Better Runtime Guarantees Via Stochastic Domination (Hot-off-the-Press Track at GECCO 2018)

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

ABSTRACT

Apart from few exceptions, the mathematical runtime analysis of evolutionary algorithms is mostly concerned with expected runtimes. In this work, we argue that stochastic domination is a notion that should be used more frequently in this area. Stochastic domination allows to formulate much more informative performance guarantees than the expectation alone, it allows to decouple the algorithm analysis into the true algorithmic part of detecting a domination statement and probability theoretic part of deriving the desired probabilistic guarantees from this statement, and it allows simpler and more natural proofs.

As particular results, we prove a fitness level theorem which shows that the runtime is dominated by a sum of independent geometric random variables, we prove tail bounds for several classic problems, and we give a short and natural proof for Witt's result that the runtime of any (μ, p) mutation-based algorithm on any function with unique optimum is subdominated by the runtime of a variant of the $(1 + 1)$ EA on the OneMax function.

This abstract for the Hot-off-the-Press track of GECCO 2018 summarizes work that has appeared in *Benjamin Doerr. Better runtime guarantees via stochastic domination. In Evolutionary Computation in Combinatorial Optimization (EvoCOP 2018), pages 1–17. Springer, 2018.*

CCS CONCEPTS

• **Theory of computation → Theory of randomized search heuristics**; *Optimization with randomized search heuristics*;

KEYWORDS

Run time analysis, theory of evolutionary computation.

SUMMARY OF OUR RESULTS

The analysis of evolutionary algorithms via mathematical means is an established part of evolutionary computation research. The subarea of *runtime analysis* aims at giving proven performance guarantees for the time an evolutionary algorithm takes to find optimal or near-optimal solutions. Traditionally, this area produces estimates for the expected runtime, which are occasionally augmented by tail bounds. A justification for this is that for already very simply evolutionary algorithms and optimization problems, the stochastic processes arising from running this algorithm on this problem are so complicated that any more detailed analysis is infeasible. See the analysis how the $(1 + 1)$ evolutionary algorithm optimizes linear functions $x \mapsto a_1x_1 + \cdots + a_nx_n$ in [9] for an example for this complexity.

In this work, we shall argue that the restriction to expectations is only partially justified and propose stochastic domination as an alternative. It is clear that the precise distribution of the runtime of an evolutionary algorithms in all cases apart from very trivial ones is out of reach. Finding the precise distribution is maybe not even an interesting target because most likely already the result will be too complicated to be useful. What would be very useful is a possibly not absolutely tight upper bound-type statement that concerns the whole distribution of the runtime.

One way to formalize such statement is via the notion of *stochastic domination*. We say that a real-valued random variable Y stochastically dominates another one X if and only if for each $\lambda \in \mathbb{R}$, we have $\Pr[X \leq \lambda] \geq \Pr[Y \leq \lambda]$. If X and Y describe the runtimes of two algorithms A and B, then this domination statement is a very strong way of saying that algorithm A is at least as fast as B. Clearly, stochastic domination implies the same order on the expectations, that is, here $E[X] \leq E[Y]$. Also, stochastic domination is invariant under monotonic rescaling. Hence if users have different (monotonic) utility functions for the runtime, then stochastic domination still tells them correctly which algorithm to prefer.

In [3] (extended version [4]), we give three main arguments for a more frequent use of domination arguments in runtime analysis, supported by a number of technical results of independent interest.

Stochastic domination is often easy to show. Surprisingly, despite being a much stronger type of assertion, stochastic domination statements are often easy to obtain. The reason is that many of the classic proofs implicitly contain all necessary information, they only fail to formulate the result as a domination statement.

As an example, we prove a natural *domination version of the classic fitness level method*. In analogy to the classic result, which translates pessimistic improvement probabilities $p_1, \ldots, p_{m-1}$ into the expected runtime estimate $E[T] \leq \sum_{i=1}^{m-1} \frac{1}{p_i}$, we show that under the same assumptions the runtime T is dominated by the sum of independent geometric random variables with success probabilities $p_1, \ldots, p_{m-1}$.

This statement implies the classic result, but also implies tail bound for the runtime via Chernoff bounds for geometric random variables. We note that, while our extension is very natural, the proof surprisingly is not totally obvious, which might explain why previous works were restricted to the expectation version.

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07.
https://doi.org/10.1145/3205651.3208209

Stochastic domination allows to separate the core algorithm analysis and the probability theoretic derivation of probabilistic runtime statements. The reason why stochastic domination statements are often easy to obtain is that they are close to the actions of the algorithm. When we are waiting for the algorithm to succeed in performing a particular action, then it is a geometric distribution that describes this waiting time. To obtain such a statement, we need a good understand of how the algorithm solves this particular problem, but usually no greater expertise in probability theory. We give in [3] several examples, mostly using the fitness level method, but also for the single-source shortest paths problem, where the fitness level method is not suitable to give the best known results.

Once we have a domination statement formulated, e.g., that the runtime is dominated by a sum of independent geometric distributions, then deeper probability theoretic arguments like Chernoff-type tail bounds come into play. This part of the analysis is independent of the algorithm and only relies on the domination statement. Exemplarily, we derive in [3] tail bounds for the runtime of the $(\mu + 1)$ EA on OneMax, the $(1 + 1)$ EA for sorting, and the multi-criteria $(1 + 1)$ EA for the single-source shortest path problem.

