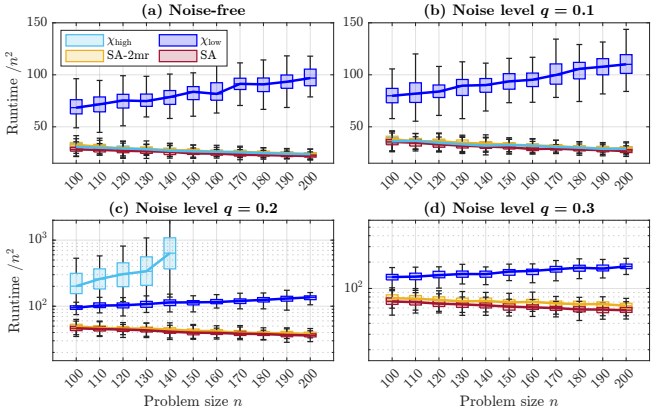

Figure 2: Runtimes of 2-tournament EAs on LeadingOnes under symmetric noise with different noise levels ($C = 0$).

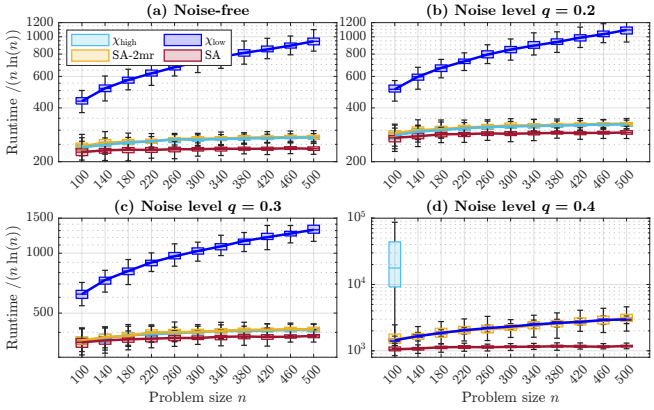

Figure 3: Runtimes of 2-tournament EAs on OneMax under symmetric noise with different noise levels ($C = 0$).

noise models, as well as self-adaptation of mutation rates within a given interval. In this section, we empirically analyse the performance of 2-tournament EAs using fixed and self-adaptive mutation rates on LeadingOnes and OneMax under different levels of noise. Additionally, we investigate the behaviour of mutation rates in self-adaptation under noise.

We use the parameter settings satisfying the runtimes analyses in Sections 3-5. For fixed mutation rates, we use $\frac{\chi_{\text{high}}}{n} = 1/(2n)$ and $\frac{\chi_{\text{low}}}{n} = 5/n^2$ which are less than the *error threshold* $\ln(2)/n$ for the 2-tournament EA [19]. For the SA-2mr, we self-adapt mutation rates from $\{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ with a self-adaptation parameter $p_c = 1/(10n)$. For the SA, we set self-adaptive parameters $A = 1.2$ and $p_{\text{inc}} = 0.4$ as previously utilised in [28]. All algorithms use the same population size of $\lambda = 200\ln(n)$, and a uniformly sampled initial population. For symmetric noise, we study algorithms on LeadingOnes and OneMax with noise levels $q \in \{0.2, 0.3, 0.4\}$ and $q \in \{0.2, 0.3, 0.4\}$, respectively. For one-bit noise, we study algorithms on LeadingOnes and OneMax with noise levels $q \in \{0.4, 0.6, 0.8\}$ and $q \in \{0.85, 0.90, 0.95\}$, respectively. For bit-wise noise, we examine algorithms applied to LeadingOnes and OneMax with noise levels $p \in \{0.8/n, 1.0/n, 1.2/n\}$ and $p \in \{5\ln(n)/n, 6\ln(n)/n, 7\ln(n)/n\}$, respectively, which are set with respect to the problem size n. For each setting, we independently run each algorithm 100 times for LeadingOnes and OneMax with problem size $n = 100$ to 200 with step size 10 and $n = 100$ to 500 with step size 40, respectively, and record runtimes. To monitor the behaviour of self-adaptive algorithms, we record mutation parameters χ of individuals during each run. Additionally, we independently perform each self-adaptive algorithm 30 times on each setting. As a comparison, we also run the same experiments without noise. Outcomes for both the one-bit noise and bit-wise noise models are presented in Sections 6.2 and 6.3, respectively. Comprehensive statistical results of the experiments can be found in Appendix C, including medians and hypothesis test results.

