Run-Time Analysis of Population-Based Evolutionary Algorithm in Noisy Environments

DOI: 
10.1145/2725494.2725498

Citation for published version (APA): 

Published in:

Citing this paper
Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights
Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy
If you believe that this document breaches copyright please refer to the University of Manchester’s Takedown Procedures [http://man.ac.uk/04Y6Bo] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.
Run-Time Analysis of Population-Based Evolutionary Algorithm in Noisy Environments

Adam Prügel-Bennett  
Electronics and Computer Science  
University of Southampton  
Southampton SO17 1BJ, UK  
apb@ecs.soton.ac.uk
Jonathan Rowe  
Department of Computer Science  
University of Birmingham  
Birmingham, UK  
J.E.Rowe@cs.bham.ac.uk
Jonathan Shapiro  
Department of Computer Science  
University of Manchester  
Manchester, UK  
jls@cs.man.ac.uk

ABSTRACT
This paper analyses a generational evolutionary algorithm using only selection and uniform crossover. With a probability arbitrarily close to one the evolutionary algorithm is shown to solve onemax in \(O(n \log^2(n))\) function evaluations using a population of size \(cn \log(n)\). We then show that this algorithm can solve onemax with noise variance \(n\) again in \(O(n \log^2(n))\) function evaluations.

Categories and Subject Descriptors
F.2 [Analysis Of Algorithms And Problem Complexity]; G.1.6 [Optimization]: Global optimization

General Terms  
Theory

Keywords  
Run-time analysis; uniform crossover; noisy optimisation.

1. INTRODUCTION
This paper is aimed at clarifying the mechanisms whereby a population-based evolutionary algorithm (EA) might provide a substantial advantage over local search. In particular we focus on the ability of crossover to focus search and a population to not be mislead by high levels of noise. The run-time analysis reveals the scaling behaviour of the mechanisms being analysed. The run-time analysis for the noise problem builds on an analysis of an evolutionary algorithm that uses uniform crossover and selection to solve onemax (that is, to find a binary string of all ones, where the only heuristic information is the number of ones in the string).

In the last few years there has been a significant push on obtaining run-time results for EAs. Mechanisms where crossover is necessary were first investigated by Jansen and Wegner who introduced a problem in which the performance gap between mutation-based and mutation-crossover based algorithms is superpolynomial [8], and a class of Royal Road functions in which the performance gap is exponential [7]. Subsequently, investigators have discovered other problems in which the addition of crossover improves performance significantly. For example, Sudholt [11] showed that a \((\mu + \lambda)\) EA with a constant population size needs exponential time to solve the Ising model on a complete binary tree, whereas a \((2+2)\) EA with crossover, mutation, and fitness-sharing has polynomial runtime. Doerr et. al. [2] showed for the all-pairs shortest path problem a polynomial speedup by using forms of crossover appropriate to the problem. Recently, Corus et al. [1] developed a technique which extends the level-based approach to single-point and uniform crossover and applied it to the onemax and leading ones problem.

We believe our paper makes two contributions. First, we analyse an algorithm with crossover and selection alone. We believe that this brings into focus the role of crossover in solving problems. Other analyses of crossover of which we are aware contain mutation as well. Of course, real practical EAs do contain mutation, so it is natural to include it in the algorithms studied. We introduce it as a theoretical tool to allow us to compare crossover as a search operator to mutation as a search operator. We know that one role of mutation is to maintain diversity in the population, so using crossover alone will likely required larger population sizes.

The second contribution is the addition of a further problem which crossover is necessary, in the strongest sense. We consider the problem of onemax plus noise of order \(\sqrt{n}\). We show that our selection plus crossover EA can solve the problem efficiently while a \((1+1)\) EA fails to make substantial progress towards the solution at all. This extends previous rigorous work on this problem to much higher amplitude noise. Droske [4] considered a \((1+1)\) EA to onemaxes in which one bit was flipped with probability \(p\) and the string was unchanged with probability \(1-p\), and showed that the run time was polynomial only if \(p = O(\ln n/n)\). Gießen and Kötzing generalise the form of noise and show that the introduction of populations can allow the EA can increase the size of noise that the EA can handle. We increase this much further.

