The 1/5-th Rule with Rollbacks: On Self-Adjustment of the Population Size in the $(1 + (\lambda, \lambda))$ GA

Anton Bassin
ITMO University
Saint Petersburg, Russia
anton.bassin@gmail.com

Maxim Buzdalov
ITMO University
Saint Petersburg, Russia
mbuzdalov@gmail.com

ABSTRACT

Self-adjustment of parameters can significantly improve the performance of evolutionary algorithms. A notable example is the $(1 + (\lambda, \lambda))$ genetic algorithm, where the adaptation of the population size helps to achieve the linear runtime on the OneMax problem. However, on problems which interact badly with the self-adjustment procedure, its usage can lead to performance degradation compared to static parameter choices. In particular, the one fifth rule is able to raise the population size too fast on problems which are too far away from the perfect fitness-distance correlation.

We propose a modification of the one fifth rule in order to have less negative impact in scenarios when the original rule reduces the performance. Our modification, while still having a good performance on OneMax, both theoretically and in practice, also shows better results on linear functions with random weights and on random satisfiable MAX-SAT instances.

CCS CONCEPTS

• **Theory of computation** → **Theory of randomized search heuristics**; **Online learning algorithms**.

KEYWORDS

Parameter adaptation, $(1 + (\lambda, \lambda))$ GA, linear functions, MAX-SAT.

ACM Reference Format:
Anton Bassin and Maxim Buzdalov. 2019. The 1/5-th Rule with Rollbacks: On Self-Adjustment of the Population Size in the $(1 + (\lambda, \lambda))$ GA. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3322067

1 INTRODUCTION AND PRELIMINARIES

The $(1 + (\lambda, \lambda))$ genetic algorithm [2] is a bright example of a successful application of self-adjustment of parameters. Despite first successes on optimizing problems other than OneMax — on linear functions with random weights taken from [1; 2] and royal road functions [2], or a surprisingly good performance in practice on MAX-SAT problems [4], supported theoretically in [1] — this algorithm is quite slow to conquer other territories. A possible explanation is that the method of parameter adjustment can behave badly on problems with low fitness-distance correlation.

We investigate this problem by first performing a landscape analysis for a few problems. Then we propose a modification to the one fifth rule which slows its (dis)adaptation. We then conduct experiments on benchmark problems and confirm that the negative consequences of the one fifth rule's misguidance are damped.

The extended version of this paper[1] also proves that the runtime of the modified algorithm on OneMax is still linear. It also shows that, if the $(1 + (\lambda, \lambda))$ GA takes the best values for λ from Section 2, it outperforms other parameter choices on *all* tested problems.

Algorithm 1 $(1 + (\lambda, \lambda))$ GA, modified self-adjustment of $\lambda \le \bar{\lambda}$

