Emergent Behaviour, Population-based Search and Low-pass Filtering

Riccardo Poli
Department of Computer Science
University of Essex, UK

Alden H. Wright
Department of Computer Science
University of Montana, USA

Nicholas F. McPhee
Division of Science and Mathematics
University of Minnesota, Morris, USA

William B. Langdon
Department of Computer Science
University of Essex, UK

Technical Report CSM-446
Department of Computer Science
University of Essex
ISSN: 1744-8050
February 2006

Abstract

In recent work we have formulated a model of emergent coordinated behaviour for a population of interacting entities. The model is a modified spring mass model where the masses can perceive the environment and generate external forces. As a result of the interactions the population behaves like a single organism moving under the effect the vector sum of the external forces generated by each entity. When such forces are proportional to the gradient of a resource distribution $f(x)$, the resultant force controlling the single emergent organism is proportional to the gradient of a modified food distribution. This is the result of applying a filtering kernel to $f(x)$. The kernel is typically a low-pass filter.

This model can be applied to genetic algorithms (GAs) and other population-based search algorithms. For example, in previous research, we have found kernels (via genetic programming) that allow the single organism model to track the motion of the centre of mass of GAs and particle swarm optimisers accurately for many generations.

In this paper we corroborate this model in several ways. Firstly, we provide a mathematical proof that on any problem and for any crossover operator, the effect of crossover is that of reducing the amplitude of the derivatives (slopes) of the population distribution. This implies that a GA perceives an effective fitness landscape which is a smoothed, low-pass filtered version of the original. Then, taking inspiration from this result and our active mass-spring model, we propose a class of fitness functions, OneMix, where there is an area of the landscape with high frequency variations. This area contains the global optimum but a genetic algorithm with high crossover probability should not be able "see" it due to its low-pass behaviour. So, a GA with strong crossover should be deceived and attracted towards a local optimum, while with low crossover probability this should not happen. This is, indeed, what happens as we demonstrate with a variety of empirical runs and with infinite-population model simulations. Finally, following our earlier approach, we also evolved kernels for OneMix, obtaining again a good fit between the behaviour of the "single-organism" hill-climber and the GA.

1 Introduction

Many animals live within groups, although, from an individual's point of view, there are clear disadvantages in doing so. For example, in a school, individual fish have limited perception, as their vision is occluded by other individuals and very little light reaches the interior of a school. They have reduced oxygen (so much so that when the school is very tight, like for example in the presence of a predator, fish may even faint). Also, food found by a group member will have to be shared with the rest of the group. So, one may ask, why should individuals stay in a group?

There are obvious benefits of living in groups. For example, there are many things that a group of individuals can do that an isolated individual cannot. The group has increased action and reaction capabilities. Furthermore, a group may have increased sensing capabilities, in the sense that, due to their natural drive towards copying the behaviour of other individuals, it is sufficient to have a small number of in informed individuals for the information to rapidly spread to the whole group. A group clearly has collective information processing capabilities the extent of which is still largely unknown (but, there is evidence that they lead to some form of group intelligence). All this increases the survival capabilities of each individual. Through these extended capabilities, in many circumstances, *a group behaves like a single organism.*

One might think that modelling group behaviour may be very difficult. It appears, however, that this is not the case. That is, rich sets of behaviours can often be explained and modelled using very simple models. For example, it is possible to observe the emergence of realistic fish school behaviours in models [1] where each fish is a simple agent controlled by forces which implement three basic desires: a) behave like your neighbours, b) stay in proximity of other fish, and c) avoid hurting yourself by keeping at a minimum distance from other fish.

Many population-based search algorithms take inspiration from some natural system involving groups of interacting individuals. E.g., particle swarm optimisers (PSOs) are controlled by forces representing cognition and sociality, respectively. In genetic algorithms (GAs) individuals interact through crossover and sample the subspace represented by the convex hull of the population, which is a sort of "social" bias towards sampling areas where other individuals are located. As a result of these interactions search algorithms show complex emergent behaviours, like the ability to avoid getting stuck in local optima. That is, these systems too appear to have new, extended sensing and acting capabilities. A fundamental question is then *to what degree can we interpret and model the behaviour of a population-based search algorithms as that of a corresponding single organism "living" in the same space as the original individuals?*

In the following sections we will try to answer this question.