The paper is organised as follows. In the next section we introduce the techniques that we use and define the EA that we use through out the paper. In section 3, we derive run-times bound for solving onemax. Section 4 considers the problem of coping with noise on the fitness evaluation. We
consider the problem of solving onemax plus noise of order \(\sqrt{n}\). We conclude in section 5.

2. BACKGROUND MATERIAL

In this section we briefly recap on the main proof tools that we use throughout the paper and then describe the evolutionary algorithm that we use.

2.1 Proof Tools

The main proof tools we will use are the drift bounds. We use the multiplicative drift theorem which provides an upper bound on the expected time to achieve some goal in terms of the expected progress towards the goal. For completeness we state the theorem without a proof.

**Theorem 1.** Let \((X_t)_{t \geq 0}\) be a series of positive random variables describing a Markov chain. Define \(T\) to be the first time to reach a state with value less than \(a > 0\)

\[ T = \min\{t \geq 0 : X_t < a\}. \]

If there exists a \(\delta > 0\) such that at any step \(t \geq 0\), and at any state \(X_t\) with \(X_t > a\)

\[ \mathbb{E}[X_{t+1} - X_t | X_t] \geq \delta X_t \]

then the expected time, \(T\), for \(X_t\) to reach a value less than \(a\) is bounded by

\[ \mathbb{E}[T] \leq \frac{1 + \log(\mathbb{E}[X_0] / a)}{\delta}. \]

The theorem is proved in [3]. In fact we give a slightly modified version of the theorem in that we consider the expected time to reach a value less than (rather than equal to) some positive constant \(a\) and we take the expected initial position, but both extensions are trivial.

We also require a negative drift theorem. We could use the standard theorem of [9] [10], however, we can obtain a tighter bound using the following theorem.

**Theorem 2.** Let \((X_t)_{t \geq 0}\) be a series of random variables describing a Markov chain. We consider the points \(a < b < c\) with \(b - a = c - b = \ell\).

\[ \ell \]

\[ a \quad b \quad c \]

If \(X_0 \geq b\) and the following two conditions apply

1. For \(X_{t-1} > c\) there exists a constant \(\varepsilon > 0\) such that

\[ \mathbb{P}(X_t \leq b | X_{t-1} = c) \leq e^{-\varepsilon t} \]

2. For \(a \leq X_{t-1} \leq c\) there exists constants \(\Delta\) and \(\nu > 0\) such that

\[ \log\left(\mathbb{E}[e^{-\lambda(X_t - X_{t-1})} | X_{t-1}]\right) \leq -\lambda \Delta + \frac{\nu \lambda^2}{2} \]

then

\[ \mathbb{P}(X_T \leq a) \leq T \max(e^{-\varepsilon t}, e^{-2\ell \Delta / \nu}). \]

Proof. The first condition states that the probability of making a jump from \(X_{t-1} > c\) to \(X_t < b\) is exponentially unlikely. Assuming no such jump happens then for \(X_{t-1} \leq a\) for some \(\tau\) the system must have made a series of steps across the interval \([a, b]\) where at each step the second condition holds. We show that such a series of steps is exponentially unlikely. Using the usual Chernoff-bound construction we have for \(\lambda > 0\)

\[ \mathbb{P}(X_T - X_0 \leq -\ell) = \mathbb{P}(e^{\lambda(X_T - X_0)} \geq e^{\lambda \ell}) \leq \mathbb{E}[e^{\lambda(X_T - X_0)}] \leq e^{\lambda \ell} \]

using Markov’s inequality. This is true for any \(\lambda > 0\), thus

\[ \mathbb{P}(X_T - X_0 \leq -\ell) \leq e^{-\psi(\ell)}, \]

\[ \psi(\ell) = \max_{\lambda > 0} \lambda \ell - \log\left(\mathbb{E}[e^{\lambda(X_T - X_0)}]\right). \]

However

\[ \mathbb{E}[e^{\lambda(X_T - X_0)}] = \mathbb{E}[\mathbb{E}[e^{\lambda(X_T - X_{t-1})} | X_{t-1}]] e^{\lambda(X_{t-1} - X_0)} \]

but, by the assumption, \(a \leq X_{t-1} \leq b\) and from condition 2

\[ \mathbb{E}[e^{\lambda(X_T - X_0)}] \leq e^{-\lambda \Delta + \frac{\nu \lambda^2}{2}} \mathbb{E}[e^{\lambda(X_{t-1} - X_0)}]. \]

