Runtime Analysis via Symmetry Arguments (Hot-off-the-Press Track at GECCO 2021)

Benjamin Doerr
Laboratoire d'Informatique (LIX)
École Polytechnique, CNRS
Institut Polytechnique de Paris
Palaiseau, France

ABSTRACT

We use an elementary argument building on group actions to prove that the selection-free steady state genetic algorithm analyzed by Sutton and Witt (GECCO 2019) takes an expected number of $\Omega(2^n/\sqrt{n})$ iterations to find any particular target search point. This bound is valid for all population sizes μ. Our result improves and extends the previous lower bound of $\Omega(\exp(n^{\delta/2}))$ valid for population sizes $\mu = O(n^{1/2-\delta})$, $0 < \delta < 1/2$. This paper for the Hot-off-the-Press track at GECCO 2021 summarizes the work *Benjamin Doerr. Runtime Analysis of Evolutionary Algorithms via Symmetry Arguments. Information Processing Letters, 166:106064. 2021.* [5].

CCS CONCEPTS

• **Theory of computation → Theory and algorithms for application domains**; *Theory of randomized search heuristics.*

KEYWORDS

Runtime analysis, theory, group actions.

ACM Reference Format:
Benjamin Doerr. 2021. Runtime Analysis via Symmetry Arguments (Hot-off-the-Press Track at GECCO 2021). In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3462720

SUMMARY OF OUR RESULTS

The theory of evolutionary algorithms (EAs) has produced a decent number of mathematically proven runtime analyses. They explain the working principles of EAs, advise how to use these algorithms and how to choose their parameters, and have even led to the invention of new algorithms. We refer to [1, 7, 9, 10] for introductions to this area.

Due to the complexity of the probability space describing a run of many EAs, the majority of the runtime analyses regard very simple algorithms. In particular, there are only relatively few works discussing algorithms that employ crossover, that is, the generation of offspring from two parents. Among these, again very few present lower bounds on runtimes; we are aware of such results only in [8, 11, 13, 15].

In the most recent of these works, Sutton and Witt [15, Section 3] consider a simple crossover-based algorithm called StSt $\binom{\mu}{2}$ GA$_0$, a selection-free variant of a steady state genetic algorithm proposed earlier in [16]. It works with the search space $\Omega = \{0,1\}^n$ of bit strings of length n. The algorithm uses a population of size $\mu \geq 2$. Each iteration consists of (i) choosing two different individuals randomly from the population, (ii) applying a two-offspring uniform crossover, and (iii) replacing the two parents with the two offspring in the population.

For two parents x and y, the two offspring x' and y' are generated as follows. For all $i \in [1..n] := \{1,\dots,n\}$ with $x_i = y_i$, we have $x'_i = y'_i = x_i$ with probability one. For all $i \in [1..n]$ with $x_i \neq y_i$, we have $(x'_i, y'_i) = (1,0)$ and $(x'_i, y'_i) = (0,1)$ each with probability $1/2$.

The pseudocode of the StSt $\binom{\mu}{2}$ GA$_0$ is given in Algorithm 1. We did not specify a termination criterion since we are interested in how long the algorithm takes to find a particular solution when not stopped earlier. We also did not specify how to initialize the population since we will regard a very particular initialization later. We generally view populations as multisets, that is, an individual can be contained multiple times and this is reflected in the uniform random selection of individuals. Formally speaking, this means that a population is a μ-tuple of individuals and individuals should be referred to via their index in this tuple.

Algorithm 1: The StSt $\binom{\mu}{2}$ GA$_0$ with population size $\mu \geq 2$ operating on the search space $\{0,1\}^n$.

```
1 $t \leftarrow 0$;
2 Initialize $P_0$ with $\mu$ individuals from $\{0,1\}^n$;
3 for $t = 1, 2, \dots$ do
4     Choose from $P_{t-1}$ two random individuals $x$ and $y$
        without replacement;
5     $(x_t, y_t) \leftarrow$ crossover$(x, y)$;
6     $P_t \leftarrow P_{t-1} \setminus \{x, y\} \cup \{x_t, y_t\}$;
```

There is no fitness-based selection and no mutation in this simple process. Clearly, an algorithm of this kind is not expected to be very useful in practice. The reason to study such algorithms is rather that they allow to analyze in isolation how crossover works (more reasons to study this particular algorithm are described in [15]).

