Theoretical Computer Science 561 (2015) 3–23

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Optimizing linear functions with the $(1+\lambda)$ evolutionary algorithm—Different asymptotic runtimes for different instances ☆

Benjamin Doerr [a,b,c], Marvin Künnemann [b,d,*]

[a] Laboratoire d'Informatique (LIX), École Polytechnique, Palaiseau, France
[b] Max Planck Institute for Informatics, Saarbrücken, Germany
[c] Saarland University, Saarbrücken, Germany
[d] Saarbrücken Graduate School of Computer Science, Germany

ARTICLE INFO

Article history:
Received 1 November 2013
Accepted 7 March 2014
Available online 24 April 2014

Keywords:
Population-based EA
Runtime analysis

ABSTRACT

We analyze how the $(1+\lambda)$ evolutionary algorithm (EA) optimizes linear pseudo-Boolean functions. We prove that it finds the optimum of any linear function within an expected number of $O(\frac{1}{\lambda}n\log n+n)$ iterations. We also show that this bound is sharp for some linear functions, e.g., the binary value function. Since previous works shows an asymptotically smaller runtime for the special case of OneMax, it follows that for the $(1+\lambda)$ EA different linear functions may have run-times of different asymptotic order. The proof of our upper bound heavily relies on a number of classic and recent drift analysis methods. In particular, we show how to analyze a process displaying different types of drifts in different phases. Our work corrects a wrongfully claimed better asymptotic runtime in an earlier work [13]. We also use our methods to analyze the runtime of the $(1+\lambda)$ EA on the OneMax test function and obtain a new upper bound of $O(n\log\log\lambda/\log\lambda)$ for the case that λ is larger than $O(\log n\log\log n/\log\log\log n)$; this is the cut-off point where a linear speed-up ceases to exist. While our results are mostly spurred from a theory-driven interest, they also show that choosing the right size of the offspring population can be crucial. For both the binary value and the OneMax test function we observe that once a linear speed-up ceases to exist, in fact, the speed-up from a larger λ reduces to sub-logarithmic (still at the price of a linear increase of the cost of each generation).

1. Introduction

If there is one single problem that was most influential on the theory of evolutionary computation, then clearly it is the innocent question of how simple evolutionary algorithms optimize linear pseudo-Boolean functions, that is, functions $f:\{0,1\}^n\to\mathbb{R}; x\mapsto\sum_{i=1}^n w_i x_i$ with $w_1,\dots,w_n\in\mathbb{R}$. While easy to state, this problem is surprisingly difficult and already for the simple $(1+1)$ evolutionary algorithm (EA) was subject to a sequence of works [10,11,14–16,18,6,5,3,2,19,35,7,4,36]. More importantly, the quest for understanding this linear functions problem led to a large number of strong analysis tools like artificial fitness functions [11], additive drift analysis [14], average drift [18], multiplicative drift [6] and adaptive drift [3]

☆ A preliminary version of this work has been presented at the 15th Genetic and Evolutionary Computation Conference (GECCO 2013) [8].

* Correspondence to: Max Planck Institute for Informatics, Campus E1 4, 66123 Saarbrücken, Germany. Tel.: +49 681 9325 1027; fax: +49 681 9325 1099.
E-mail address: marvin@mpi-inf.mpg.de (M. Künnemann).

http://dx.doi.org/10.1016/j.tcs.2014.03.015
0304-3975/

(see Section 3 for an explanation of these terms). These techniques subsequently were heavily used and gave rise to a number of remarkable results in different areas, e.g., to the ICARIS 2011 best paper award winner [22] analyzing an immune inspired B-cell algorithm for the vertex cover problem.

In this work, we significantly extend this line of research and analyze how the $(1+\lambda)$ EA solves the linear functions problem. We prove a sharp upper bound on the runtime for arbitrary linear functions. This is the first time a sharp runtime analysis for a population-based EA for the linear functions problem is given (note that the only previous work in this direction is not correct, see the discussion below).

1.1. Optimizing linear functions

Rigorously analyzing how evolutionary algorithms solve optimization problems, and supporting this understanding with mathematical proofs, is one of the main goals in the theory of evolutionary computation. The hope is that a deeper understanding tells us which problems are tractable with evolutionary methods, what particular methods are best suited to solve a particular problem, how to choose the parameters of these methods in the right way, and what computational efforts to expect.

