Theoretical and Empirical Study of the $(1 + (\lambda, \lambda))$ EA on the LeadingOnes Problem

Vitalii Karavaev
ITMO University
St. Petersburg, Russia

Denis Antipov
ITMO University
St. Petersburg, Russia

Benjamin Doerr
École Polytechnique
Laboratoire d'Informatique (LIX)
Palaiseau, France

ABSTRACT

In this work we provide a theoretical and empirical study of the $(1 + (\lambda, \lambda))$ EA on the LeadingOnes problem. We prove an upper bound of $O(n^2)$ fitness evaluations on the expected runtime for all population sizes $\lambda < n$. This asymptotic bound does not depend on the parameter λ.

We show via experiments that the value of λ has a small influence on the runtime (less than a factor of two). The value of λ that optimizes the runtime is small relative to n. We propose an extension of the existing $(1 + (\lambda, \lambda))$ EA by using different population sizes in the mutation and in the crossover phase of the algorithm and show via experiments that this modification can outperform the original algorithm by a small constant factor.

CCS CONCEPTS

• **Theory of computation → Random search heuristics**.

KEYWORDS

Runtime Analysis, Crossover, Theory

ACM Reference Format:
Vitalii Karavaev, Denis Antipov, and Benjamin Doerr. 2019. Theoretical and Empirical Study of the $(1 + (\lambda, \lambda))$ EA on the LeadingOnes Problem. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3319619.3326910

1 INTRODUCTION

Although the crossover is seen as an essential part of genetic algorithms (GAs), we still experience a lack of the theoretical understanding of this concept. The common trend of the latest years is to fill this gap in our knowledge about the GAs, which has given rise to a plenty of theoretical works that concentrate on finding the reasons of the crossover usefulness.

In the early works like [8, 9, 11] the crossover is seen as a tool that helps an algorithm to effectively leave local optima or a plateau of the fitness function exploiting the diversity of the genotype in the population. Although these works consider relatively artificial functions that were designed to model some particular properties, they inspired other authors to consider some (simplified) real-world problems, e.g., Ising model on rings [6] or on trees [12]. The latter two works showed that even the most simple crossover-based algorithm can asymptotically outperform the mutation-based $(1 + 1)$ evolutionary algorithm (EA) on those problems.

Recently Corus and Oliveto have shown in [2] that the crossover can effectively exploit the diversity of the population and as a consequence outperform by a constant factor any mutation-based EA on the benchmark problem OneMax. This result was surprising, since it was one of the first to show the efficiency of the $(\mu + 1)$ GA on the landscapes, where hill climbers are supposed to be the most effective algorithms.

A totally different point of view on crossover was introduced together with the $(1 + (\lambda, \lambda))$ EA in [4]. This algorithm unlike any canonical GA uses crossover after mutation as a repair mechanism. This leads to a speedup by a superconstant factor compared to any mutation-based algorithm on the OneMax problem. The further research in [3] has shown that with some parameter adaptation mechanism this algorithm can optimize OneMax in a linear time (in terms of the number of fitness evaluations) which at the moment is the best known performance for the algorithms that use only binary unbiased operators [5].

The empirical study of the $(1 + (\lambda, \lambda))$ EA is not so unambiguously optimistic. On the one hand it was empirically shown that the $(1 + (\lambda, \lambda))$ EA is effective on different linear functions, royal road function and some instances of the maximum satisfiability problem in [7]. On the other hand, the other experiments presented in [10] show that the $(1 + (\lambda, \lambda))$ EA is outperformed by the $(1 + 1)$ EA on the problem of the hard-test generation.

To achieve a better understanding of the crossover mechanism, we study the $(1 + (\lambda, \lambda))$ EA on the LeadingOnes problem. The $(1 + (\lambda, \lambda))$ EA is already well-studied on the problems with a strong fitness-distance correlation (such as OneMax) and has shown good performance on such problems. However, there are no such results for problems with weak fitness-distance correlation. In this paper we aim to show that $(1 + (\lambda, \lambda))$ EA perfoms at least as good (in asymptotical sense) as $(1 + 1)$ EA on the LeadingOnes.

