

Improved Upper Bound of the $(1 + (\lambda, \lambda))$ GA Escape Time from Local Optima on Jump Functions

Anton Ereemeev
eremeev@ofim.oscsbras.ru
Sobolev Institute of Mathematics
Novosibirsk, Russia
Dostoevsky Omsk State University
Omsk, Russia

Dmitriy Silaev
silaev@ofim.oscsbras.ru
Sobolev Institute of Mathematics
Novosibirsk, Russia

Valentin Topchii
topchij@gmail.com
Sobolev Institute of Mathematics
Novosibirsk, Russia

Abstract

We consider the genetic algorithm $(1 + (\lambda, \lambda))$ GA from (Doerr, Doerr and Ebel, 2015) with tunable parameters of the mutation rate, crossover bias, and two intermediate population sizes, and study the time it escapes from the plateau in the case of Jump_k fitness function when np tends to infinity. The main result of this work is a tightened upper bound on the escape time from the work of Antipov, Doerr and Karavaev (2022). Besides that, the obtained bound applies to a wider range of algorithm parameters.

CCS Concepts

• Theory of computation → Theory of randomized search heuristics.

Keywords

Genetic Algorithm, Optimization Time, Limit Theorems, Local Optima

ACM Reference Format:

Anton Ereemeev, Dmitriy Silaev, and Valentin Topchii. 2026. Improved Upper Bound of the $(1 + (\lambda, \lambda))$ GA Escape Time from Local Optima on Jump Functions. In *Genetic and Evolutionary Computation Conference (GECCO Companion '26)*, July 13–17, 2026, San Jose, Costa Rica. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3795101.3805384>

1 Introduction

Given a pseudo-Boolean maximization problem $f : \{0, 1\}^n \rightarrow \mathbb{R}$, an evolutionary algorithm (EA) seeks for the maximum of f , using a population of one or more individuals, corresponding to the points in the solution space $\{0, 1\}^n$, and the *fitness* of the individuals is equal to the value of the objective function $f(x)$. New test points in the EAs are produced using the random operators of *mutation* and *crossover*. When using the latter, the EA is usually called a *genetic algorithm*. One of the most frequently used mutation operators is the *standard mutation* [7], where each bit of the given string $x \in \{0, 1\}^n$, independently of the other bits, changes its value with a given probability p . In this paper, we will assume that in the case of standard mutation, at each iteration of the EA, the number of bits ℓ to be mutated is selected, using the binomial distribution $\text{Bin}(n, p)$.

Each new descendant is obtained from the parent individual by making changes in randomly selected ℓ bits.

The authors of [6] developed the $(1 + (\lambda, \lambda))$ genetic algorithm (GA) for pseudo-Boolean optimization, where a crossover operator is able to eliminate some “unsuccessful” mutations. At each iteration of the $(1 + (\lambda, \lambda))$ GA, λ offspring are generated from a single parent individual independently of each other, at the Hamming distance ℓ from the parent, $\ell \sim \text{Bin}(n, p)$. After that, the best individual in terms of fitness is selected from these λ solutions and the crossover operator is applied, combining it with the parent, given a crossover parameter c : With probability c the crossover operator uses bits from the best child, and with probability $1 - c$, it uses bits from the parent solution. This way λ individuals are created, and the best of these λ individuals is accepted as a new parent, provided it is at least as fit as the previous parent. Otherwise the current individual remains unchanged. Theoretical analysis of optimization time showed that the algorithm $(1 + (\lambda, \lambda))$ GA for many values of configurable parameters turns out to be asymptotically faster on the ONEMAX fitness function than most classical evolutionary algorithms. Here and below, $\text{ONEMAX}(x) := \sum_{i=1}^n x_i$ for any $x \in \{0, 1\}^n$. In particular, as shown in [4] for the ONEMAX , the optimal choice of fixed mutation parameter p for the entire running time gives the optimization time $E[T] = \Theta\left(n\sqrt{\log(n) \log \log \log(n) / \log \log n}\right)$. In [2], the first mathematical runtime analysis for the $(1 + (\lambda, \lambda))$ GA optimizing a multimodal optimization problem, namely the classical Jump_k benchmark, was carried out. It was observed there that a combination of aggressive mutation with crossover as repair mechanism works even better on this benchmark. The $(1 + (\lambda, \lambda))$ GA can optimize jump functions with gap size $k \leq n/4$ in expected time at most $n^{(k+1)/2} e^{O(k)} k^{-k/2}$. This is almost the square root of $\Omega(n^k)$, the runtime many classic mutation-based algorithms have. For the problem of leaving the local optimum of JUMP , the authors of [2] perform parameters tuning that leads to a runtime $(n/k)^{k/2} e^{O(k)}$.