Iterating we have

\[ \log\left(\mathbb{E}[e^{\lambda(X_T - X_0)}]\right) \leq -\lambda \Delta t + \frac{\nu \lambda^2}{2} \]

where \(t + 1\) is the length of time since \(X_t > b\).

\[ \psi(\ell) \geq \max_{\lambda > 0} \lambda (\ell + \Delta t) - \frac{\nu \lambda^2}{2} = \left(\frac{\ell + \Delta t}{2}\right) \]

giving

\[ \mathbb{P}(X_T \leq a) \leq e^{-(\ell + \Delta t)^2 / (2 \nu t)}. \]

The right-hand side reaches its maximum when \(t = \ell / \Delta\) so that

\[ \mathbb{P}(X_T \leq a) \leq e^{-2\ell \Delta / \nu} \]

assuming that \(a \leq X_t \leq b\) for \(\tau < t\). The system may have reached the \(X_t = c\) (possibly many time) and either made a very large jump or have jumped back into the interval \([b, c]\) and subsequently walked the interval \([a, b]\). Taking a union bound on the most likely way to reach a state with \(X_t \leq a\) for all time steps \(T\) we obtained the result given in the theorem.

The basic tool we use for obtaining concentration results are bounds for the cumulant generating function of a random variable \(X - \mathbb{E}[X]\), defined by

\[ \tilde{G}_X(\lambda) = \log\left(\mathbb{E}[e^{\lambda(X - \mathbb{E}[X])}]\right). \]

We use the following well known inequality associated with Bernstein and Chernoff.

**Theorem 3.** For a random variable \(X\) if \(\tilde{G}_X(\lambda) \leq c \lambda^2\) then

\[ \mathbb{P}(|X - \mathbb{E}[X]| \geq \varepsilon) \leq e^{-\varepsilon^2/(4c)}. \]
A useful result due to Hoeffding [6] is

**Theorem 4.** For a bounded random variable \( a \leq X \leq b \)
then

\[
\tilde{G}_{X}(\lambda) \leq \frac{(b-a)^2}{8} - \lambda^2.
\]

or

\[
\log\left( \mathbb{E}\left[ e^{\lambda X} \right] \right) \leq \lambda \mathbb{E}[X] + \frac{(b-a)^2}{8} - \lambda^2.
\]

Using theorems 3 and 4 on the sum, \( S_n \), of \( n \) independent random variables bounded in an interval from \( a \) to \( b \) we obtain Hoeffding’s inequality

\[
\mathbb{P}(S_n - \mathbb{E}[S_n] \geq t) \leq e^{-\frac{2t^2}{n(b-a)^2}}.
\]

Finally we need the following identity for the binomial sum

\[
S_n = \frac{1}{2^n} \sum_{i=0}^{n} \binom{n}{i} - \frac{n}{2} \left( \frac{n}{i} \right) = \frac{1}{2^n} \left[ \binom{n}{i} \left( \frac{n}{i} \right) \right] \geq \sqrt{n} \frac{1}{8} \quad (2)
\]

To prove the equality we note

\[
2^n S_n = \frac{1}{2} \sum_{i=0}^{[n/2]} \binom{n}{i} - \frac{n}{2} - \sum_{i=0}^{[n/2]-1} (n-i) \binom{n}{i} = \frac{1}{2} \sum_{i=0}^{[n/2]} \binom{n}{i} - \frac{n}{2i} \binom{n}{i} - \sum_{i=0}^{[n/2]-1} \binom{n}{i} \frac{n}{i} = \frac{1}{2} \sum_{i=0}^{[n/2]} \frac{n-n/2}{i} \binom{n}{i}.
\]

where equality 1 follows from the symmetry of the binomial, \( \binom{n}{i} = \binom{n}{n-i} \); equality 2 splits the coefficient and throws out the \( i = 0 \) term in the second sum which vanishes, equality 3 follows from the binomial identities

\[
\binom{n}{i} = \frac{n}{i} \binom{n-1}{i-1}, \quad i \binom{n}{i} = \frac{n}{i} \binom{n-1}{i-1}.
\]

equality 4 follows from cancelling terms in the two sum, finally equality 5 uses \( \binom{n}{n-1} = \binom{n}{i} \). The inequality for \( S_n \) follows from using Stirling’s bound for the factorial.