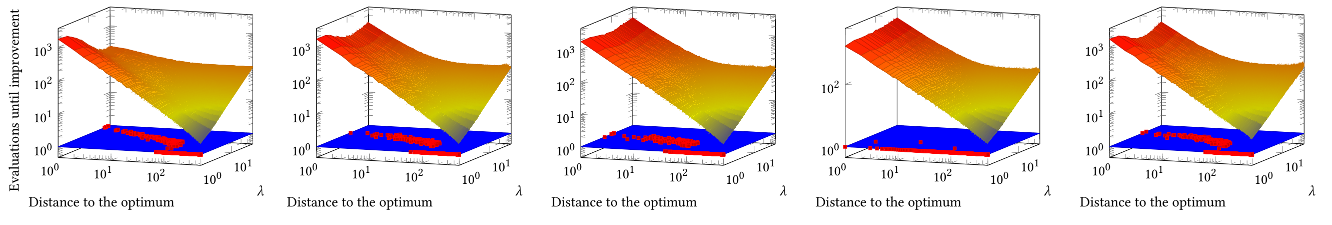
```
 1: $F \leftarrow \text{const} \in (1; 2)$, $U \leftarrow 5$  ▷ Update strength, the "1/5-th rule"
 2: $B \leftarrow 0$, $\Delta \leftarrow 10$, $\lambda_+ \leftarrow 1$  ▷ Bad loops, $\lambda$ growth span, base $\lambda$
 3: $x \leftarrow$ UniformRandom($\{0,1\}^n$)
 4: for $t \leftarrow 1, 2, 3, \dots$ do
 5:     $p \leftarrow \lambda/n$, $c \leftarrow 1/\lambda$, $\lambda' \leftarrow [\lambda]$, $\ell \sim \mathcal{B}(n, p)$
 6:     for $i \in [1..\lambda']$ do  ▷ Phase 1: Mutation
 7:         $x^{(i)} \leftarrow$ Mutate($x, \ell$)
 8:     $x' \leftarrow$ UniformRandom($\arg\max_{x^{(i)}} f$)
 9:     for $i \in [1..\lambda']$ do  ▷ Phase 2: Crossover
10:         $y^{(i)} \leftarrow$ Crossover($x, x', c$)
11:     $y \leftarrow$ UniformRandom($\arg\max_{y^{(i)}} f$)
12:     if $f(y) > f(x)$ then  ▷ Selection and Adaptation
13:         $x \leftarrow y$, $\lambda \leftarrow \max\{\lambda/F, 1\}$, $\lambda_0 \leftarrow \lambda$, $B \leftarrow 0$, $\Delta \leftarrow 10$
14:     else
15:         if $f(y) = f(x)$ then $x \leftarrow y$
16:         if $(B \leftarrow B + 1) = \Delta$ then $B \leftarrow 0$, $\Delta \leftarrow \Delta + 1$
17:         $\lambda \leftarrow \min\{\lambda_0 F^{B/(U-1)}, \bar{\lambda}\}$
```

2 ON EVALUATIONS UNTIL IMPROVEMENT

In Fig. 1a–1e we measured the impact of choosing particular values for λ, depending on the Hamming distance to the optimum, for problems OneMax, LinInt$_2$, LinInt$_5$, LinInt$_n$ and MAX-SAT [3] with logarithmic clause density. For a single problem size $n = 10^3$, and for all $\lambda_0 \in [1..50]$ we ran the $(1 + (\lambda, \lambda))$ GA with $\lambda = \lambda_0$, for 10^3 times for each λ_0 and for each problem. For all distances to the optimum $1 \le d \le 500$ we recorded the total number of evaluations $E(d, \lambda)$ spent in this location, and the total number of events $I(d, \lambda)$ that the algorithm find the better solution.

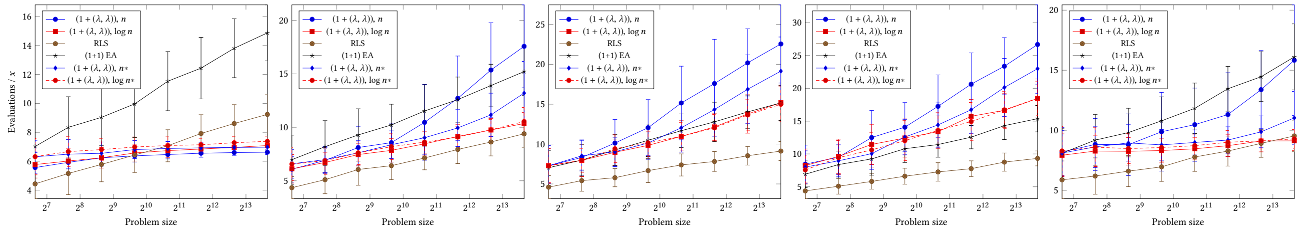
The top surfaces of the figures display an approximation of the *expected number of evaluations until improvement* $= E(d, \lambda)/I(d, \lambda)$.

[1]Available at http://arxiv.org/abs/1904.07284



(a) OneMax (b) LinInt$_2$ (c) LinInt$_5$ (d) LinInt$_n$ (e) MAX-SAT, $m = 4n \log n$

Figure 1: Performance plots for different functions, $n = 10^3$ everywhere, red cubes are choices for $\lambda(d)$ within 2% of the best



(a) OneMax (b) LinInt$_2$ (c) LinInt$_5$ (d) LinInt$_n$ (e) MAX-SAT, $m = 4n \log n$

Figure 2: Runtimes on different functions

Red cubes on the bottom surface show near-optimal λ values, which lie within 2% of the experimentally determined optimal choice.