It is shown in [1] that choosing *all* parameter values (population size λ , mutation rate p , and crossover bias c) in each iteration randomly, using power-law distribution, is a robust approach. It was proved that this algorithm on the one hand can imitate simple hill-climbers on problems like ONEMAX , LEADINGONES , or $\text{Minimum Spanning Tree}$. On the other hand, the algorithm is also efficient on JUMP functions, where the best static parameters are very different from those necessary to optimize simple problems.

In the present paper, we consider the $(1 + (\lambda, \lambda))$ GA from [6] with tunable parameters of mutation rate p , crossover bias c , and two intermediate population sizes λ_M and λ_C , and study the time to escape from the local optima in the case of Jump_k fitness function



when np tends to infinity. The main result of this work is a tightened upper bound on the time to escape a local optimum $(n/k)^{k/2} e^{\Theta(k)}$ from [2]. Besides that, the obtained bound applies to a wider range of algorithm's parameters.

2 Preliminaries

Let $\mathbb{N}_{0,n}$ denote the set of all integers from 0 to n . The notation $\ell \in \text{Bin}(n, p)$, where $n \in \mathbb{N}$ and $p \in (0, 1)$, means that the random variable ℓ has a binomial probability distribution

$$P(\ell = l) = \binom{n}{l} p^l q^{n-l} =: P_p(n, l), \quad l \in \mathbb{N}_{0,n},$$

where $q = 1 - p$.

2.1 The $(1 + (\lambda, \lambda))$ Genetic Algorithm

We consider the same version of the $(1 + (\lambda, \lambda))$ GA as in [2] (analogous to the one initially proposed in [5]) with the following outline.

Algorithm 1. $(1 + (\lambda, \lambda))$ GA with the offspring population sizes λ_M and λ_C , mutation rate p , and crossover bias c , maximizing an objective function $f : \{0, 1\}^n \rightarrow \mathbb{R}$.

1. $x \leftarrow$ random bit string of length n ;
2. **while not terminated do**
3. Mutation phase:
4. Choose $\ell \sim \text{Bin}(n, p)$;
5. **for** $i \in [1.. \lambda_M]$ **do**
6. $x^{(i)} \leftarrow$ a copy of x ;
7. Flip ℓ bits in $x^{(i)}$ chosen uniformly at random;
8. **end**
9. $x' \leftarrow \arg \max_{z \in \{x^{(1)}, \dots, x^{(\lambda_M)}\}} f(z)$;
10. Crossover phase:
11. **for** $i \in [1.. \lambda_C]$ **do**
12. Create $y^{(i)}$ by taking each bit from x' with
13. probability c and from x with probability $1 - c$;
14. **end**
15. $y \leftarrow \arg \max_{z \in \{y^{(1)}, \dots, y^{(\lambda_C)}\}} f(z)$;
16. **if** $f(y) \geq f(x)$ **then**
17. $x \leftarrow y$;
18. **end**
19. **end**

In what follows, we will refer to execution of Lines 3–18 of Algorithm 1 as *one iteration* of the algorithm. Note that sometimes in our analysis we consider an iteration conditional on ℓ being chosen, and in this case we use the symbol l instead of ℓ to highlight that this is a fixed number. We will refer to Lines 3–9 of Algorithm 1 as the *mutation phase*, and Lines 10–15 as the *crossover phase*.

2.2 Jump Functions

Let us define the function JUMP_k , where k is a parameter, $2 \leq k \leq n$, in terms of ONEMAX :

$$\text{JUMP}_k(x) := \begin{cases} n + 1 & \text{if } \text{ONEMAX}(x) = n \\ k + \text{ONEMAX}(x) & \text{if } \text{ONEMAX}(x) \leq n - k \\ n - \text{ONEMAX}(x) & \text{otherwise} \end{cases}.$$

The global optimum of JUMP_k corresponds to the bit string of all ones, which we denote by x_m . The infinite family of functions

JUMP_k serves as an example of a multimodal function, where all local optima are equally distant from the unique global optimum and a number of papers are devoted to the runtime analysis of evolutionary algorithms on these functions, see the references in [2].