The rest of the paper is organized as follows. In Section 2 we introduce the notation that we use in the paper and state the problem. In Section 3 we prove that the expected number of fitness evaluations before the $(1 + (\lambda, \lambda))$ EA reaches the optimum of LeadingOnes is $O(n^2)$ for any $\lambda < n$. In Section 4 we inspect the influence of the parameter λ on the leading constant in the expected runtime. We also propose an improvement of the $(1 + (\lambda, \lambda))$ EA and provide an empirical study of its performance on LeadingOnes.

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3326910

2 PRELIMINARIES AND NOTATION

2.1 Notation

We write $[a..b]$ to denote an integer interval including its borders and $(a..b)$ to denote an integer interval excluding its borders.

By $H(x, y)$ we denote the Hamming distance between bit-strings x and y.

For any probability distribution $\mathcal{L}$ and random variable X, we write $X \sim \mathcal{L}$ to indicate that X follows the law $\mathcal{L}$. We denote the binomial law with parameters $m \in \mathbb{N}$ and $p \in [0, 1]$ by $\mathrm{Bin}(m, p)$.

2.2 Problem Statement

We analyze the $(1 + (\lambda, \lambda))$ EA, that is a crossover-based evolutionary algorithm, on the pseudo-Boolean functions. This algorithm has a parameter λ that can be adjusted during the run [3], however in our work we focus on a stationary value of λ that may be any integer in $[2..n-1]$, where n is the dimension of the search space. Note that we omit $\lambda = 1$, since in this case the $(1 + (\lambda, \lambda))$ EA turns into the $(1 + 1)$ EA, which is already well-studied.

The $(1 + (\lambda, \lambda))$ EA stores a bit-string x that is initialized with a random bit-string. After the initialization it performs iterations that consist of the mutation phase and the crossover phase until some stopping criterion is met. In the mutation phase the algorithm first chooses the mutation strength ℓ from the binomial distribution with parameters n and $\frac{\lambda}{n}$. Then it creates λ mutants $x^{(1)}, \dots, x^{(\lambda)}$, each of them is a copy of x with exactly ℓ bits flipped. The positions of the flipped bits are chosen uniformly at random, independently for each mutant. The mutant with the best fitness is chosen as the winner of the mutation phase x'. In the crossover phase the algorithm creates λ offspring $y^{(1)}, \dots, y^{(\lambda)}$ by applying a biased crossover to x and x' (independently for each offspring). The crossover operator for each position takes a bit value from x with probability $\frac{\lambda-1}{\lambda}$ and it takes a bit value from x' with probability $\frac{1}{\lambda}$ (independently for each position and each offspring). If the best offspring y is not worse than x then it replaces x. The pseudocode of the $(1 + (\lambda, \lambda))$ EA optimizing some pseudo-Boolean function f is shown in Algorithm 1.

Our research focuses on the optimization of the LeadingOnes function, that returns the position of the first zero-bit in its argument. Note that in this paper *we count bit positions in a bit string starting with zero*. More formally, the LeadingOnes is defined as

$$\textsc{LeadingOnes}(x) = \sum_{i=0}^{n-1} \prod_{j=0}^{i} x_j.$$

For brevity we denote the LeadingOnes by f and in the rest of the paper by *the fitness function* we mean the LeadingOnes. The value of f on a bit-string x is *the fitness of* x.

3 RUNTIME ANALYSIS

In this section we denote by i the current fitness, that is, $f(x)$. The main idea of our proof is to show that the mutation phase winner x' has the i-th bit equal to one with probability $\Omega(\frac{\lambda}{n})$. Afterwards, we prove that there is at least a constant probability to generate an offspring y such that the i-th bit of y is taken from x' and all other bits that differ in x and x' are taken from x. For the latter estimate to hold we assume that ℓ does not exceed $2\lambda+1$. Combining the estimations and applying the method of artificial fitness levels from [13] we get an upper bound on the expected runtime of $O(\frac{n^2}{\lambda})$ iterations.