2 Active-mass/Spring model of populations

To start addressing this question, in [2] we proposed an active-mass/spring model of emergent coordinated behaviour. Since this is the effective inspiration for the work presented here, we summarise it in this section.

Let us imagine that the environment is simply represented by a function $f(x)$ which describes the resource distribution at each point in space. In the case of an artificial system, like a population-based search algorithm, we can think of this as some objective function.

We assume that each individual in a population is a little mass located at some point in the landscape $f(x)$. The individual has an active control system and can perceive the environment and act accordingly to maximise its resources.

2.1 Perception

Perception is limited to a certain area around each individual. Individual animals have to spend time and energy to move to different locations, so sometimes a nearby place may be preferable to further away places even if more distant places provide more food. So, the net value that an animal could gain per unit of food available is some decreasing function $\omega(d)$ of the distance d from the current location. Also, rather naturally, animals prefer places where there are plenty of resources, so the attractiveness of a particular location depends on the average of the available perceived resources. So, we model the *perceived attractiveness* of the environment as

$$a(x) = \int f(y)\omega(x-y)dy$$

2.2 Action

We expect each individual to move in order to maximise the perceived attractiveness of its position. We model this by assuming the individual follows the gradient of $a(x)$ by generating a force

$$F(x) = \eta \nabla a(x) = \eta \int f(y)\nabla\omega(x-y)dy$$

where η is a constant.

2.3 Emergent coordinated behaviour

Now let us assume that individuals interact with other individuals via springs. If the springs are stiff, the population behaves like a rigid body. The motion of the centre of mass $\overline{x}$ of the population is controlled by a force

$$F(\overline{x}) = \sum_i F_i = \eta \sum_i \nabla a(x_i) = \eta \sum_i \nabla a(\overline{x} + \delta_i)$$

where $\delta_i = x_i - \overline{x}$. So,

$$F(\overline{x}) \;=\; \eta \int f(y) \nabla \omega_p(\overline{x} - y) dy,$$

where

$$\omega_p(z) = \sum_i \omega(z + \delta_i).$$

Therefore, the motion of the population is controlled by a force which is proportional to the convolution between the resource distribution and the gradient of a new, coarser-grain kernel, ω_p, representing the perceptual and motor capabilities of the population seen as a single individual. That is, the interactions between individuals transfer information and coordinate behaviour, without any centralised mechanism. The situation is not very different if we reduce the stiffness of the springs.

2.4 What is like to be a population

This single (emergent) organism moves in the environment following the gradient of an attractiveness function

$$a_p(x) = \int f(y) \omega_p(x - y) dy.$$

So, the food distribution (read fitness function) is seen by the population through the lens of a filter/kernel. If the receptive fields, ω, of each individual are delta functions, i.e., each individual has no knowledge about its neighbourhood, then ω_p *is a moving-average-type low-pass filter.*

To give a feel for what being a population might be like, Figure 1 shows a simple fitness function obtained by summing three Gaussians and two-low pass filtered versions of it. These simulate the landscape explored by a population (seen as a single individual) with different interaction strengths between individuals. They have been obtained by convolving the original with Gaussian low-pass filters of with different cut-off frequencies. As one can see, the landscape may look very different to the population, for example, changing from multimodal to unimodal, and, so, becoming potentially easier to search. The low-pass filtering performed by a population, however, does not always imply making the problem easier, as exemplified in Figure 2, where due to interactions a population is deceived into search an area which does not contain a global optimum.