That these two stages of the analysis are of a different nature is also visible in the history of runtime analysis. As discussed in the previous paragraph, the classic fitness level method essentially contains all ingredients to formulate a runtime domination statement. However, the mathematical tools to analyze sums of independent geometric random variables were developed much later (and this development is still ongoing).

This historical note also shows that from the viewpoint of research organization, it would have been profitable if previous works would have been formulated in terms of domination statements. This might have spurred a faster development of suitable Chernoff bounds and, in any case, it would have made it easier to apply recently found Chernoff bounds like [5, 6, 10, 13].

Stochastic domination often leads to more natural and shorter proofs. To demonstrate this, we regard the classic lower-bound result which, in simple words, states that OneMax is the easiest fitness function for many mutation-based algorithms. This statement, of course, should be formulated via stochastic domination (and this has indeed been done in previous work). However, as we argue in [3], we also have the statement that when comparing the optimization of OneMax and some other function with unique optimum, then the distance of the current-best solution from the optimum for the general function dominates this distance for the OneMax function. This natural statement immediately implies the domination relation between the runtimes. We make this precise for the current-strongest OneMax-is-easiest result [12]. This will shorten the previous, two-and-a-half pages long complicated proof to an intuitive proof of less than a page.

Related work. The use of stochastic domination is not totally new in the theory of evolutionary computation, however, the results so far appear rather sporadic than systematic. Possibly the first to use this notion was Droste, who in [7, 8] employed it to make precise an argument of the type "this artificial process is not faster than the process describing a run of this algorithm on that problem." For an example of such an argument appearing at this conference see [2]. Domination arguments were extensively used in [11] to analyze the runtime of binary PSO algorithms. In [1], the notion of stochastic domination (without mentioning its name) was used to compare the performance of different evolutionary algorithms.

What comes closest to this work is the paper [14], which also tries to establish runtime analysis beyond expectations. The notion proposed in [14], called *probable computational time* $L(\delta)$, is the smallest time T such that the algorithm under investigation within the first T fitness evaluations finds an optimal solution with probability at least $1 - \delta$. To prove results on the probable computational time, the domination version of the fitness level method is used but not proven. At the time of writing of [14], good tail bounds for sums of geometric random variables with different success probabilities where not yet available (these appeared only in [5, 13]). For this reason, a weaker, self-made tail bound had to be used, which unfortunately gives a non-trivial tail probability only for values above twice the expectation. With this restriction, tail bounds are proven for the runtimes of the algorithms RLS, $(1 + 1)$ EA, $(\mu + 1)$ EA, MMAS* and binary PSO on the optimization problems OneMax and LeadingOnes.

Conclusion: This work shows that with stochastic domination, we can obtain performance guarantees for evolutionary algorithms that are much more insightful than bounds just on the expectation. Since often this additional strength comes at little extra cost, our recommendation is formulate future results first as domination results and only then deduce bounds on the expected runtime or bounds that hold with high probability.

REFERENCES

[1] Pavel A. Borisovsky and Anton V. Eremeev. 2008. Comparing evolutionary algorithms to the (1+1)-EA. *Theoretical Computer Science* 403 (2008), 33–41.

[2] 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 *Genetic and Evolutionary Computation Conference, GECCO 2018*. ACM.

[3] Benjamin Doerr. 2018. Better runtime guarantees via stochastic domination. In *Evolutionary Computation in Combinatorial Optimization, EvoCOP 2018*. Springer, 1–17.

[4] Benjamin Doerr. 2018. Better runtime guarantees via stochastic domination. *CoRR* abs/1801.04487 (2018). 41 pages.

[5] Benjamin Doerr and Carola Doerr. 2015. A tight runtime analysis of the (1+(λ, λ)) genetic algorithm on OneMax. In *Genetic and Evolutionary Computation Conference, GECCO 2015*. ACM, 1423–1430.

[6] Benjamin Doerr, Edda Happ, and Christian Klein. 2011. Tight analysis of the (1+1)-EA for the single source shortest path problem. *Evolutionary Computation* 19 (2011), 673–691.

[7] Stefan Droste. 2003. Analysis of the (1+1) EA for a dynamically bitwise changing OneMax. In *Genetic and Evolutionary Computation Conference, GECCO 2003*. 909–921.

[8] Stefan Droste. 2004. Analysis of the (1+1) EA for a noisy OneMax. In *Genetic and Evolutionary Computation Conference, GECCO 2004*. 1088–1099.

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

[10] Swante Janson. 2017. Tail bounds for sums of geometric and exponential variables. *ArXiv e-prints* arXiv:1709.08157 (2017). 8 pages.

[11] Dirk Sudholt and Carsten Witt. 2008. Runtime analysis of binary PSO. In *Genetic and Evolutionary Computation Conference, GECCO 2008*. ACM, 135–142.

[12] Carsten Witt. 2013. Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability & Computing* 22 (2013), 294–318.

[13] Carsten Witt. 2014. Fitness levels with tail bounds for the analysis of randomized search heuristics. *Information Processing Letters* 114 (2014), 38–41.

[14] Dong Zhou, Dan Luo, Ruqian Lu, and Zhangang Han. 2012. The use of tail inequalities on the probable computational time of randomized search heuristics. *Theoretical Computer Science* 436 (2012), 106–117.