3 Technical Definitions

Suppose that an individual x have the norm $|x| = n - k$, i.e. x is in a local maximum of JUMP_k . Let $p_M^{(1)}(n, l, k, p)$ be the probability that k zero bits and $l - k$ one bits are flipped during a single mutation in Line 7 of Algorithm 1 to the individual x . We denote this event by $M^{(1)}(n, l, k, p)$. When the event $M^{(1)}(n, l, k, p)$ occurs, the resulting individual $\tilde{x}$ has the norm $|\tilde{x}| = n - l + k$. Otherwise, $\tilde{x}$ has the norm $|\tilde{x}| < n - l + k$. The explicit form of the probability of event $M^{(1)}(n, l, k, p)$ is

$p_M^{(1)}(n, l, k, p) = P_{n,k}(l, k) = (n - k)!l! / ((l - k)!n!)$, $k \leq l$, (1) here $P_{n,k}(l, s) := \binom{k}{s} \binom{n-k}{l-s} / \binom{n}{l}$ is the hypergeometric distribution and we assume $P_{n,k}(l, k) = 0$ for $l < k$.

Let $p_M^{(\lambda_M)}(n, l, k, p)$ be the probability that, given λ_M independent mutations of x , the event $M^{(1)}(n, l, k, p)$ occurs at least once throughout the mutation phase. We denote this event by $M^{(\lambda_M)}(n, l, k, p)$. The event $M^{(\lambda_M)}(n, l, k, p)$ leads to the choice of the individual x' with the norm $|x'| = n - l + k$. If not all k positions with “0” components in x have mutated, then $\tilde{x}$ has a norm $|\tilde{x}| < n - l + k$. In this notation, the following relation holds:

$$p_M^{(\lambda_M)}(n, l, k, p) = 1 - \left(1 - p_M^{(1)}(n, l, k, p)\right)^{\lambda_M}. \quad (2)$$

Let $p_C^{(1)}(n, l, k, p, c) = c^k (1 - c)^{l-k}$ be the probability that in the crossover phase, within a single crossover of x and x' with $|x'| = n - l + k$ in Lines 12, 13, only the k zero-components in x are replaced (i.e. these components are selected from x'). We denote this event by $C^{(1)}(n, l, k, p, c)$. Let $p_C^{(\lambda_C)}(n, l, k, p, c)$ be the probability that throughout the crossover phase, conditioned that the event $M^{(\lambda_M)}(n, l, k, p)$ occurred (i.e., in the case of $|x| = n - k$ and $|x'| = n - l + k$), the event $C^{(1)}(n, l, k, p, c)$ will occur at least once. We denote this event by $C^{(\lambda_C)}(n, l, k, p, c)$. Then we have

$$p_C^{(\lambda_C)}(n, l, k, p, c) = 1 - \left(1 - p_C^{(1)}(n, l, k, p, c)\right)^{\lambda_C} = 1 - (1 - c^k (1 - c)^{l-k})^{\lambda_C}. \quad (3)$$

Let us denote by $A_{\lambda_M, \lambda_C}(n, l, k, p) = M^{(\lambda_M)}(n, l, k, p) C^{(\lambda_C)}(n, l, k, p, c)$ the event in which an iteration of the algorithm results in creating the optimum, i.e. we get $y = x_m$. By definition, $p_{\lambda_M, \lambda_C}(n, l, k, p) := P(A_{\lambda_M, \lambda_C}(n, l, k, p))$ is the probability of reaching the optimum as a result of an iteration, it is equal to

$$p_{\lambda_M, \lambda_C}(n, l, k, p, c) = p_M^{(\lambda_M)}(n, l, k, p) p_C^{(\lambda_C)}(n, l, k, p, c). \quad (4)$$

4 Asymptotic Approach

Our goal is to bound the expected number of fitness evaluations and the expected number of iterations till the global optimum is reached, if the algorithm is started in a local optimum. Let us denote these random variables by T_F and T_I , respectively.

The overall proof scheme is quite natural and similar to the proof of Lemmas 6 and 7 and Theorem 8 from [2].