3 Reality or science fiction?

The model presented in the previous section is very simple, and yet it appears to capture the fundamental features of systems, like schools of fish and some population based algorithms, where complex coordinated behaviour emerges from the interaction of relatively simple active components. There are, however, obvious questions regarding the applicability of this model. For example, one might object that a GA is not a rigid body made up of masses

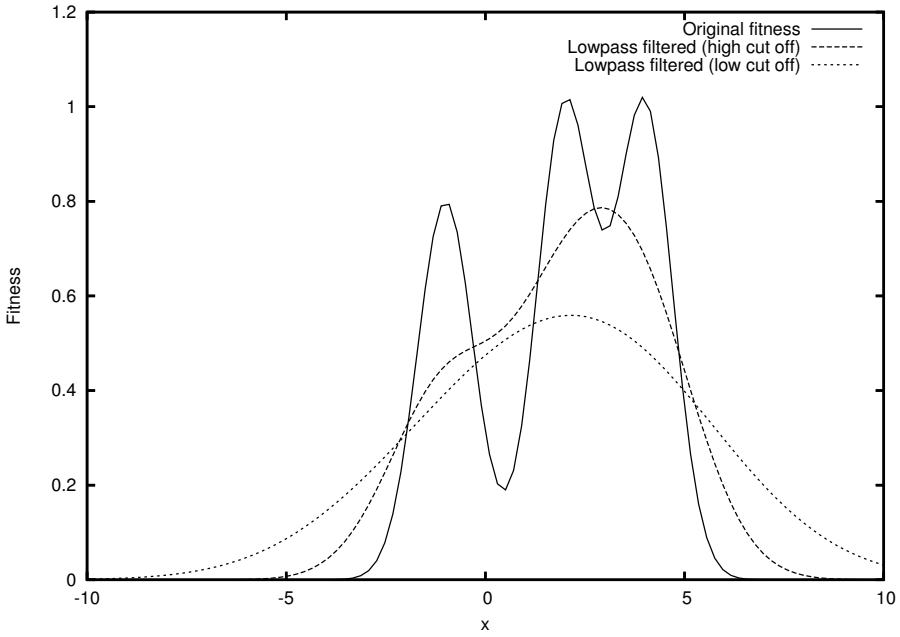

Figure 1: A simple fitness function obtained by summing three Gaussians of same amplitude and standard deviation, and two-low pass filtered versions of it, which have been obtained by convolving the original with Gaussian low-pass filters with different cut-off frequencies.

and that crossover does not provide spring-like interactions. The model might seem more suitable to particle swarm optimisers, but even in that case there are things that don't seem to fit. For example, particles in a PSO have memory and friction. So, really, in what sense can be apply a spring-mass model to these systems?

The key idea is to think about the modelling process for these systems in a new way. GAs, PSOs, evolution strategies, etc., are complex systems with rich dynamics that are hard to understand. Some exact models exist for these systems. For example, both microscopic [7] and coarse-grained [5] models of genetic algorithms exist. These models are very useful and have provided a rich set of results. However, despite the fact that these models have been applied to study the behaviour of algorithms on landscapes that are easy to understand (onemax, sphere, etc.), it is fair to say that they can only occasionally provide simple and intuitive (albeit approximate) explanations of the behaviour of genetic algorithms.

What one could propose to fix this problem is to turn the modelling approach on its head. Let us choose a system that we understand really well – for example, a simple hill-climber – and let us use it as an effective model for our population-based algorithm. Let us then modify the original fitness function in such a way that our simple model behaves as closely as possible to the original. In replacing the original algorithm/system with a simpler one we ignore many of the internal operations of the original, and we should thus only expect to obtain approximations. However, because we understand the new