Without fitness-based selection, and thus without regarding the problem to be optimized, one would expect that this algorithm

This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH..

takes an exponential time to find any particular search point of the search space $\Omega = \{0,1\}^n$. Surprisingly, this is not so obvious, at least not when working with a particular initialization of the population.

Since a typical reason why crossover-based algorithms become inefficient is a low diversity in the population, Sutton and Witt consider an initialization of the StSt $\binom{\mu}{2}$ GA$_0$ which has "extremely high diversity", namely $\mu/2$ copies of the string $z = (1010\dots10)$ and $\mu/2$ copies of the string $z' = (0101\dots01)$. This population has the same number of zeros and ones in each bit position and has the maximal number of pairs of individuals with maximal Hamming distance n.[1] Still, this initialization is fair with respect to the target of generating the string $x^* = (11\dots1)$ in the sense that all initial individuals have from x^* a Hamming distance of $n/2$, which is the expected Hamming distance of a random string from x^* (and the expected Hamming distance of any string from a random target).

Sutton and Witt [15, Theorem 10] show that their algorithm with this initialization and with population size $\mu = O(n^{1/2-\delta})$, $\delta < 1/2$ a constant, takes an expected number of $\Omega(\exp(n^{\delta/2}))$ iterations to generate the target string $x^* = (11\dots1)$. Apparently, this lower bound is subexponential, namely at most of order $\exp(n^{1/4})$, for all population sizes. It becomes weaker with increasing population size and is trivial for $\mu = \Omega(\sqrt{n})$. We remark that this lower bound proof served as an example of an application of a more general result, which quantifies how the individuals in the population of the StSt $\binom{\mu}{2}$ GA$_0$ converge, in distribution, to individuals in which each bit is sampled independently and with a probability for a one equal to the rate of ones in this position in the initial population.

In this work, we do not touch this general result, but only regard the runtime of the StSt $\binom{\mu}{2}$ GA$_0$. By exploiting symmetries in the stochastic process, we improve the lower bound to $\Omega(2^n/\sqrt{n})$ for all values of μ.

Theorem. *Let $t, \mu, n \in \mathbb{N}$ with μ and n even. Consider a run of the StSt $\binom{\mu}{2}$ GA$_0$ initialized with $\mu/2$ copies of $z = (1010\dots10)$ and $\mu/2$ copies of $z' = (0101\dots01)$. Let T be the first iteration t in which the search point $x^* = (11\dots1)$ is generated.*

Then $\Pr[T \le t] \le 2t/\binom{n}{n/2}$. In particular, the expected time to generate x^* satisfies $E[T] \ge \frac{1}{4}\binom{n}{n/2} = \Omega(2^n/\sqrt{n})$.

Our proof is based on a simple group action or symmetry argument. We observe that the automorphisms of the hypercube $\{0,1\}^n$ (viewed as graph) commute with the operations of the StSt $\binom{\mu}{2}$ GA$_0$. Consequently, if an automorphism σ stabilizes the initial individuals z and z' (that is, $\sigma(z) = z$ and $\sigma(z') = z'$), then for any $x \in \{0,1\}^n$ at all times t the probability that the algorithm generates x equals the probability that it generates $\sigma(x)$.

From this symmetry, we conclude that if B is the set of all x such that there is an automorphism of the hypercube that stabilizes the initial individuals and such that $x = \sigma(x^*)$, then at all times the probability that x^* is generated, is at most $1/|B|$. We compute that B has exactly $\binom{n}{n/2}$ elements. Hence each search point generated by the StSt $\binom{\mu}{2}$ GA$_0$ is equal to x^* only with probability $\binom{n}{n/2}^{-1}$. A simple union bound over the $2t$ search points generated up to iteration t gives the result.