6.1 Symmetric Noise

Figures 2-3 illustrate the runtimes on LeadingOnes and OneMax under symmetric noise, respectively. The corresponding statistical

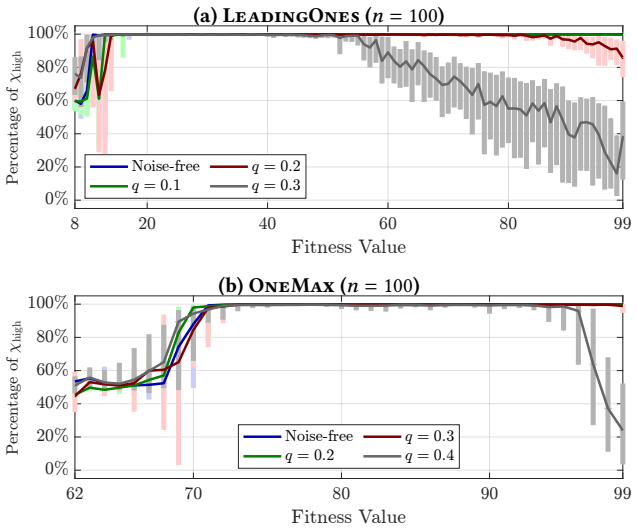

Figure 4: The percentage of $\frac{\chi_{\text{high}}}{n}$ individuals and the highest real fitness value per generation for 2-tour' EA with SA-2mr under symmetric noise with different noise levels ($C = 0$, 30 runs) .

results are displayed in Tables 2-5 and Tables 6-9, respectively. Note that the y-axes in Figures 2 (c)-(d) and Figures 7 (a)-(d) are log-scaled, and all runtimes are divided by n^2 for LeadingOnes and $n\ln(n)$ for OneMax, respectively. These divisions correspond to the well-known runtime results for the LeadingOnes and OneMax function in noise-free scenarios. Note that the runtime of the 2-tournament EA using the high mutation rate exceeds the evaluation budget of 5×10^7 for optimising LeadingOnes for $n \geq 150$ under symmetric noise with noise level $q = 0.2$, and for $n \geq 100$ with noise level $q = 0.3$. Similarly, on OneMax, the runtime of the 2-tournament EA using the high mutation rate exceeds the evaluation budget of 2×10^8 for $n \geq 110$ under symmetric noise with noise level $q = 0.4$.

From Theorems 2.1 and 2.2, we can conclude that the runtime of the 2-tournament EA using the mutation rate $\chi/n = 0.5/n$ on

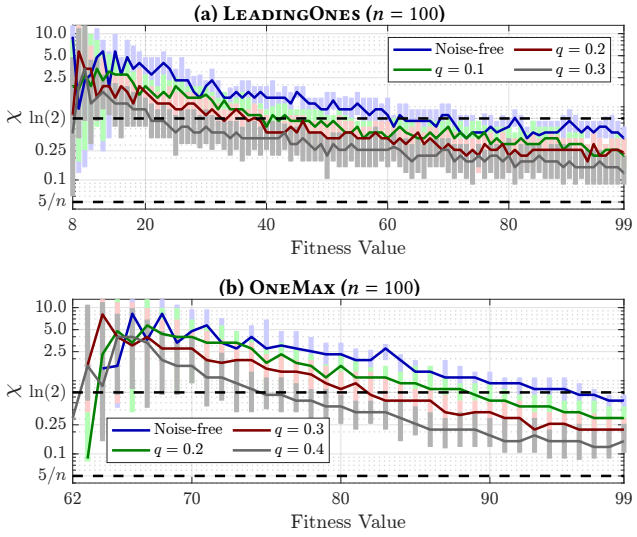

Figure 5: Real fitness and mutation parameter of the highest real fitness individual per generation of 2-tour' EA with SA under symmtric noise with different noise levels ($C = 0$, 30 runs).

LeadingOnes under symmetric noise is polynomial when the noise level $q < 0.1756$, and exponential when $q > 0.1757$. Figure 2 supports these theoretical results, indicating that using a high mutation rate of $\chi/n = 0.5/n$ may fail to optimise LeadingOnes under high-level noise $q \geq 0.2$. On the other hand, employing a low mutation rate is slower than using a high mutation rate under low-level symmetric noise ($q \leq 0.1$) when optimising LeadingOnes. However, the 2-tournament EA using SA-2mr achieves comparable performance to the high mutation rate when the noise level is $q \leq 0.1$. Furthermore, it outperforms the low mutation rate under high-level symmetric noise, specifically when $q \geq 0.2$. Most notably, the 2-tournament EA using SA outperforms all other algorithms across all tested noise levels.