### 2.2 Paired-Crossover EA

The results presented in this paper are for an algorithm we call the *paired-crossover evolutionary algorithm* or PCEA. The algorithm consists of a population of \( P \) strings. It generates a new population of \( P \) strings by taking pairs of strings, \( X^\alpha \) and \( X^\beta \), and applying uniform crossover to generate two complementary children, \( X^\alpha_c \) and \( X^\beta_c \), such that the variables at the \( i \)th site of the children are

\[
X_i^{\alpha_c} = a_i X_i^{\alpha} + (1-a_i) X_i^{\beta}, \quad X_i^{\beta_c} = (1-a_i) X_i^{\alpha} + a_i X_i^{\beta}
\]

where \( a_i \in \{0,1\} \) are random independent variables with a probability of a half of being in either state. Having generated two complementary children, it chooses the fitter of the two, or chooses either with equal probability if they have the same fitness and puts it into the new population. This is repeated \( P \) times to generate the new population. Each pair of strings is selected independently from the parent population.

The initial population consists of strings drawn uniformly at random. Note that our algorithm has no mutation, this is because we wish to investigate crossover as a search operator. Although this means that the search can fail, it prevents the need either for elitism or to anneal the mutation rate to stop the population from reaching a mutation selection balance away from the optimum.

### 3. ONEMAX

We apply the PCEA to the onemax problem. We consider strings \( X = (X_1, X_2, \ldots, X_n) \), where each element of the string is binary (i.e. \( X_i \in \{0,1\} \)). We denote the number of ones in a string \( X \) by \( \lVert X \rVert \). The fitness of the string for onemax is equal to the number of ones in the string

\[
F_{\text{onemax}}(X) = \lVert X \rVert = \sum_{i=1}^{n} X_i.
\]

The only heuristic information we are given is the fitness of the strings. The optimum is the all ones string.

In uniform crossover, for a child \( \mu \) produced from parents \( \alpha \) and \( \beta \),

\[
\lVert X_\mu \rVert = \sum_{i=1}^{n} X_\mu^i = \frac{1}{2} \sum_{i=1}^{n} (X_\alpha^i + X_\beta^i + (2a_i - 1)(X_\alpha^i - X_\beta^i)) = \frac{\lVert X_\alpha \rVert + \lVert X_\beta \rVert}{2} + \eta^{\alpha\beta}
\]

where

\[
\eta^{\alpha\beta} = \frac{1}{2} \sum_{i=1}^{n} (2a_i - 1)(X_\alpha^i - X_\beta^i).
\]

Since \( (2a_i - 1) \) is equal to \( \pm 1 \) with equal probability and \( X_\alpha^i - X_\beta^i \) is equal to \( \pm 1 \) at the sites where the two parents differ, and 0 otherwise,

\[
\mathbb{P}(\eta^{\alpha\beta} = k) = 2^{d^{\alpha\beta}/2} \left( \frac{d^{\alpha\beta}}{d^{\alpha\beta}/2} \right)
\]

where \( d^{\alpha\beta} \) is the Hamming distance between the parents two (i.e. the number of sites where the parents differ). In the PCEA where we perform complementary crossover, the number of ones in the children will be

\[
\lVert X_\alpha^{\mu} \rVert = \frac{1}{2} \left( \lVert X_\alpha \rVert + \lVert X_\beta \rVert \right) + \eta^{\alpha\beta}
\]

\[
\lVert X_\beta^{\mu} \rVert = \frac{1}{2} \left( \lVert X_\alpha \rVert + \lVert X_\beta \rVert \right) - \eta^{\alpha\beta}.
\]

For the onemax problem we choose the child with the largest number of ones. Thus the number of ones for the selected child is

\[
\frac{1}{2} \left( \lVert X_\alpha \rVert + \lVert X_\beta \rVert \right) + |\eta^{\alpha\beta}|
\]

Applying equation (2), we find the expected increase in the number of ones is

\[
\mathbb{E}(|\eta^{\alpha\beta}|) = 2^{-d^{\alpha\beta}/2} \left( \frac{d^{\alpha\beta}}{d^{\alpha\beta}/2} \right) \geq \sqrt{\frac{d^{\alpha\beta}}{8}}.
\]
We denote the average number of zeros per string by
\[
\bar{z} = n - \frac{1}{P} \sum_{\alpha=1}^{P} \|X^\alpha\| = \frac{1}{P} \sum_{\alpha=1}^{P} \sum_{i=1}^{n} (1 - X_i^\alpha).
\]