Unfortunately, even for simple evolutionary algorithms and very simple optimization problems such an understanding is surprisingly hard to obtain. For example, the innocent-looking question how the $(1+1)$ EA finds the minimum of a pseudo-Boolean linear function $f : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_{i=1}^n w_i x_i$ with positive coefficients $w_1, \dots, w_n$ kept the field busy for around 15 years. It seems obvious that any reasonable randomized search heuristic should easily find the unique optimum $x^* = (0, \dots, 0)$, since flipping any 1-bit into a 0-bit strictly improves the objective value. Unfortunately, if the w_i are not all equal, mutations may be accepted that increase the number of 1-bits (namely if the gain from flipping fewer 1-bits to 0-bits is larger than the loss from flipping more bits in the other direction). Consequently, it is easy to prove that *randomized local search (RLS)* (flipping only single bits) finds the optimum of any linear function in $n \ln n + O(n)$ iterations. However, the analogous bound of $en \ln n + O(n)$ for the $(1+1)$ EA was only proven last year in Witt's remarkable STACS paper [35], ending series of works [10,11,14–16,18,6,5,3,2,19,7,4,36] from various research groups over the last 15 years. See the extended version [36] of Witt's paper for a detailed account of the history of this problem.

One noteworthy consequence of Witt's result is that the $(1+1)$ EA finds *all* linear functions (with non-zero coefficients) equally easy to optimize (ignoring lower order terms). This follows from the lower bound [1] of $(1-o(1))en \ln n$ shown for the particular linear function of OneMax together with the general result [7] that OneMax has the shortest optimization time among all linear functions (when using the $(1+1)$ EA).

1.2. Population-based EA

While the linear functions problem for the $(1+1)$ EA is now well understood and, moreover, many tight or near-tight analyses for combinatorial optimization problems like minimum spanning trees or shortest paths exist (see the book [27]), our understanding of population-based EA, even simple ones like the $(\mu+1)$ EA or the $(1+\lambda)$ EA, is comparably weak.

What among the few runtime analyses on population-based EA comes closest to our work is the paper by Jansen, De Jong, and Wegener [21]. Among other results, they determine the asymptotic optimization times of the $(1+\lambda)$ EA on the classic test functions OneMax and LeadingOnes. While for the LeadingOnes function an optimization time of $\Theta(\frac{1}{\lambda}n^2 + n)$ generations is not too difficult to show, the optimization behavior of the simple linear function OneMax : $\{0,1\}^n \to \mathbb{R}$; $x \mapsto \sum_{i=1}^n x_i$ is already surprisingly rich. For all $\lambda = O(\log n \log\log n / \log\log\log n)$, only $O(\frac{1}{\lambda}n \log n)$ generations are needed. Hence for these λ, a *linear speed-up* is observed. For larger values of λ, a linear speed-up provably does not exist, that is, $\omega(\frac{1}{\lambda}n \log n)$ generations are necessary. Note that these results are far from trivial, in fact, their proofs span several pages in [21]. Similar analyses on how the $(\mu+1)$ EA solves the OneMax problem, also technically highly demanding, have been conducted by Storch [32] and Witt [34]; results on how the $(1,\lambda)$ EA optimizes OneMax were given by Jägersküpper and Storch [20] as well as Rowe and Sudholt [31].

There are some more results on runtimes of population-based EAs on artificial functions and a few scattered results on combinatorial optimization problems [28,26], but the overall impression remains that our understanding of population-based EAs is much inferior to the one of simpler randomized search heuristics.

1.3. Our result

In this work, we extend both the sequence of results on the $(1+1)$ EA optimizing linear functions as well as the analysis [21] of how the $(1+\lambda)$ EA optimizes the OneMax function.

Our technically most demanding result (Theorem 8) is that the $(1+\lambda)$ EA with arbitrary population size λ (which may and usually will be a function of the bit-string length n) finds the optimum of any linear function in $O(\frac{1}{\lambda}n \log(n) + n)$ generations. When $\lambda = O(n^{1-\varepsilon})$, we also show that this bound holds with probability $1 - n^{-s}$, s any constant, if the implicit constant in the runtime statement is made large enough.

These bounds are larger than some of the bounds in [21] for the OneMax function. This is with good reason. We show (Theorems 19 and 21) that our bound is sharp (apart from constant factors, but again with strong concentration) for the linear function BinVal : $\{0,1\}^n \to \mathbb{R}$; $x \mapsto \sum_{i=1}^n x_i 2^{i-1}$ and all $\lambda = O(n)$. Consequently, not all linear functions have the same