Suppose that on some set $B = B(n) \in \mathbb{N}_n := \{1, 2, \dots, n\}$ of some positive probability $P(B) > C_B > 0$ the following bound is valid

$$p_{\lambda_M}(n, l, k, p) > C_M > 0, \quad l \in B. \quad (5)$$

Here the sequences $C_B = C_B(n) > 0$ and $C_M = C_M(n) > 0$ may depend on n . If condition (5) and the following inequality hold

$$\sum_{l \in B} p_C^{(\lambda_C)}(n, l, k, p, c) P_p(n, l) \geq C_{C,B} > 0,$$

for binomial distribution $P_p(n, l)$ and some sequence $C_{C,B} = C_{C,B}(n) > 0$ then the inequality (6) below describes the probability of reaching the optimum as a result of one iteration:

$$\begin{aligned} p_{\lambda_M, \lambda_C}(n, k, p, c) &:= E[p_{\lambda_M, \lambda_C}(n, \ell, k, p, c)] \\ &\geq C_M \sum_{l \in B} p_C^{(\lambda_C)}(n, l, k, p, c) P_p(n, l) \geq C_M C_{C,B} > 0. \end{aligned} \quad (6)$$

The expected number of iterations $E[T_I]$ and the of fitness evaluations $E[T_F]$ till reaching the optimum may be expressed as

$$E[T_I] = p_{\lambda_M, \lambda_C}^{-1}(n, k, p, c) \leq \frac{1}{C_M C_{C,B}}, \quad E[T_F] \leq \frac{\lambda_M + \lambda_C}{C_M C_{C,B}}. \quad (7)$$

If $p_{\lambda_C}(n, l, k, p, c) > C_C > 0, l \in B$, then $C_{C,B} \geq C_C C_B$. Here we have used the fact that T_I has a geometric distribution with a probability of success $p_{\lambda_M, \lambda_C}(n, k, p, c)$.

Let us discuss the choice of the set B . In [2] (in the proof of Lemma 6) it is chosen in the form $B = [np, 2np]$, where $np \geq 2k$. We propose to choose it in the form $B \subseteq [\max\{k, np - N\sqrt{npq}\}, np + N\sqrt{npq}]$, where $N \in \mathbb{R}^+$. We will analyze only the case $k \leq np - N\sqrt{npq}$ and in the final asymptotic steps we set $B = \Delta_n(N) := [np - N\sqrt{npq}, np + N\sqrt{npq}]$. Dropping the $k \leq np - N\sqrt{npq}$ condition is easy, but it will lead to cumbersome calculations.

By the Moivre–Laplace theorem (see e.g. Theorem 5.3.1 from [3]) for $N = N_n = o((npq)^{1/6})$ we have

$$P(\ell \in \Delta_n(N)) \sim 2\Phi(N) - 1 =: c_\Phi(N), \quad (8)$$

where $\Phi(N)$ is the cumulative distribution function of the standard normal random value. In the case of $N \rightarrow \infty$ we get

$$P(\ell \in \Delta_n(N)) \rightarrow 1, \quad P(\ell \notin \Delta_n(N)) \rightarrow 0. \quad (9)$$

Note that $2\Phi(2) - 1 = 0.99730020$ for $N = 2$ and $2\Phi(5) - 1 = 0.99999942$ for $N = 5$, and then converges very quickly to 1. But increasing of N worsens the estimates for $p_C^{(\lambda_C)}(n, l, k, p, c) \geq C_C$ and $p_M^{(\lambda_M)}(n, l, k, p) \geq C_M$ for $l \in B$.

The length of the interval $\Delta_n(N)$ is infinitesimally small relative to $[np, 2np]$, this is the reason why the upper bounds on the runtime obtained by means of $B = [np, 2np]$ are by more than a factor of $2^{k(1+o(1))}$ higher than ours.

Let us consider the asymptotic behavior of $p_M^{(1)}(n, l, k, p)$ as $n \rightarrow \infty$. First, we present some technical results.

LEMMA 1. Suppose $k \leq (1-\varepsilon)n$ and $l \geq (1+\varepsilon)k$ for some $\varepsilon \in (0, 1)$, then the following representation is valid:

$$p_M^{(1)}(n, l, k, p) = \sqrt{(1-k/n)/((1-k/l))} e^{n\epsilon_1(k/n) - l\epsilon_1(k/l)} (l/n)^k R_{n,l,k},$$

where $\epsilon_1(x) = (1-x)\ln(1-x) + x = \sum_{i=2}^{\infty} \frac{x^i}{i(i-1)}$, $R_{n,l,k} = R(n-k)R(l)R^{-1}(l-k)R^{-1}(n)$, for some function $R(s)$, that satisfies the inequalities $1 < e^{\frac{1}{12s+1}} < R(s) < e^{\frac{1}{12s}} < 1.08690405$. Note that $R_{n,l,k} = 1 + o(1)$ if $n, n-k, l, l-k \rightarrow \infty$.