Since in the PCEA we use a crossover where every member of the population is used twice to create the next population, the expected number of zeros in the population at the next generation is
\[
E[\bar{z}'] = \bar{z} - E[\eta^{\alpha\beta}]
\]
where the expectation is with respect to all possible pairings of the string and all possible values of \(\eta^{\alpha\beta}\). Using equation (4), the expected number of zeros at the next generation is thus
\[
E[\bar{z}'] \leq \bar{z} - \sqrt{\frac{d^{\alpha\beta}}{8}}
\]
where the expectation is over Hamming distances produced by all pairings.

We see that there is a drift towards reducing the number of zeros in the population. However, for an EA using only crossover and selection it is possible to reach a state where, at a particular site, the allele values of all the strings equals 0. This is known as fixation as it is no longer possible to obtain a 1 at this site. When fixation occurs the EA is no longer able to solve onemax. We will show that, for a population of size \(P = c \sqrt{n} \log(n)\) we can always choose a \(c\) such that the probability of fixation to all 0’s occurs with arbitrarily small probability.

To establish this and to obtain a run time bound for solv- ing onemax we consider the evolution of the number of ones in the population at each site
\[
M_i = \sum_{\alpha=1}^{P} X_i^\alpha.
\]

Crossover can yield a 1 at site \(i\), either if both parents had 1 at site \(i\) or one parent had a 1 at site \(i\) and the other had a 0. The probability of choosing two different parents who both have a 1 at site \(i\) is \(M_i (M_i - 1)/(P (P - 1))\). The probability that the two parents have different allele values at site \(i\) is \(2M_i (P - M_i)/(P (P - 1))\). In this case, there is a slight bias towards choosing the child with the 1 at site \(i\) over that with 0, because, in expectation over all crossovers, it will be fitter than the complementary child. As we have seen in crossing two strings \(X^\alpha\) and \(X^\beta\) the expected increase in the number of ones of the selected child is at least \(\sqrt{d^{\alpha\beta}}/8\). As we are using an unbiased uniform crossover the probability of obtaining a 1 at each site where the strings differ is at least
\[
\frac{1}{2} + \frac{1}{\sqrt{d^{\alpha\beta}}} \geq \frac{1}{2} + \frac{1}{\sqrt{8n}}.
\]

It follows that the expected number of ones at site \(i\) is bounded by
\[
E[M_i] \geq P \left( \frac{M_i (M_i - 1)}{P (P - 1)} + \frac{2M_i (P - M_i)}{P (P - 1)} \left( \frac{1}{2} + \frac{b}{\sqrt{n}} \right) \right)
\]
\[
= M_i + 2M_i \left( \frac{P - M_i}{P - 1} \right) \frac{b}{\sqrt{n}}
\]
(6)

where \(b = 1/\sqrt{8}\). We use this result both to prove that we never get near to fixation at any site and to obtain a run-time bound on solving onemax. Later on we consider modified problems where we obtain a similar equation as equation (6), but with different numerical values of \(b\). In the following theorems we therefore take \(b\) to be an arbitrary constant.

The next theorem shows that with high probability we do not get near to fixation during the run time of our algorithm. We consider this for an arbitrary site. Using a union bound we can then show that the probability of fixation at any site can be arbitrarily small by choosing a sufficiently large population.

**Theorem 5.** For the PCEA, with a population of size \(P = c \sqrt{n} \log(n)\), the probability that the number of ones, \(M_i(t)\), in the population at site \(i\) and time \(t\) is bounded by
\[
\mathbb{P}(M_i(t) < \frac{P}{4}) \leq t n^{-c b/8},
\]
where for onemax \(b = 1/\sqrt{8}\). For sufficiently large \(c\) this can be made arbitrarily small.