Algorithm 1: The $(1 + (\lambda, \lambda))$ EA maximizing $f : \{0, 1\}^n \to \mathbb{R}$

```
x ← random bit string of length n;
while not terminated do
    Mutation phase:
    Choose ℓ following Bin(n, λ/n);
    for i ∈ [1..λ] do
        x^(i) ← a copy of x;
        Flip ℓ bits in x^(i) chosen uniformly at random;
    end
    x' ← arg max_{z ∈ {x^(1),...,x^(λ)}} f(z);
    Crossover phase:
    for i ∈ [1..λ] do
        y^(i) ← a copy of x;
        Flip each bit in y^(i) that is different in x' with
          probability 1/λ;
    end
    x ← arg max_{z ∈ {y^(1),...,y^(λ),x}} f(z);
end
```

3.1 Mutation Phase

In this section we estimate the probability to obtain an individual with the i-th bit flipped as the winner of the mutation phase.

For that purpose, we notice that when we consider one particular individual $x^{(m)}$ generated in the mutation phase, we can omit the process of first choosing $\ell \sim \mathrm{Bin}(n, \frac{\lambda}{n})$ and then flipping ℓ random bits, but assume that we apply a standard bit mutation with probability $\frac{\lambda}{n}$ to flip each bit, since the distribution over all possible outcomes is the same for these two mutation operators. We regard the two cases of the outcome of this mutation operator in Lemmas 3.1 and 3.2.

Lemma 3.1. *Let $x^{(m)}$ be some particular offspring generated in the mutation phase. Then the probability that the the i-th bit is flipped in $x^{(m)}$ and the total number X of the bits that were flipped in $x^{(m)}$ is at most $2\lambda + 1$ conditional on $f(x^{(m)}) = j < f(x)$ is at least $(1 - e^{-\frac{\lambda}{3}})\frac{\lambda}{n}$.*

Proof. We use the definition of the conditional probability to estimate

$$Pr\left[x_i^{(m)} = 1 \cap X \le (2\lambda + 1) \mid f(x^{(m)}) = j < f(x)\right] = \frac{\Pr\left[x_i^{(m)} = 1 \cap X \le (2\lambda + 1) \cap f(x^{(m)}) = j < f(x)\right]}{\Pr\left[f(x^{(m)}) = j < f(x)\right]}. \quad (1)$$

We assume that $x^{(m)}$ was obtained through a standard bit mutation with mutation rate $\frac{\lambda}{n}$. Hence, denoting by X' the number of flipped bits in positions $(j..n) \setminus \{i\}$ we estimate the probability in

the numerator as

$$\left(\frac{\lambda}{n}\right)^2 \left(1 - \frac{\lambda}{n}\right)^j Pr[X' \leq (2\lambda - 1)], \tag{2}$$

since for this outcome we need to flip bits i and j, not to flip bits $[0..j)$ and not to flip more than $(2\lambda-1)$ bits in other $n-j-2$ positions. Note that $X' \sim \text{Bin}(n - j - 2, \frac{\lambda}{n})$ and is dominated by $\text{Bin}(n, \frac{\lambda}{n})$. Therefore, by Chernoff bounds we have

$$Pr[X' \leq (2\lambda - 1)] = 1 - Pr[X' \geq 2\lambda] \geq 1 - e^{-\frac{\lambda}{3}}. \tag{3}$$

We also estimate the probability in the denominator of (1) as

$$\frac{\lambda}{n}\left(1 - \frac{\lambda}{n}\right)^j, \tag{4}$$

since for this outcome we only need to flip bit j and not to flip bits in positions $[0..j)$. By putting (2), (3) and (4) into (1) we obtain

$$Pr\left[x_i^{(m)} = 1 \cap X \leq (2\lambda + 1) \mid f(x^{(m)}) = j < f(x)\right] \geq \frac{\lambda}{n}\left(1 - e^{-\frac{\lambda}{3}}\right). \qquad \square$$

Lemma 3.2. *Let $x^{(m)}$ be some particular offspring generated in the mutation phase. Then the probability that the the i-th bit is flipped in $x^{(m)}$ and the total number X of the bits that were flipped in $x^{(m)}$ is at most $2\lambda + 1$ conditional on $f(x^{(m)}) \geq f(x)$ is at least $(1 - e^{-\frac{\lambda}{3}})\frac{\lambda}{n}$.*

We omit the proof for reasons of space and since it generally replicates the proof of Lemma 3.1.

