Theoretical Computer Science 545 (2014) 20–38

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm [☆]

Jonathan E. Rowe [a], Dirk Sudholt [b,*,1]

[a] School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK
[b] Department of Computer Science, University of Sheffield, Sheffield, S1 4DP, UK

ARTICLE INFO

Keywords:
Evolutionary algorithms
Comma strategies
Offspring populations
Runtime analysis
Drift analysis
Theory

ABSTRACT

We extend the theory of non-elitist evolutionary algorithms (EAs) by considering the offspring population size in the $(1, \lambda)$ EA. We establish a sharp threshold at $\lambda = \log_{\frac{e}{e-1}} n \approx 5 \log_{10} n$ between exponential and polynomial running times on OneMax. For any smaller value, the $(1, \lambda)$ EA needs exponential time on every function that has only one global optimum. We also consider arbitrary unimodal functions and show that the threshold can shift towards larger offspring population sizes. In particular, for the function LeadingOnes there is a sharp threshold at $\lambda = 2 \log_{\frac{e}{e-1}} n \approx 10 \log_{10} n$. Finally, we investigate the relationship between the offspring population size and arbitrary mutation rates on OneMax. We get sharp thresholds for λ that decrease with the mutation rate. This illustrates the balance between selection and mutation.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Runtime analysis has emerged as a very fruitful research area in the theory of evolutionary algorithms (EAs) and other metaheuristics. By rigorously estimating the expected running time of EAs on interesting classes of problems, we gain valuable insight into the working principles of EAs. Many running time bounds depend on parameters of the algorithm such as the mutation rate or the (offspring) population size. This allows for conclusions about optimal parameters or best possible design choices.

However, most runtime analyses presented so far have been focusing on plus strategies like $(\mu + \lambda)$ EAs, particularly special cases thereof like the $(1 + \lambda)$ EA [12,14], the $(\mu + 1)$ EA [34,6] or the $(1 + 1)$ EA [4]. In these algorithms the new population is chosen among both parents and offspring. On hard problems this bears the risk of getting stuck in local optima. One common approach for dealing with multimodal problems is to use comma strategies where the new population is selected solely among the offspring [5, p. 81].

Analysing non-elitist EAs like these is an interesting and challenging topic as, unlike for plus strategies, one cannot rely on the best fitness in the population to increase monotonically over time. Recent advances include runtime analyses of EAs with ranking selection [19], fitness-proportional selection [22,29], negative drift results [18] and Lehre's extension of the fitness-level method towards non-elitist EAs [17].

Jägersküpper and Storch [11] were among the first to investigate comma strategies like the $(1, \lambda)$ EA, see Algorithm 1. The $(1, \lambda)$ EA creates λ offspring independently and selects the best among these as parent. Offspring are created by flipping each bit in the parent independently with a given mutation rate u. The default choice is $u = 1/n$, where n is the number of bits.

☆ A preliminary conference version appeared at GECCO 2012 [31].
* Corresponding author.
[1] Part of this work was done while the author was at the University of Birmingham, supported by EPSRC grant EP/D052785/1.

0304-3975/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.tcs.2013.09.036

Algorithm 1 $(1, \lambda)$ EA with mutation rate u.

```
1: repeat
2:     Choose $x \in \{0,1\}^n$ uniformly at random.
3:     for each $1 \leqslant i \leqslant \lambda$ do
4:         Create $x^i$ by copying $x$ and flipping each bit in $x^i$ independently with prob. $u$.
5:     end for
6:     Let $x := \arg\max\{x^1, x^2, \dots, x^\lambda\}$.
7: until global optimum found.
```

The *running time* is defined as the number of generations needed to find a global optimum for the first time. In many studies the number of function evaluations is being considered as performance measure, as this often represents the most expensive operation. It is by a factor of λ larger than the number of generations, hence it suffices to consider the number of generations as running time.