From Figures 3 (a), (b), (c), it is evident that the 2-tournament EA employing a low mutation rate is slower than using a high mutation rate under low-level symmetric noise (i.e., $q \leq 0.3$) when optimising OneMax. However, as shown in Figure 3 (d), the high mutation rate may fail to optimise LeadingOnes under high-level noise, whereas employing a low mutation rate can be more efficient. Similarly, the 2-tournament EA using SA-2mr achieves performance comparable to the high mutation rate when the noise level is $q \leq 0.3$. Furthermore, it outperforms the low mutation rate under high-level symmetric noise, specifically when $q = 0.4$. Most notably, the 2-tournament EA using SA outperforms all other algorithms across all tested noise levels.

Figures 4-5 present the relationships between mutation rates and real fitness values under different levels of one-bit noise in SA-2mr and SA, respectively. The lines indicate the median of values of 30 runs. The corresponding shadows indicate the IQRs. We observe a decrease in the mutation rate when the noise level increases on LeadingOnes and OneMax in both self-adaptations.

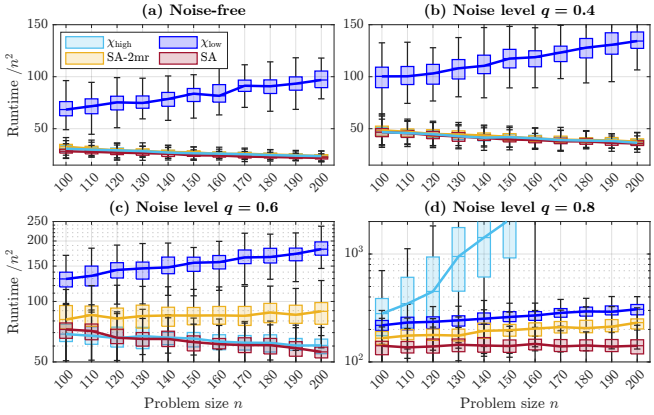

Figure 6: Runtimes of 2-tournament EAs on LeadingOnes under one-bit noise with different noise levels.

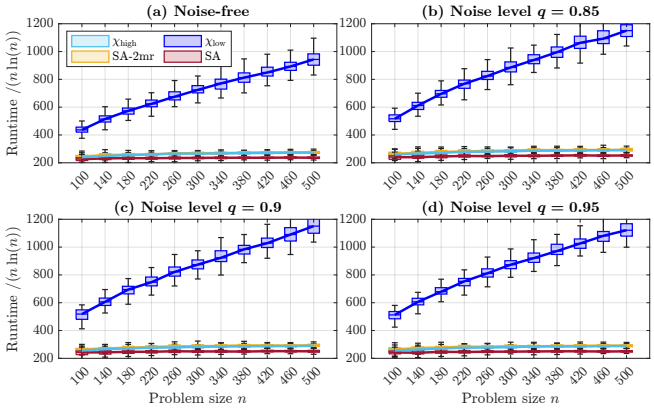

Figure 7: Runtimes of 2-tournament EAs on OneMax under one-bit noise with different noise levels.

Particularly, using SA not only reduces the mutation rate below the error threshold ($\chi/n < \ln(2)/n$) [19], but also furthermore reduces it with respect to the noise level on LeadingOnes and OneMax.

6.2 One-bit Noise

Figures 6-7 illustrate runtimes on LeadingOnes and OneMax under one-bit noise, respectively. The corresponding statistical results are displayed in Tables 10-12 and Tables 13-15, respectively. Note that the y-axes in Figures 6 (c)-(d) and Figures 7 (a)-(d) are log-scaled, and all runtimes are divided by n^2 for LeadingOnes and $n \ln(n)$ for OneMax, respectively.