From Lemma 3.1 and Lemma 3.2 we conclude that we have two options. The winner of the mutation phase can have fitness j that is less than $f(x)$ and then with probability $(1 - e^{-\frac{\lambda}{3}})\frac{\lambda}{n}$ we flipped no more than $2\lambda + 1$ bits, but surely flipped the i-th bit in it. Or the winner is at least as good as x and then it has no more than $2\lambda + 1$ bits flipped and the i-th bit surely flipped with the same probability. So independently of the selection, we have

$$\Pr[x_i' = 1 \cap H(x, x') \leq (2\lambda + 1)] \geq \left(1 - e^{-\frac{\lambda}{3}}\right)\frac{\lambda}{n}.$$

3.2 Crossover Phase

Now we consider the probability to obtain progress in the crossover phase. For that purpose we estimate the probability to get y such that y_i is a one-bit and none of other bits are changed compared to x (and hence we increase the fitness by at least 1). We estimate the probability to obtain progress in one particular offspring in the following lemma.

Lemma 3.3. *Assume that $x_i' = 1$ and the Hamming distance between x and x' is at most $2\lambda + 1$. Then for any offspring $y^{(m)}$ we have $\Pr[f(y^{(m)}) > f(x)]$ is at least $\frac{1}{16\lambda}$.*

Proof. To obtain an offspring $y^{(m)}$ that is better than x we can take bit i from x' with probability $\frac{1}{\lambda}$ and take all other 2λ bits that differ in x and x' from x with probability $(1 - \frac{1}{\lambda})$ each. Therefore, we have

$$\Pr[f(y^{(m)}) > f(x) \mid x_i' = 1 \cap H(x, x') \leq 2\lambda + 1] \geq \frac{1}{\lambda}\left(1 - \frac{1}{\lambda}\right)^{2\lambda} \geq \frac{1}{16\lambda}. \qquad \square$$

The winner of the crossover phase y is strictly better than x, if at least one offspring $y^{(m)}$ is better than x. Hence, we have

$$\Pr[f(y) > f(x) \mid x_i' = 1 \cap H(x, x') \leq (2\lambda + 1)] \geq 1 - \left(1 - \frac{1}{16\lambda}\right)^{\lambda} \geq 1 - e^{-\frac{1}{16}}.$$

3.3 Upper bound summary

In order to apply the method of artificial fitness levels, for all $i \in [0..n]$ we define the level A_i as a set of all bit-strings of length n with the value of LeadingOnes equal to i. We note the the probability to leave each level is at least the probability that we create x' such that $x_i' = 1$ and $H(x, x') \leq 2\lambda + 1$ multiplied by the probability to create y that is better than x conditional on this event. We estimate such probability as

$$p_i \geq \left(1 - e^{-\frac{\lambda}{3}}\right)\left(1 - e^{-\frac{1}{16}}\right)\frac{\lambda}{n}.$$

Through the method of artificial fitness levels we compute

$$E[T] \leq \sum_{i=1}^{n-1} \frac{1}{p_i} \leq \sum_{i=1}^{n-1} \left(1 - e^{-\frac{\lambda}{3}}\right)^{-1}\left(1 - e^{-\frac{1}{16}}\right)^{-1}\frac{n}{\lambda} = \left(1 - e^{-\frac{\lambda}{3}}\right)^{-1}\left(1 - e^{-\frac{1}{16}}\right)^{-1}\frac{n^2}{\lambda}.$$

Therefore, we have the estimate of $O(\frac{n^2}{\lambda})$ iterations to optimize LeadingOnes function with $(1 + (\lambda, \lambda))$ EA which gives us $O(n^2)$ fitness evaluations, since we perform exactly 2λ fitness evaluations per iteration.