We analyze the properties of the second bound from formulas (7). Let us write the relations (2) and (3) in the form

$$p_i^{(L_i)} = 1 - (1 - p_i)^{L_i} = 1 - e^{L_i \ln(1-p_i)} = 1 - e^{-L_i p_i (1+o(1))}, \quad (10)$$

as $0 < p_i = p_i(n) \rightarrow 0, L_i = L_i(n) \rightarrow \infty$, for $n \rightarrow \infty, i = 1, 2$. Note that for $L_i p_i \rightarrow 0$ holds following bound

$$p_i^{(L_i)} = L_i p_i (1 + o(1)). \quad (11)$$

Let us introduce the notation $\mathcal{P}_{k,p}(c_1)$ for the set of sequences $\{k\} = \{k(n)\}_{n \in \mathbb{N}}$ and $\{p\} = \{p(n) \in (0, p_0) \subseteq (0, 1)\}_{n \in \mathbb{N}}$ that satisfy the conditions

$$\lim_{n \rightarrow \infty} k(n)p^{-1}(n)/n \leq c_1, \quad (12)$$

where $c_1 \in [0, 1)$ and $p_0 \in [0, 1)$ are fixed numbers. Note that $\Delta_n(N) \subseteq [k+1, n]$ as $n \rightarrow \infty$ in the case $p \in (0, p_0)$. Denote

$$\begin{aligned} \tilde{J}_{n,k,p,c,N} &:= \sqrt{(1-k/n)/(1-p)} (cp)^k e^{-k(1-p)(1+2N^2)/(2np)} \\ &\cdot (1-c)^{np+N\sqrt{npq}-k} e^{-kN\sqrt{q}/\sqrt{np}}. \end{aligned}$$

THEOREM 1. Suppose that $k \in \mathcal{P}_{k,p}(c_1)$, $p \in (0, p_0)$ with any fixed $c_1 \in [0, 1)$, $N \in \mathbb{N}$, $p_0 \in (0, 1)$. Let $k \rightarrow \infty$, $np \rightarrow \infty$ and the conditions $\lambda_M p^k e^{-kN\sqrt{q}/\sqrt{np}} = o(1)$ and $\lambda_C (c^k (1-c)^{np+N\sqrt{npq}-k}) = o(1)$, as $n \rightarrow \infty$, hold, then

$$E[T_I] \leq c_\Phi^{-1}(N) \lambda_M^{-1} \lambda_C^{-1} \tilde{J}_{n,k,p,c,N}^{-1}, \quad E[T_F] \leq (\lambda_M + \lambda_C) E[T_I].$$

If we assume that $\lambda_M = \lambda_C = \lambda$, and conditions $\lambda_M (p(1-k/n)^{1/(np)-1/n} e^{N/\sqrt{np}})^k = c_M^{(0)} (1+o(1))$, $c_M^{(0)} > 0$, and $\lambda_C (c^k (1-c)^{np+N\sqrt{npq}-k}) = o(1)$, as $n \rightarrow \infty$, are satisfied, then

$$\begin{aligned} E[T_I] &\leq \left(c_\Phi(N) \lambda (1-c)^{np+N\sqrt{npq}-k} (1-e^{c_M^{(0)}}) \right)^{-1} (1+o(1)), \\ E[T_F] &\leq 2\lambda E[T_I]. \end{aligned}$$

Let us compare the statement of Theorem 1 above and Theorem 8 from [2] using the example of $E[T_I]$. The upper bound established in [2] is $E[T_I] \leq 4/(q_l \min\{1, \lambda_M(p/2)^k\} \min\{1, \lambda_C c^k (1-c)^{2pn-k}\})$. Let us note the advantages of our formula (16) in comparison with the one just given in (8). Instead of $(p/2)^k$, we use $J_{\lambda_M, n, k, p, N}$, which is almost equivalent to $(p(1+o(1)))^k$, and instead of $(1-c)^{2pn-k}$ from Theorem 8, we use $(1-c)^{np+N\sqrt{npq}-k}$. Instead of the estimate $\min\{1, a\}$, we use the more precise formula $1 - e^{-a}$. The constant $4/q_l$ corresponds to $c_\Phi^{-1}(N)$ here, which can be chosen arbitrarily close to 1. In addition, all constants in the bounds are given explicitly here. The bounds from Theorem 1 significantly refine their analogs from [2]. Naturally, our upper bounds for λ_M and λ_C , leading to infinitesimal $p_{\lambda_M, \lambda_C}(n, k, p, c)$ are significantly lower than the analogs from [2].