From Figures 6 (a), (b), (c), it is evident that the 2-tournament EA employing a low mutation rate is slower than using a high mutation rate under low-level one-bit noise (i.e., $q \leq 0.6$) when optimising LeadingOnes. Conversely, utilising a high mutation rate results in faster optimisation. However, as shown in Figure 6 (d), the high mutation rate may fail to optimise LeadingOnes under high-level noise. Note that the runtime of the 2-tournament EA utilizing a high mutation rate exceeds the evaluation budget of 2×10^8 when optimising LeadingOnes for $n \geq 170$ under one-bit noise with noise level $q = 0.8$. Specifically, the runtimes of using

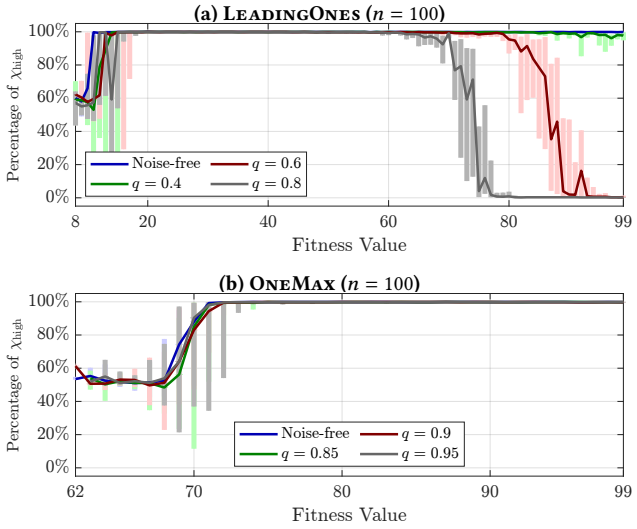

Figure 8: The percentage of $\frac{\chi_{\text{high}}}{n}$ individuals and the highest real fitness value per generation for 2-tour' EA with SA-2mr under one-bit noise with different noise levels (30 runs).

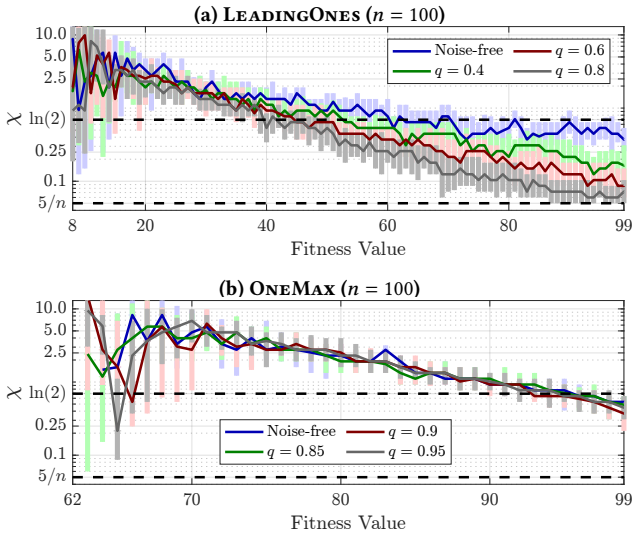

Figure 9: Real fitness and mutation parameter of the highest real fitness individual per generation of 2-tour' EA with SA under one-bit noise with different noise levels (30 runs).

a high mutation rate increase sharply as the problem size grows under high-level one-bit noise ($q = 0.8$), whereas employing a low mutation rate can be more efficient. This observation is consistent with the theoretical study presented in Section 3. On the other hand, the 2-tournament EA using SA-2mr achieves performance comparable to the high mutation rate when the noise level is $q \leq 0.4$ and is only slightly slower than the high mutation rate when the noise level is $q = 0.6$. Furthermore, it outperforms the low mutation rate under high-level one-bit noise, specifically when

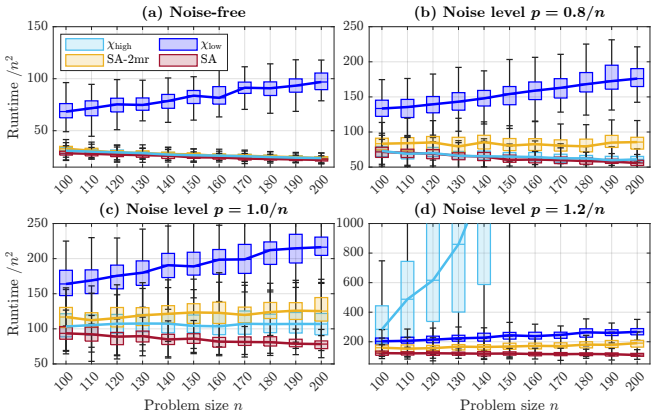

Figure 10: Runtimes of 2-tournament EAs on LeadingOnes under bit-wise noise with different noise levels.