4 EXPERIMENTAL STUDY

In this section we introduce the results of our empirical analysis. The main purpose of this section is to investigate how different values of λ affect the runtime. Therefore we hold n constant and equal to 512 whilst varying λ from $2^0 = 1$ to $2^8 = 256$.

In Figure 1 one can see that generally for different values of λ the runtime of the $(1 + (\lambda, \lambda))$ EA differ from the runtime of the $(1 + 1)$ EA not more than by a factor of two. However, for the small values of λ it outperforms $(1 + 1)$ EA. In particular, for the optimal value $\lambda = 4$ its runtime is less by about 25%, compared by the median values. On the other hand, despite the good performance on the small values of λ the runtime of the $(1 + (\lambda, \lambda))$ EA gets slightly worse with the parameter growth. We assume that the reason for such behavior is that for the small values of λ it is easier to make x' that is better than x than to make x' that is worse, but with the i-th bit flipped. This gives us much better chances to obtain a better individual in the crossover phase.

Looking for the ways of improving the algorithm we noticed that if we disregard selection after the mutation phase then every offspring at the crossover phase is generated via standard bit mutation

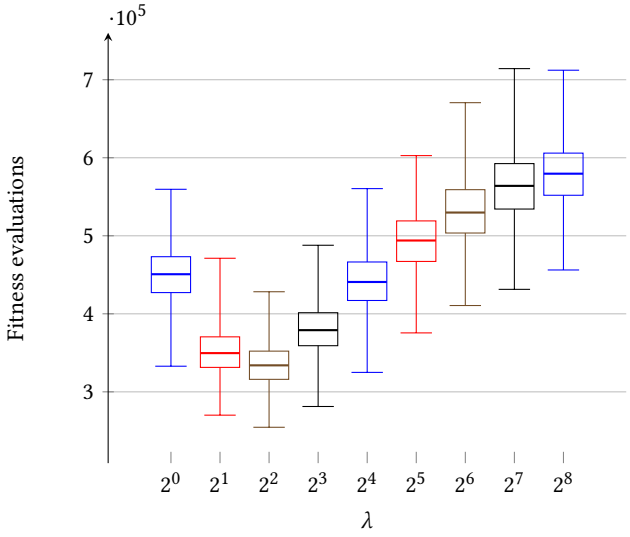

Figure 1: The distributions of the 1024 runtimes of the $(1 + (\lambda, \lambda))$ EA on LeadingOnes with $n = 512$ for various values of the parameter λ. With $\lambda = 1$ the $(1 + (\lambda, \lambda))$ EA is simply the $(1 + 1)$ EA.

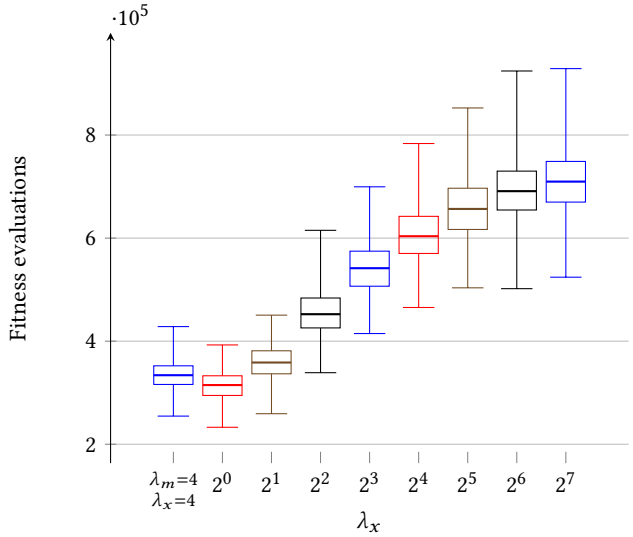

Figure 2: The distributions of the 1024 runtimes of the $(1 + (\lambda, \lambda))$ EA on LeadingOnes with $n = 512$ for various values of the parameters λ_x and $\lambda_m = 2\lambda_x$.