We believe that our lower bound is close to the truth, which we expect to be $\Theta(2^n)$, but we do not have a proof for this conjecture (in fact, we do not even know if the runtime is exponential – unfortunately, the few existing exponential upper bounds only regard mutation-based EAs, see [3]).

We note that when using a random initialization instead of the particular one proposed in [15], then a lower bound of $\Omega(2^n)$ follows simply from the fact that each search point that is generated is uniformly distributed. This argument, in a sense a toy version of ours, is apparently not widely known in the community; it was used in [4, Theorem 1.5.3] for a problem which previously [12, Theorem 5] was attacked with much deeper methods.

It thus seems that the difficulty of the problem posed in [15] not only stems from the use of crossover, but also from the fact that a non-random initialization was used. We note that so far the impact of different initializations has not been discussed intensively in the literature on runtime analysis of EAs. The only works we are aware of are [2, 6, 14].

In the light of this state of the art, the two **open problems** of improving the lower bound to $\Omega(2^n)$ and showing an exponential upper bound appear interesting. Any progress here might give us a broader understanding how to analyze EAs using crossover or non-random initializations.

REFERENCES

[1] Anne Auger and Benjamin Doerr (Eds.). 2011. *Theory of Randomized Search Heuristics.* World Scientific Publishing.

[2] Axel de Perthuis de Laillevault, Benjamin Doerr, and Carola Doerr. 2015. Money for nothing: speeding up evolutionary algorithms through better initialization. In *Genetic and Evolutionary Computation Conference, GECCO 2015.* ACM, 815–822.

[3] Benjamin Doerr. 2020. Exponential upper bounds for the runtime of randomized search heuristics. In *Parallel Problem Solving From Nature, PPSN 2020, Part II.* Springer, 619–633.

[4] Benjamin Doerr. 2020. Probabilistic tools for the analysis of randomized optimization heuristics. In *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, Benjamin Doerr and Frank Neumann (Eds.). Springer, 1–87. Also available at https://arxiv.org/abs/1801.06733.

[5] Benjamin Doerr. 2021. Runtime analysis of evolutionary algorithms via symmetry arguments. *Information Processing Letters* 166 (2021), 106064.

[6] Benjamin Doerr and Carola Doerr. 2016. The impact of random initialization on the runtime of randomized search heuristics. *Algorithmica* 75 (2016), 529–553.

[7] Benjamin Doerr and Frank Neumann (Eds.). 2020. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization.* Springer. Also available at https://cs.adelaide.edu.au/~frank/papers/TheoryBook2019-selfarchived.pdf.

[8] Benjamin Doerr and Madeleine Theile. 2009. Improved analysis methods for crossover-based algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2009.* ACM, 247–254.

[9] Thomas Jansen. 2013. *Analyzing Evolutionary Algorithms – The Computer Science Perspective.* Springer.

[10] Frank Neumann and Carsten Witt. 2010. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity.* Springer.

[11] Pietro S. Oliveto, Dirk Sudholt, and Carsten Witt. 2020. A tight lower bound on the expected runtime of standard steady state genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2020.* ACM, 1323–1331.

[12] Pietro S. Oliveto and Carsten Witt. 2011. Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica* 59 (2011), 369–386.

[13] Pietro S. Oliveto and Carsten Witt. 2015. Improved time complexity analysis of the simple genetic algorithm. *Theoretical Computer Science* 605 (2015), 21–41.

[14] Dirk Sudholt. 2013. A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 17 (2013), 418–435.

[15] Andrew M. Sutton and Carsten Witt. 2019. Lower bounds on the runtime of crossover-based algorithms via decoupling and family graphs. In *Genetic and Evolutionary Computation Conference, GECCO 2019.* ACM, 1515–1522.

[16] Carsten Witt. 2018. Domino convergence: why one should hill-climb on linear functions. In *Genetic and Evolutionary Computation Conference, GECCO 2018.* ACM, 1539–1546.

[1] We recall that the *Hamming distance* $H(x,y)$ of two bit strings $x, y \in \{0,1\}^n$ is defined by $H(x,y) = |\{i \in [1..n] \mid x_i \neq y_i\}|$.