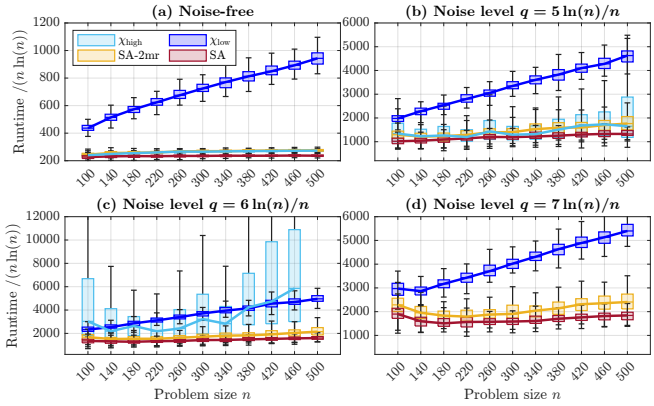

Figure 11: Runtimes of 2-tournament EAs on OneMax under bit-wise noise with different noise levels.

$q = 0.8$. Most notably, the 2-tournament EA using SA outperforms all other algorithms across all tested noise levels.

In Figure 7, which presents results on the OneMax problem, both the high mutation rate EA and self-adaptive EAs outperform the low mutation rate EA across all noise levels. The efficiency of the high mutation rate under high-level noise on OneMax can be explained by Theorem 4 in [21], which states that the 2-tournament EA with a constant mutation parameter χ can achieve the optimum in expected time $O(n \log(n))$ on OneMax under any level of one-bit noise. Despite this, the 2-tournament EA using SA-2mr is only marginally slower than the high mutation rate for all noise levels, and using SA results in faster performance compared to all others.

Similar with Section 6.1, Figures 8-9 show a decrease in the mutation rate when the noise level increases on LeadingOnes in both self-adaptations, where the results are consistent with the MOSA-EA in [28].

6.3 Bit-wise Noise

Figures 10-11 illustrate runtimes on LeadingOnes and OneMax under bit-wise noise, respectively. The corresponding statistical results are displayed in Tables 17-18 and Tables 19-21, respectively.

Note that the y-axes in Figures 11 (c)-(d) are log-scaled. In Figure 10, we observe that the 2-tournament EA employing SA-2mr is faster than the low mutation rate and slower than the high mutation rate for noise levels $p = 0.8/n$ and $1.0/n$, but the gap is not substantial. Consistent with previous observations in the one-bit noise model, under high-level noise $p = 1.2/n$, using SA-2mr outperforms all static algorithms. Note that the runtime of the 2-tournament EA utilizing a high mutation rate exceeds the evaluation budget of 2×10^8 when optimising LeadingOnes and OneMax for $n \geq 160$ and $n \geq 100$ under bit-wise noise with noise levels $p = 1.2/n$ and $7\ln(n)/n$, respectively. Furthermore, the 2-tournament EA using SA consistently achieves the best performance regardless of the noise level. In Figure 11, the 2-tournament EA employing SA-2mr demonstrates better performance than the low mutation rate. Additionally, it exhibits comparable performance to the high mutation rate when the noise level is $p = 5\ln(n)/n$ and faster performance for other noise levels, namely $p = 6\ln(n)/n$ and $7\ln(n)/n$. As always, the 2-tournament EA using SA demonstrates consistently the best performance in this setting. Similar to the results of symmetric and one-bit noise, Figures 12-13 (shown in Appendix D) show that self-adaptive EAs self-adapt the mutation rate to the noise level.

7 CONCLUSION

In this paper, we conduct runtime analysis and empirical analysis on the 2-tournament EAs with self-adaptive mutation rates in a noisy environment. Although the noise model examined in the theoretical study is relatively simplistic and artificial, our findings still provide a compelling indication that the self-adaptive EA remarkably adapts to the presence of noise. The empirical results further affirm that self-adaptation can adjust mutation rates according to noise, thereby leading to more efficient optimisation than other algorithms. Future work includes the investigation on more parameter control mechanisms including self-adaptation and self-adjusting, under noise, e.g., [7, 8, 16, 17, 23, 28, 29].

ACKS. Lehre was supported by a Turing AI Fellowship (EPSRC grant ref EP/V025562/1).

REFERENCES

[1] 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* 24, 4 (2020), 650–663.

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

[3] 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 (GECCO '21)*. Association for Computing Machinery, 1133–1141.

[4] Duc-Cuong Dang and Per Kristian Lehre. 2015. 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*. ACM Press, 62–68.

[5] 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, 803–813.