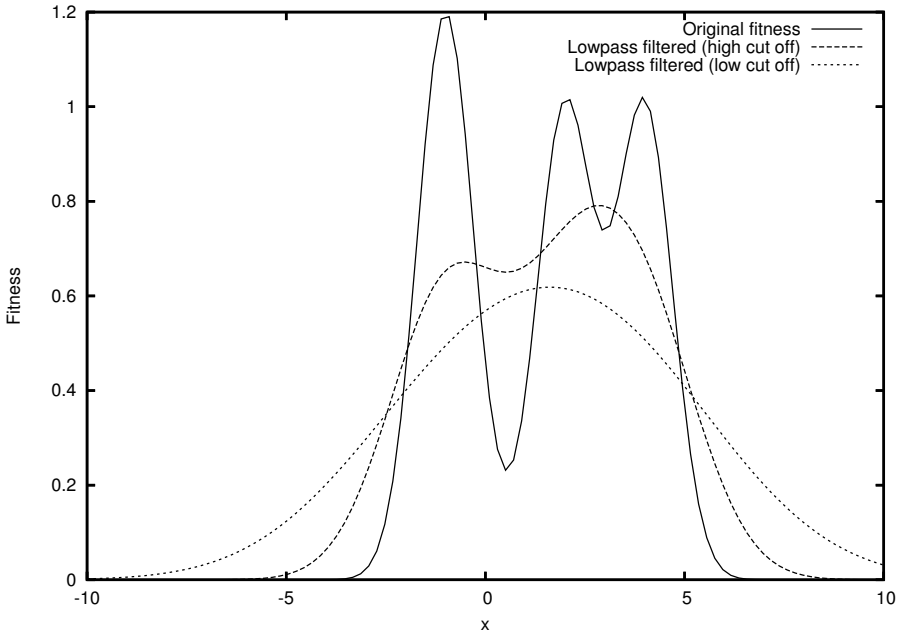

Figure 2: A simple fitness function obtained by summing three Gaussians of same standard deviation but different amplitudes. The two-low pass filtered versions of it show that a population-based algorithm might be deceived into focusing the search in the wrong region.

system really well, in return, we can reasonably hope to understand its behaviour (even if the transformed landscape is complex). This behaviour provides an "effective" intuitive explanation for the behaviour of the original system.

So, the purpose of the modelling effort is not to model the original system in all its details, but to model the effective behaviour of the original system. So, the resulting models are useful if they can provide insights and predictions on the behaviour of the original system.

In [2] we put this idea to the test. There, we used genetic programming to evolve kernels such that a gradient-based hill-climber could track the center of mass of a real algorithm for 20 generations in 5 independent runs. The fitness function for the kernels was

$$f_{\text{kernel}} = \sum_{r=1}^{5} \sum_{t=1}^{20} ||\bar{x}_{\text{pop}}(t) - x_{\text{HC}}(t)||$$

with $x_{\text{HC}}(0) = \bar{x}_{\text{pop}}(0)$. We used this approach for both PSOs and GAs. The RMSE in tracking was surprisingly good (despite our making the problem harder by initialising the populations outside the area where the optima were located). In all cases, the evolved kernels were *low-pass filters*. Figures 3 and 4 show the landscapes as seen by the population (interpreted as a single individual performing hill-climbing) in the case in which $f(x)$ was the Rastrigin function.

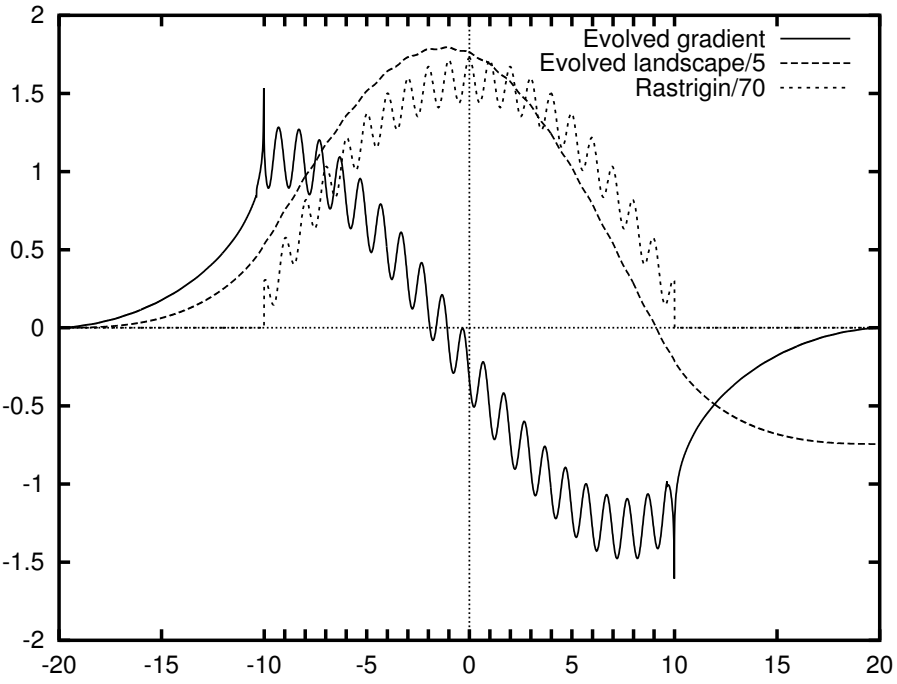

Figure 3: Plot of the Rastrigin function, together with the low-pass filtered version of it seen by a single gradient-based hill-climber behaving like a PSO. The gradient of the resulting landscape ("evolved gradient") is also shown.