**Proof.** This theorem follows from theorem 2 applied to the \(M_i(t)\), the number of ones in the population at site \(i\) at generation \(t\). Consider the random variable
\[
M_i(t) - M_i(t-1) = \sum_{\mu=1}^{P} (X_i^\mu(t) - X_i^\mu(t-1))
\]
where \(X_i^\mu(t) - X_i^\mu(t-1) \in \{-1, 0, 1\}\). As this is a sum of variables that differ by at most 2 and are independent we can apply Hoeffding’s inequality. Note that each child is independent of any other child since the parents are chosen independently and the crossover between each pair of parents is independent (of course, the probability of \(X_i^\mu(t)\) and \(X_i^\mu(t-1)\) are not independent, but we use a union bound to show that fixation does not occur at any of the sites). Using Hoeffding’s inequality for the cumulant generating function (theorem 4)
\[
\log(\mathbb{E}[e^{-\lambda M_i(t)}]) \leq -\mathbb{E}[\Delta M_i(t)] \lambda + \frac{P \lambda^2}{2}
\]
where \(\Delta M_i(t) = M_i(t) - M_i(t-1)\). But, from equation (6), for \(P/4 \leq M_i(t-1) \leq 3P/4\)
\[
\mathbb{E}[M_i(t)] - \mathbb{E}[M_i(t-1)] > \frac{P b}{4 \sqrt{n}}
\]
thus
\[
\log(\mathbb{E}[e^{-\lambda \Delta M_i(t)}]) \leq -\frac{P b \lambda}{4 \sqrt{n}} + \frac{P \lambda^2}{2}.
\]

This satisfies condition 2 of theorem 2 with \(\Delta = P b/4 \sqrt{n}\), \(\sigma^2 = P\) and \(\ell = P/4\). To obtain a bound on the probability of jumping to \(M_i(t) \leq P/2\) given \(M_i(t-1) > 3P/4\), we can
ignore any drift and using a simple Hoeffding’s inequality, equation (1),
\[ P \left( M_i(t) \leq \frac{P}{2} \right) \leq e^{-\frac{P^2}{4}} \leq e^{-\frac{3P^2}{4}} = e^{-P^2/64}. \]

Thus, condition 1 of theorem 2 is also satisfied so we have the bound
\[ P \left( M(t) \leq \frac{P}{4} \right) \leq t e^{-P b/(8\sqrt{n})}. \]
Choosing \( P = c \sqrt{n} \log(n) \) then
\[ P \left( M(t) \leq \frac{P}{4} \right) \leq t n^{-c/8}. \]

\[ \square \]

The theorems above show that provided we make the population sufficiently large (through choosing the constant \( c \)) then we can make the probability of fixation at a single site arbitrarily small. By using a union bound we can also choose a constant \( c \) so that fixation at any site is arbitrarily small. Thus, in all but a very small number of runs we can assume that \( M_i \geq P/4 \), through out the run. Using this result we can now establish our main run-time theorem.

**Theorem 6.** The PCEA with a population of size \( P = c \sqrt{n} \log(n) \) will solve onemax in \( O(\sqrt{n} \log(n)) \) generation in expectation, in a proportion of, at least, \( 1 - n^{-O(\alpha)} \) of its runs.

**Proof.** We can use equation (6) together with the multiplicative drift to obtain a bound on the run time to solve onemax. Using the result of theorem 5, we can assume \( M_i \geq P/4 \), substituting this into equation (6) we find
\[ E[M_i] \geq M_i + \frac{b P}{2 \sqrt{n}} \left( \frac{P - M_i}{P - 1} \right). \]
To use the standard drift theorem, we consider the total number of zeros in the whole population
\[ Z = n P - \sum_{i=1}^{n} M_i = \sum_{i=1}^{n} \sum_{a=1}^{P} \left( 1 - X_i^a \right). \]
Summing the inequality for the expected number of ones over all sites and subtracting from \( n P \) we find
\[ E[Z] \leq Z - \frac{b P}{2 \sqrt{n}} \left( \frac{Z}{P - 1} \right) \leq Z \left( 1 - \frac{b}{2 \sqrt{n}} \right). \]
This shows we have a multiplicative drift towards states with a smaller number of zeros. We are guaranteed that the first member of the population will have found the optimal solution when \( Z < P \). Using the multiplicative drift theorem (theorem 1) with \( \delta = b/(2 \sqrt{n}) \) then the expected time for \( Z < P \) is
\[ E[T] \leq \frac{2 \sqrt{n}}{b} \left( 1 + \log(n) \right). \]
For onemax, \( b = 1/\sqrt{5} \) and \( E[T] < 12 \sqrt{n}(1 + \log(n)) \).