The choice of the offspring population size λ is crucial with regard to the running time of the $(1, \lambda)$ EA. As shown in [11], the $(1, \lambda)$ EA needs exponential time on any function with a unique optimum if $\lambda \leqslant (\ln n)/14$, with high probability. The reason is that the number of offspring is so small that the algorithm tends to move away from the global optimum once it gets close. This "backward drift" leads to the exponential running time. Contrarily, if $\lambda \geqslant 3 \ln n$ the algorithm can effectively perform hill climbing and approach the global optimum on easy functions like OneMax. Moreover, there are examples of multimodal functions for which the $(1, \lambda)$ EA is provably more efficient than the $(1 + \lambda)$ EA [11].

Note that there is a gap by a large factor of 42 between the upper and lower bounds on λ. Neumann, Oliveto and Witt [22] narrowed this gap to 12 with regard to a similar algorithm working in a related setting which requires fitness values to be scaled appropriately. But this large gap still gives little guidance to practitioners as to how precisely λ should be set. We close this gap by showing that there is a sharp phase transition at $\lambda = \log_{\frac{e}{e-1}} n \approx 2.18 \ln n \approx 5 \log_{10} n$ for OneMax. In fact, any smaller value (by a constant factor $1 - \varepsilon$) leads to exponential running times on all functions with a unique optimum. Any larger value gives a polynomial expected running time for OneMax. For optimal λ the expected number of function evaluations is $O(n \log n)$. This means that then the $(1, \lambda)$ EA is able to perform simple hill climbing tasks efficiently. In a more general sense, this is a minimum requirement for an algorithm searching for good local optima. Experiments show that this threshold is indeed sharp, if the problem dimension is not too small.

On more complicated unimodal functions the phase transition is shifted to higher λ-values. For general unimodal functions with $d+1$ function values, we get polynomial running times in n and d if $\lambda \geqslant \log_{\frac{e}{e-1}}(dn) \approx 2.18 \ln(dn)$. For the function LeadingOnes the threshold for λ is twice as large as the threshold for OneMax, and we prove that it is tight in the same sense: any smaller value (by a constant factor $1 - \varepsilon$) leads to exponential running times on LeadingOnes.

The phase transition also depends on the mutation rate. When decreasing the mutation rate from its default value $1/n$, the threshold for the offspring population size decreases. We make this mutation–selection balance precise for the function OneMax and also show that only two offspring can be effective if the mutation rate is sufficiently small.

Empirical results accompany our theoretical thresholds to illustrate how, for various problem dimensions n, the average running time increases near phase transitions when λ is decreased.

Along the way, we also refine popular drift analysis tools. In particular, we investigate the balance between "forward drift" (a tendency to move towards the optimum) and "backward drift" (a tendency to move away from the optimum), in dependence of the offspring population size. This perspective is also useful for analysing parallel EAs: in island models with migration it might happen that in certain situations single islands exhibit a backward drift with regard to the evolution within that island. But migration between islands and the exchange of beneficial mutations that it entails can turn this backward drift into a forward drift.[2] We are confident that these tools will prove useful in further studies on non-elitist and parallel EAs.

The paper extends a preliminary version [31] and is structured as follows. After reviewing related work and clarifying some notation, we revisit and refine drift analysis results such as variable drift for proving upper bounds. We strengthen the negative drift theorem for exponential lower bounds to take account of large self-loop probabilities. In Section 4 we apply these results to determine the exact value for λ for which the runtime on OneMax moves from exponential to polynomial. We extend the polynomial-time result to unimodal functions, and provide a tight lower bound for LeadingOnes in Section 5. Finally, we consider arbitrary mutation rates for OneMax in Section 6.

1.1. Related work

Instead of using a fixed offspring population size throughout the run, researchers have considered different ways of adapting the offspring population size over time, according to the progress made by the algorithm [10,32]. We briefly mention work that has been underpinned by theoretical studies.

[2] In fact, the $(1, \lambda)$ EA can be seen as a special case of an island model, where each of λ islands evolves a single individual. In each generation all λ islands create one offspring that replaces its parent. Then all islands send a copy of their resident individual to all other islands. The best immigrant replaces the resident individual if it is no worse. This island model is equivalent to the $(1, \lambda)$ EA apart from the fact that the former can store different individuals of the same fitness.