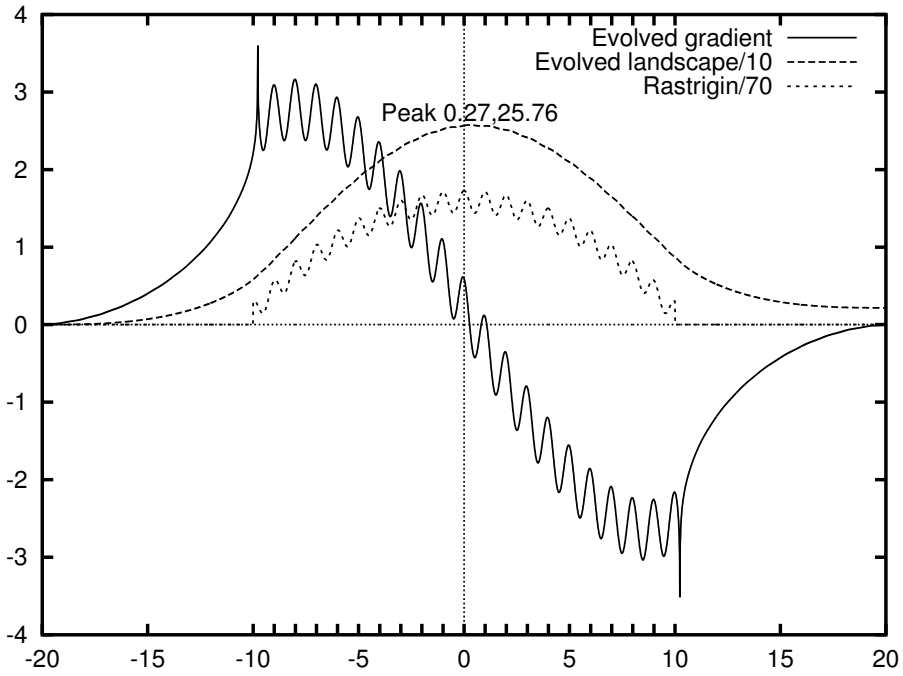

Figure 4: Plot of the Rastrigin function, together with the low-pass filtered version of it seen by a single gradient-based hill-climber behaving like a GA. The gradient of the resulting landscape ("evolved gradient") is also shown.

4 Towards a low-pass filtering theory for GAs

Although the model and results reported in the previous sections may be very interesting, one may ask to what degree can one *prove* that population based algorithms can be seen as hill-climbers on a low-pass filtered version of the landscape. In this section we describe some first steps in this direction, which are based on the analysis of the first order derivatives (differences) of the population distribution.

For simplicity, let us initially focus on a genetic two-bit problem with any assignment of fitnesses $f(00), f(01), f(10), f(11)$. Let us assume that the population is infinitely large and that the proportion of the population occupied by strings of type $x = x_1x_2 \in \{00, 01, 10, 11\}$ is $\Phi(x)$.

If crossover has a low-pass filtering effect, we should expect it to reduce the slopes in the fitness function. We want to verify this, by computing what the average slope in an arbitrary direction would be after the application of crossover.

We know (e.g., by applying the exact schema theorem in [5]) that if we apply one-point crossover with 100% probability to the population, string frequencies in the next generation are given by

$$\Phi'(x_1x_2) = \Phi(x_1*)\Phi(*x_2). \tag{1}$$

Here $*$ is interpreted as a don't care symbol so $\Phi(x_1*) = \Phi(x_10) + \Phi(x_11)$ and $\Phi(*x_2) = \Phi(0x_2) + \Phi(1x_2)$. The average slope in the direction x_2 after the application of crossover, then, is given by