By theorem 5, the probability of failure due to fixation is less than \( t n^{-c/8} \). \( \square \)

### 4. COPING WITH NOISE

Clearly solving onemax using this algorithm is of little practical interest. However, we can, with small modifications, obtain much more interesting results where we can show an evolutionary algorithm using crossover can substantially out-perform many common search algorithms that do not use crossover. We illustrate this by considering the problem of solving a problem with very large measurement noise.

#### 4.1 PCEA Solving Onemax with Noise

We consider a onemax problem on binary strings of length \( n \), but where we are returned an approximate fitness for string \( X \) of
\[ F_{\text{on+noise}}(X) = \|X\| + \sqrt{n} \eta \]
where \( \eta \) is a normally distributed random variable with zero mean and unit variance (i.e. \( \eta \sim \mathcal{N}(0,1) \)). The noise is assumed to be independent every time the fitness of the string is evaluated. We summarise our run-time analysis in the following theorem.

**Theorem 7.** For the PCEA with a population of \( P = c \sqrt{n} \log(n) \) the expected time to solve the onemax with noise problem is \( O(\sqrt{n} \log(n)) \) in all but a proportion of no more than \( n^{-O(\alpha)} \) of the runs.

**Proof.** The proof follows very closely the proof for one-max (theorem 6). We establish that there is a positive bias towards increasing the number of ones at each site. The only difference is the constant in the drift term. Everything else follows the proof of theorem 6.

As with the normal onemax problem we denote the number of ones at an arbitrary site \( i \) by \( M_i \). A child will have a 1 at site \( i \) if either: (a) its two parents have a 1 at this site, which happens with a probability \( M_i(M_i - 1)/(P(P - 1)) \); or (b) one of its parent has a 1 and the other has a 0 at this site and the child with the 1 is selected. Denoting the child with the extra 1 by \( X^\mu \) and its complement by \( X^-\mu \) then the child with the extra one is selected if
\[ \sum_{j=1}^{n} (X^\mu_j - X^-\mu_j) + \sqrt{n} \eta > -1 \]
where \( \eta \) is the noise added to child \( X^\mu \) and \( \eta \) is the noise added to the complementary child \( X^-\mu \). Child \( X\mu \) will be selected in preference to child \( X^-\mu \) if \( 1 + S > 0 \), where \( S \) is the term on the left-hand side of equation (7). We note that if \( X^\mu \) and \( X^-\mu \) differ at site \( j \) then \( X^\mu_j - X^-\mu_j = \pm 1 \) with equal probability. Furthermore, if we consider all possible crossovers then the value of \( X^\mu_j - X^-\mu_j \) are independent of each other. To calculate the probability density of \( S \), it is useful to consider the characteristic function (i.e. the Fourier transform of the probability density function) defined by
\[ \tilde{f}_S(\omega) = \int_{-\infty}^{\infty} f_S(S) e^{i\omega S} dS. \]

The characteristic function for a sum of random independent variables is just equal to the product of the characteristic function for the independent variables. Defining \( D_j = X^\mu_j - X^-\mu_j \), then assuming \( X^\mu \) and \( X^-\mu \) differ at site \( j \),
the characteristic function for $D_j$ is

$$P_{D_j}(\omega) = \sum_{D_j \in \{-1,1\}} \frac{1}{2} \delta^{D_j} = \cos(\omega).$$

The characteristic function for a normal deviate $\sqrt{n} \eta \sim \mathcal{N}(0,n)$ (note that we have absorbed the $\sqrt{n}$ factor into the variance) is $\exp(-\pi \omega^2/2)$. Thus the characteristic function for $S$ is

$$\tilde{f}_S(\omega) = \cos^{d-1}(\omega) \ e^{-n \omega^2}$$

where $d$ is the number of sites on which $X^u$ and $X^v$ differ (i.e. the Hamming distance between these two strings). Similarly to the onemax case there is a small bias towards choosing the child with the one in the $i$th position. We compute a bound for this bias below.