[6] Raphaël Dang-Nhu, Thibault Dardinier, Benjamin Doerr, Gautier Izacard, and Dorian Nogneng. 2018. A New Analysis Method for Evolutionary Optimization of Dynamic and Noisy Objective Functions. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18)*. Association for Computing Machinery, 1467–1474.

[7] Benjamin Doerr and Carola Doerr. 2018. Optimal Static and Self-Adjusting Parameter Choices for the $(1 + (\lambda, \lambda))$ Genetic Algorithm. *Algorithmica* 80, 5 (2018), 1658–1709.

[8] Benjamin Doerr, Carola Doerr, and Johannes Lengler. 2019. Self-Adjusting Mutation Rates with Provably Optimal Success Rules. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)*. Association for Computing Machinery, 1479–1487.

[9] Benjamin Doerr and Timo Kötzing. 2021. Multiplicative Up-Drift. *Algorithmica* 83, 10 (2021), 3017–3058.

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

[11] Carola Doerr. 2020. Complexity Theory for Discrete Black-Box Optimization Heuristics. In *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, Benjamin Doerr and Frank Neumann (Eds.). Springer International Publishing, 133–212.

[12] Stefan Droste. 2004. Analysis of the (1+1) EA for a Noisy OneMax. In *Genetic and Evolutionary Computation – GECCO 2004*. Vol. 3102. Springer Berlin Heidelberg, 1088–1099.

[13] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2002. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science* 276, 1 (2002), 51–81.

[14] Tobias Friedrich, Timo Kötzing, Martin S. Krejca, and Andrew M. Sutton. 2016. Robustness of Ant Colony Optimization to Noise. *Evolutionary Computation* 24, 2 (2016), 237–254.

[15] Christian Gießen and Timo Kötzing. 2016. Robustness of Populations in Stochastic Environments. *Algorithmica* 75, 3 (2016), 462–489.

[16] 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 (FOGA '21)*. Association for Computing Machinery.

[17] Mario Alejandro Hevia Fajardo and Dirk Sudholt. 2022. Hard Problems Are Easier for Success-Based Parameter Control. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22)*. Association for Computing Machinery, 796–804.

[18] Yaochu Jin and Jürgen Branke. 2005. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation* 9, 3 (2005), 303–317.

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

[20] Per Kristian Lehre and Phan Trung Hai Nguyen. 2021. Runtime Analyses of the Population-Based Univariate Estimation of Distribution Algorithms on LeadingOnes. *Algorithmica* 83, 10 (2021), 3238–3280.

[21] 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, 1160–1168.

[22] Per Kristian Lehre and Xiaoyu Qin. 2022. More Precise Runtime Analyses of Non-elitist Evolutionary Algorithms in Uncertain Environments. *Algorithmica* (2022).

[23] Per Kristian Lehre and Xiaoyu Qin. 2022. Self-Adaptation via Multi-Objectivisation: A Theoretical Study. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22)*. Association for Computing Machinery, 1417–1425.

[24] 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).

[25] Chao Qian, Chao Bian, Wu Jiang, and Ke Tang. 2019. Running Time Analysis of the (1+1)-EA for OneMax and LeadingOnes Under Bit-Wise Noise. *Algorithmica* 81, 2 (2019), 749–795.

[26] Chao Qian, Chao Bian, Yang Yu, Ke Tang, and Xin Yao. 2021. Analysis of Noisy Evolutionary Optimization When Sampling Fails. *Algorithmica* 83, 4 (2021), 940–975.

[27] Chao Qian, Yang Yu, Ke Tang, Yaochu Jin, Xin Yao, and Zhi-Hua Zhou. 2018. On the Effectiveness of Sampling for Evolutionary Optimization in Noisy Environments. *Evolutionary Computation* 26, 2 (2018), 237–267.

[28] Xiaoyu Qin and Per Kristian Lehre. 2022. Self-adaptation via Multi-objectivisation: An Empirical Study. In *Parallel Problem Solving from Nature – PPSN XVII*. Springer International Publishing, 308–323.

[29] Amirhossein Rajabi and Carsten Witt. 2020. Self-Adjusting Evolutionary Algorithms for Multimodal Optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference (GECCO '20)*. Association for Computing Machinery, 1314–1322.

[30] Dirk Sudholt. 2013. A New Method for Lower Bounds on the Running Time of Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 17, 3 (2013), 418–435.

[31] Dirk Sudholt. 2021. Analysing the Robustness of Evolutionary Algorithms to Noise: Refined Runtime Bounds and an Example Where Noise is Beneficial. *Algorithmica* 83, 4 (2021), 976–1011.