THEOREM 2. Suppose that $p \in (0, p_0)$ with any fixed $2 \leq k \in \mathbb{N}$, $p_0 \in (0, 1)$. Let $p = o(1)$, $np \rightarrow \infty$, $c = o(\max\{1, \sqrt{np}\})$, then

$$\begin{aligned} E[T_I] &\leq (1+o(1)) / ((1-e^{-\lambda_M p^k (1+o(1))}) (1-e^{-\lambda_C c^k (1-c)^{np} (1+o(1))})), \\ E[T_F] &\leq (\lambda_M + \lambda_C) E[T_I]. \end{aligned}$$

If we assume that $2 \leq k \in \mathbb{N}$ is fixed, $\lambda_M = \lambda_C = \lambda = \sqrt{n/k}$ and $p = c = \sqrt{k/n}$, then

$$E[T_I] \leq \frac{1+o(1)}{(1-e^{-1})(1-\exp\{-e^{-k}\})}, \quad E[T_F] \leq 2\sqrt{n/k} E[T_I]. \quad (13)$$

We will not go into detail on optimizing the results of Theorem 1 with respect to the parameters λ_M and λ_C , instead we will use the conditions of Corollary 9 [2] with the additional constraint $np \rightarrow \infty$ and present an analogue of the Corollary 9 from [2].

If we relax the condition $k/n = o(1)$, then a natural convenient condition for asymptotic representations in the case described below will be $\sqrt{k/n} = c_0^*(1 + o(1))$, where $c_0^* \in (0, c_1^2)$ is an absolute constant (depending on n). For the sake of compactness, we will write this condition as $\sqrt{k/n} = c^*$, $c^* \in (0, c_1^2)$. In the following theorem, we provide conditions for replacing rough estimates of the type $e^{f(y)(1+o(1))}$ to expressions of the form $e^{f(y)}(1+o(1))$ depending on the rate of convergence of $y = k/n$ to 0 in two cases. This can be done in other cases as well, but it requires many additional conditions on the k/n ratio.

If $p = c = \sqrt{k/n}$, $\lambda_M = \lambda_C = \lambda = \sqrt{n/k}^k$, then Theorem 1 implies

THEOREM 3. *Suppose that $k \in \mathcal{P}_{k,p}(c_1)$, $p \in (0, p_0)$ with any fixed $c_1 \in [0, 1)$, $N \in \mathbb{N}$, $p_0 \in (0, 1)$. Let $c = p = \sqrt{k/n}$, $\lambda_M = \lambda_C = \lambda = \sqrt{n/k}^k$, $k \rightarrow \infty$, then $E[T_F] \leq 2\sqrt{n/k}^k E[T_I]$, and*

$$c_\Phi(N)E[T_I] \leq \begin{cases} \frac{e^k(1+o(1))}{1-e^{-1}}, & k \rightarrow \infty, k = o(n^{1/3}) \\ e^{k+\sqrt{k^3/n}N^2}(1+o(1)), & k\sqrt{1/n} \rightarrow \infty, k = o(\sqrt{n}), \\ e^{k(1+o(1))}, & k/\sqrt{n} \rightarrow \infty, k = o(n), \\ e^{k \ln(1-c_0^*)(1/c_0^*-1)(1+o(1))}, & \sqrt{k/n} = c_0^* + o(1) \in (0, c_1^2). \end{cases} \quad (15)$$

It is easy to see from the proof of Corollary 9 [2] that the estimate of $E[T_I]$ obtained in that proof is greater than (15) almost by a constant factor which is greater than one, multiplied by $(1-c)^{-np}2^k$.

We do not describe here the case of $\lambda \sim r\sqrt{n/k}^k$ for $r \neq 1$, but this may be easily done, using above asymptotic results.