The probability of choosing a one on site $i$ (averaging over all crossovers and possible noise) is

$$\int_{-1}^{\infty} f(S) \ dS = \frac{1}{2} + \frac{1}{2} \sin^d(\omega) f_S(\omega) \ \frac{d\omega}{2\pi}$$

where we have used that fact that $f(S)$ is symmetric. Using the inverse Fourier transform

$$f(S) = \int_{-\infty}^{\infty} e^{-i\omega S} \tilde{f}_S(\omega) \ \frac{d\omega}{2\pi}$$

and substituting this in and performing the integral over $S$ we find the bias towards choosing the string with a one at site $i$ is

$$b' = \frac{1}{2} \int_{-1}^{1} f(S) \ dS$$

$$= \frac{1}{2} \int_{-\infty}^{\infty} \sin(\omega) \ \tilde{f}_S(\omega) \ \frac{d\omega}{2\pi}$$

$$= \frac{1}{2} \int_{-\infty}^{\infty} \sin(\omega) \ \cos^{d-1}(\omega) \ e^{-n \omega^2} \ \frac{d\omega}{2\pi}$$

where we use the standard notation that $\sin(\omega) = \sin(\omega)/\omega$.

Using $\log(\sin(x)) \geq \log(1-x^2/6)$ and $\log(\cos(x)) \geq \log(1-x^2/2)$ (which follows as the difference in the Taylor expansions involves purely positive terms) then

$$\log(\sin(\omega) \ \cos^{d-1}(\omega)) \geq \log \left(1 - \frac{3d-2}{6}\right) \omega^2$$

so that

$$b' \geq \frac{1}{2} \int_{-\infty}^{\infty} \left(1 - \frac{3n-2}{6}\right) \omega^2 \ e^{-n \omega^2} \ \frac{d\omega}{2\pi}$$

$$= \frac{1}{\sqrt{n} \pi} \left[ \frac{3}{8} + \frac{1}{12n} \right] > 0.211 \frac{1}{\sqrt{n}}.$$

Thus the probability of choosing the child with a 1 at site $i$ rather than 0 is at least $1/2 + b'/\sqrt{n}$. Just as in onemax we are left with a small drift towards fixation at all $1$ of

$$E[M_i] \geq M_i \left(1 + \frac{2b'}{(P-1)\sqrt{n}}\right).$$

The only difference to the onemax case is that we replace $b$ by $b'$. The rest of the proof follow theorem 6. □

5. CONCLUSION

This paper shows that a crossover only evolutionary algorithm is able to solve onemax with high probability provide $P = \Omega(\sqrt{n} \log(n))$ in an expected run time of $O(\sqrt{n} \log(n))$ generations. From some theoretical considerations we believe the true run time is more likely to be $\Theta(\sqrt{n})$ generations or $\Theta(n \log(n))$ function evaluations. This is also supported by empirical evidence, see Fig. 1.

![Figure 1: Empirically measured number of generations to solve onemax using a population of $P = 10 \sqrt{n} \log(n)$. The data points show the mean from simulations with the error bars showing estimated errors in the mean. The dashed line is a fit showing the data is consistent with a run time of approximately $2.8 / \sqrt{n}(1+o(1))$ generations.](image)

We have further shown that using the same algorithm we can also solve onemax in the presence of noise of variance $n$. Gießen and Kötzing had previously shown in [5] that a $(\mu+1)$-EA was able to solve onemax in the presence of noise of variance $O(1)$. However, for an mutation-based search algorithm the variation in fitness of a child population will be dwarfed by our noise. As mutation has a natural tendency to explore the high-entropy part of the solution space the very small bias towards solutions with more ones will be dominated by larger number of solutions with less ones. In contrast, using crossover as a search operator the total number of ones in the children will be same as the parents. Furthermore the fluctuations in the number of ones also has variance of order $n$ so there is considerably more gradient information available. It is the combined effect of focused search and the averaging effect of a population that allows our algorithm to solve onemax with a large degree of noise.

Of course, the problem could also be solved by a hill-climber by re-sampling the same points until the noise was of order 1. This would allow a hill-climber to solve the problem in around $n^2$ iterations. We conjecture, however, that our algorithm would be able to find high quality solutions for onemax plus a random function, where the random function has the same statistical properties as the noise (but does not vary each time the point is sampled). The reason for this is that we believe our algorithm rarely visits the same points of the search space, at least, until the algorithm is close to converging. Thus, our algorithm should behave very
similarly on the two problems, at least, until it is only a small way from the all ones configuration. This is a problem we believe would be hard for algorithms not using crossover.

6. REFERENCES