with mutation rate equals to $\frac{\lambda_m}{n\lambda_x}$, where λ_m and λ_x are values for the parameter λ at the mutation and crossover phases respectively. Since the optimal mutation rate was shown to be approximately $\frac{1.59}{n}$ in [1], we assumed that it could be profitable to use different values for the λ_m and λ_x. However, both λ_m and λ_x are integer values, so we define $\lambda_m = 2\lambda_x$, which gives us a mutation rate that is close to the optimal one. As one can see in Figure 2 for the smallest values of λ such approach outperforms ordinary $(1 + (\lambda, \lambda))$ EA with optimal value of $\lambda = 4$ by a small constant factor.

5 CONCLUSION

In this work we proved an upper bound of $O(n^2)$ fitness evaluations for the $(1 + (\lambda, \lambda))$ EA on the LeadingOnes problem. This result shows that the $(1 + (\lambda, \lambda))$ EA is asymptotically at least as efficient as the $(1 + 1)$ EA on this problem. Our empirical study has revealed that for relatively small values of λ the $(1 + (\lambda, \lambda))$ EA outperforms the $(1 + 1)$ EA by a small constant factor.

The most interesting way of the further theoretical work is to prove the matching lower bound, however so far it seems to be a challenging problem. The empirical research has a plenty of further directions, which includes a tuning of the different parameters of the $(1 + (\lambda, \lambda))$ EA, such as the mutation strength, the crossover biasness and the population sizes on each phase. Finding an optimal interplay between these parameters may become fruitful for the development of new crossover-based algorithms.

ACKNOWLEDGMENTS

Theoretical investigations were supported by the Paris Ile-de-France Region. Experimental investigations were supported by the Russian Science Foundation (Grant 17-71-20178).

REFERENCES

[1] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. 2010. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Parallel Problem Solving from Nature - PPSN XI, Proceedings, Part I.* 1–10.
[2] Dogan Corus and Pietro Simone Oliveto. 2018. Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 22, 5 (2018), 720–732.
[3] Benjamin Doerr and Carola Doerr. 2015. Optimal parameter choices through self-adjustment: applying the 1/5-th rule in discrete settings. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015.* 1335–1342.
[4] Benjamin Doerr, Carola Doerr, and Franziska Ebel. 2015. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science* 567 (2015), 87–104.
[5] Benjamin Doerr and Carola Winzen. 2014. Reducing the arity in unbiased black-box complexity. *Theoretical Computer Science* 545 (2014), 108–121.
[6] Simon Fischer and Ingo Wegener. 2004. The Ising model on the ring: mutation versus recombination. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2004.* 1113–1124.
[7] Brian W. Goldman and William F. Punch. 2014. Parameter-less population pyramid. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2014.* 785–792.
[8] Thomas Jansen and Ingo Wegener. 2002. The analysis of evolutionary algorithms - A proof that crossover really can help. *Algorithmica* 34, 1 (2002), 47–66.
[9] Thomas Jansen and Ingo Wegener. 2005. Real royal road functions–where crossover provably is essential. *Discrete Applied Mathematics* 149, 1-3 (2005), 111–125.
[10] Vladimir Mironovich and Maxim Buzdalov. 2015. Hard test generation for maximum flow algorithms with the fast crossover-based evolutionary algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2015.* 1229–1232.
[11] Tobias Storch and Ingo Wegener. 2004. Real royal road functions for constant population size. *Theoretical Computer Science* 320, 1 (2004), 123–134.
[12] Dirk Sudholt. 2005. Crossover is provably essential for the ising model on trees. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2005.* 1161–1167.
[13] Ingo Wegener and Carsten Witt. 2005. On the analysis of a simple evolutionary algorithm on quadratic pseudo-boolean functions. *Journal of Discrete Algorithms* 3, 1 (2005), 61 – 78.