The Fig. 1a generally confirms the fact that $\lambda \approx \sqrt{n/(n - f(x))}$ is a near-optimal choice for the $(1 + (\lambda, \lambda))$ GA on OneMax. For LinInt$_2$ and LinInt$_5$ the trend is probably still linear in logarithmic axes. The top surfaces might mean that the progress is too slow for the one fifth rule to keep λ in a good shape. The extreme LinInt$_n$ shows that the optimal values of λ are concentrated around $\lambda = 1$, and the entire landscape is too complicated for the GA. On the MAX-SAT problem the curve of optimal λ appears to be bent compared to OneMax. For large distances to the optimum it looks just like OneMax, but gets more complicated towards the optimum.

3 MODIFICATION AND EXPERIMENTS

The main idea of the proposed modification (Algorithm 1) to the self-adjustment rule of the $(1 + (\lambda, \lambda))$ GA is to prohibit the immediate growth of λ on long unsuccessful runs, while allowing raising it arbitrarily high in more steps if needed. It also retains the chances to perform iterations with rather small λ, which may be of use when small λ are better. This modification roughly squares the number of iterations, needed by the $(1+(\lambda, \lambda))$ GA to reach a certain distant value of λ. As a result, when the maximal λ overshoots the optimal value for the current distance, the algorithm still has some iterations to spend around the optimal values even if there is no fitness improvement yet, unlike the original $(1 + (\lambda, \lambda))$ GA.

We have evaluated the original $(1 + (\lambda, \lambda))$ GA with $\overline{\lambda} = n$ and $\overline{\lambda} = 2 \log n$, the same algorithm with the modified adaptation, the $(1 + 1)$ EA with the standard bit mutation and the randomized local search. The same five problems as in Section 2 are used in experiments. Each algorithm was run for 100 times on each problem with $n \in \{100, 200, 400, 800, 1600, 3200, 6400, 12800\}$.

The results are presented in Fig. 2a–2e. On OneMax, all variants of $(1 + (\lambda, \lambda))$ GA behave well. The proposed algorithm is slightly (about 10%) inferior to the original one, however, the dynamic is still linear. On LinInt$_2$ the logarithmically constrained versions behave not linearly, but better than the unconstrained versions. The modified unconstrained GA is below the $(1 + 1)$ EA, so at least the constant factor is smaller comparing to the original GA. With LinInt$_{5,n}$ the general trend of the unconstrained algorithms is to rise above the runtime of the $(1 + 1)$ EA, but the one with the modified self-adjustment strategy is always better. The slopes of these plots allows conjecturing that the runtime scales as $\Theta(n(\log n)^2)$. On the MAX-SAT problem the original unconstrained version seems to climb the plot at much higher rates than the modified one.

4 CONCLUSION

We proposed a modification of the one fifth rule in self-adjustment of the parameter λ for the $(1 + (\lambda, \lambda))$ GA. It is aimed at reducing the unwanted effects, resulting in the decreased performance on problems with low fitness-distance correlation. On OneMax the proposed strategy works by maybe 10% worse, the runtime is still linear in practice and theory. In cases pathological for the original self-adjustment scheme, we were able to see stable improvements over the classic $(1 + (\lambda, \lambda))$ GA on all problematic functions.

Acknowledgment: This research was supported by the Russian Scientific Foundation, agreement No. 17-71-20178.

REFERENCES

[1] Maxim Buzdalov and Benjamin Doerr. 2017. Runtime Analysis of the $(1 + (\lambda, \lambda))$ Genetic Algorithm on Random Satisfiable 3-CNF Formulas. In *Proceedings of Genetic and Evolutionary Computation Conference.* 1343–1350.
[2] Benjamin Doerr, Carola Doerr, and Franziska Ebel. 2015. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science* 567 (2015), 87–104.
[3] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA.
[4] B. W. Goldman and W. F. Punch. 2014. Parameter-less Population Pyramid. In *Proceedings of Genetic and Evolutionary Computation Conference.* 785–792.