$$\begin{aligned}
&\frac{1}{2}\left(|\Phi'(11) - \Phi'(10)| + |\Phi'(01) - \Phi'(00)|\right) \\
&= \frac{1}{2}\left(|\Phi(1*)\Phi(*1) - \Phi(1*)\Phi(*0)| + |\Phi(0*)\Phi(*1) - \Phi(0*)\Phi(*0)|\right) \\
&= \frac{1}{2}\left(\Phi(1*)|\Phi(*1) - \Phi(*0)| + \Phi(0*)|\Phi(*1) - \Phi(*0)|\right) \\
&= \frac{1}{2}|\Phi(*1) - \Phi(*0)| \\
&= \frac{1}{2}|\Phi(11) - \Phi(10) + \Phi(01) - \Phi(00)| \\
&\leq \frac{1}{2}\left(|\Phi(11) - \Phi(10)| + |\Phi(01) - \Phi(00)|\right)
\end{aligned} \tag{2}$$

So, crossover either leaves derivatives of the population distribution unaltered or (if they have different sign) it reduces them, effectively producing a smoothing action. The same type of result can be proven to hold for the direction x_1 and also for longer strings (as we will show below).

To see how this gives us an indication of what the landscape looks like when seen through the crossover's looking glass, let us imagine that we start our algorithm with all strings having equal proportions, that is $\Phi(x) = \frac{1}{2^\ell}$, where ℓ is string length. If we then apply proportional selection to the population, we obtain a new distribution $\Phi(x) = \frac{1}{2^\ell} f(x)/\bar{f}$.

That is, after proportional selection, string frequencies follow the profile of the string fitnesses. If we then apply crossover, we obtain the distribution $\Phi'(x)$ given by Equation 1.

Naturally, we can interpret $\Phi'(x)$ as the result of applying selection only to a population exploring a modified fitness landscape $f_{\text{eff}}(x)$. This is called *effective fitness* in the literature [3, 4]. Because we started with a uniform distribution $\Phi(x)$ we must have that $\Phi'(x)$ is proportional to $f_{\text{eff}}(x)$. We also know that $\Phi(x)$ is proportional to $f(x)$. So, thanks to Equation 2, we can now see that

$$\begin{aligned} &\frac{1}{2}\left(|f_{\text{eff}}(11) - f_{\text{eff}}(10)| + |f_{\text{eff}}(01) - f_{\text{eff}}(00)|\right) \\ &\quad \leq \quad \frac{1}{2}\left(|f(11) - f(10)| + |f(01) - f(00)|\right). \end{aligned} \tag{3}$$

So, the algorithm behaves as if exploring of a smoothed version of the original landscape. This is generally true as shown by the following:

Theorem 1. *Under the assumption of infinite populations, the application of any type of homologous recombination on a distribution of individuals $\Phi(x)$ where $x \in \{0,1\}^\ell$, produces a new distribution $\Phi'(x)$ such that $\forall i$*

$$\begin{aligned} &\sum |\Phi'(x_1 \cdots x_{i-1}\, 1\, x_{i+1} \cdots x_\ell) - \Phi'(x_1 \cdots x_{i-1}\, 0\, x_{i+1} \cdots x_\ell)| \leq \\ &\quad \sum |\Phi(x_1 \cdots x_{i-1}\, 1\, x_{i+1} \cdots x_\ell) - \Phi(x_1 \cdots x_{i-1}\, 0\, x_{i+1} \cdots x_\ell)| \end{aligned}$$

where sums are over the indices $x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_\ell$.

Proof. In these conditions the exact schema theorem gives us

$$\Phi'(x) = \sum_{m \in \mathcal{M}} p_c(m)\Phi(\Gamma(x,m))\Phi(\Gamma(x,\bar{m}))$$

where $\mathcal{M} = \{0,1\}^\ell$ is the set of all possible crossover masks, $p_c(m)$ is the recombination distribution (the probability of choosing crossover mask m) for the crossover operator, $\Gamma(x,m)$ is a function that given a string (or schema) x returns a schema where the elements of x corresponding to 0's in m are replaced by *'s (don't care symbols), and $\bar{m}$ is the bitwise complement of m. We will use this result in the following.