We note that our theorems give an explicit form of upper bounds for the averages, which essentially depend on the main parameters depending on n . In the work [2], similar estimates are given without dependence on the main parameters and with accuracy to unknown bounded constants. Our estimates are more precise than the ones in [2], and their quality can be naturally checked in computational experiments. The general approach without an explicit form of the asymptotics of the constants does not allow to estimate the quality of heterogeneous estimates. Of course, the transition to a logarithmic scale removes almost all problems, but this will be a very rough estimate of the quality of the estimates.

5 Computational Experiment

In this section we evaluate the accuracy of the estimate (13) from Theorem 2 experimentally. In the experiments we used the problem sizes $n \in \{2^i : i = 3, \dots, 10\}$, and $k = 2, \dots, 6$. We made 100 independent runs of the algorithm with $c = p = \sqrt{k/n}$, $\lambda_M = \lambda_C = \lambda = \sqrt{n/k}^k$, which was started from a local optimum with k ones. Fig. 1

presents the upper bound $E[T_F] \leq \frac{2\sqrt{n/k}^k(1+o(1))}{\Phi(N)(1-e^{-1})(1-\exp\{-e^{-k}\})}$ for fixed k , based on Theorem 2, as well as the experimental average runtime $\hat{T}_F$. Shaded regions display estimated standard deviations. Note that $\Phi(N) \approx 1$ as $n = 5$.

It may be seen from Fig. 1 that the upper bound is relatively close to the runtime observed in experiments, and even for small values of n and k the asymptotic bound is close to the average runtime.

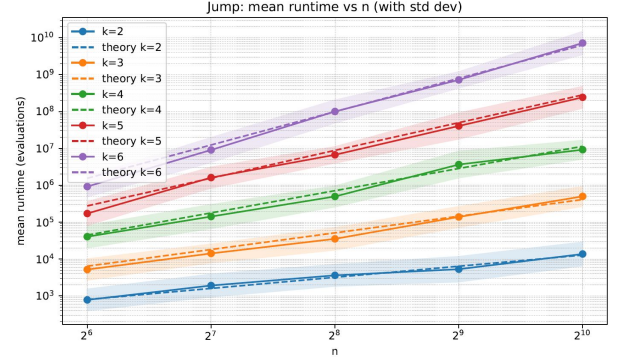


Figure 1: Average runtime as a function of n and k . Dashed lines represent the upper bound from Theorem 2

6 Conclusions

The runtime analysis of the $(1 + (\lambda, \lambda))$ GA on $JUMP_k$ fitness functions from the prospective of Moivre–Laplace theorem. We assumed that the $(1 + (\lambda, \lambda))$ GA has tunable parameters of mutation rate, crossover bias, and two intermediate population sizes, and it starts from a local optima of $JUMP_k$, and that $np \rightarrow \infty$ as $n \rightarrow \infty$. The main result of this work (Theorem 1) is a refined upper bound on the escape time from the local optimum, known from [2]. Different cases of asymptotic behaviour of parameters k, p, c are considered (Theorem 3). Computational experiment performed for some finite k agrees with the theoretical upper bound on the runtime.

7 Acknowledgments

The research was supported by Russian Science Foundation grant N 25-21-00335, <https://rscf.ru/project/25-21-00335/>.

References

- [1] D. Antipov, M. Buzdalov, and B. Doerr. 2024. Lazy parameter tuning and control: Choosing all parameters randomly from a power-law distribution. *Algorithmica* 86 (2024), 442–484.
- [2] D. Antipov, B. Doerr, and V. Karavaev. 2022. A rigorous runtime analysis of the $(1 + (\lambda, \lambda))$ GA on Jump functions. *Algorithmica* 84 (2022), 1573–1602.
- [3] A.A. Borovkov. 2013. *Probability Theory*. Springer-Verlag, London.
- [4] B. Doerr and C. Doerr. 2018. Optimal static and self-adjusting parameter choices for the $(1 + (\lambda, \lambda))$ genetic algorithm. *Algorithmica* 80 (08 2018), 1658–1709.
- [5] B. Doerr, C. Doerr, and F. Ebel. 2013. Lessons from the black-box: Fast crossover-based genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2013*. ACM, New York, NY, 781–788.
- [6] B. Doerr, C. Doerr, and F. Ebel. 2015. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science* 567 (2015), 87–104.
- [7] D.E. Goldberg. 1989. *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley, MA, USA.