For notational convenience, we divide $\mathcal{M}$ into $\mathcal{M}_0$ and $\mathcal{M}_1$, which contain all the crossover masks with the i-th bit set to 0 and 1, respectively. Also, for each crossover mask m we divide the set of summation variables $b = \{x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_\ell\}$ into two: $b(m) = \{x_j | j \neq i \wedge m_j = 1\}$ and $\bar{b}(m) = \{x_j | j \neq i \wedge m_j = 0\}$. Finally, we define $\beta_i^1 = x_1 \cdots x_{i-1}\, 1\, x_{i+1} \cdots x_\ell$ and $\beta_i^0 = x_1 \cdots x_{i-1}\, 0\, x_{i+1} \cdots x_\ell$.

Let us calculate the sum of the magnitudes of the differences in proportions $\Phi(x)$ along a particular direction i:

$$
\begin{aligned}
&\sum_b |\Phi'(\beta_i^1) - \Phi'(\beta_i^0)| \\
&= \sum_b \Big| \sum_{m\in\mathcal{M}} p_c(m)\big(\Phi(\Gamma(\beta_i^1, m))\Phi(\Gamma(\beta_i^1, \bar{m})) - \Phi(\Gamma(\beta_i^0, m))\Phi(\Gamma(\beta_i^0, \bar{m}))\big)\Big| \\
&= \sum_b | \sum_{m\in\mathcal{M}_0} p_c(m)\Phi(\Gamma(\beta_i^1, m)) \times \big(\Phi(\Gamma(\beta_i^1, \bar{m})) - \Phi(\Gamma(\beta_i^0, \bar{m}))\big) \\
&\quad + \sum_{m\in\mathcal{M}_1} p_c(m)\Phi(\Gamma(\beta_i^1, \bar{m})) \times \big(\Phi(\Gamma(\beta_i^1, m)) - \Phi(\Gamma(\beta_i^0, m))\big)| \\
&\leq \sum_b \sum_{m\in\mathcal{M}_0} p_c(m)\Phi(\Gamma(\beta_i^1, m)) \times |\Phi(\Gamma(\beta_i^1, \bar{m})) - \Phi(\Gamma(\beta_i^0, \bar{m}))| \\
&+ \sum_b \sum_{m\in\mathcal{M}_1} p_c(m)\Phi(\Gamma(\beta_i^1, \bar{m})) \times |\Phi(\Gamma(\beta_i^1, m)) - \Phi(\Gamma(\beta_i^0, m))| \\
&= \sum_{m\in\mathcal{M}_0} p_c(m) \times \sum_{\bar{b}(m)} |\Phi(\Gamma(\beta_i^1, \bar{m})) - \Phi(\Gamma(\beta_i^0, \bar{m}))| \times \underbrace{\sum_{b(m)} \Phi(\Gamma(\beta_i^1, m))}_{=1} + \\
&\quad \sum_{m\in\mathcal{M}_1} p_c(m) \times \sum_{b(m)} |\Phi(\Gamma(\beta_i^1, m)) - \Phi(\Gamma(\beta_i^0, m))| \times \underbrace{\sum_{\bar{b}(m)} \Phi(\Gamma(\beta_i^1, \bar{m}))}_{=1} \\
&= \sum_{m\in\mathcal{M}_0} p_c(m) \times \sum_{\bar{b}(m)} |\Phi(\Gamma(\beta_i^1, \bar{m})) - \Phi(\Gamma(\beta_i^0, \bar{m}))| \\
&+ \sum_{m\in\mathcal{M}_1} p_c(m) \times \sum_{b(m)} |\Phi(\Gamma(\beta_i^1, m)) - \Phi(\Gamma(\beta_i^0, m))| \\
&= \sum_{m\in\mathcal{M}_0} p_c(m) \times \sum_{\bar{b}(m)} |\sum_{b(m)} \Big(\Phi(\beta_i^1) - \Phi(\beta_i^0)\Big)| \\
&+ \sum_{m\in\mathcal{M}_1} p_c(m) \times \sum_{b(m)} |\sum_{\bar{b}(m)} \Big(\Phi(\beta_i^1) - \Phi(\beta_i^0)\Big)| \\
&\leq \sum_{m\in\mathcal{M}_0} p_c(m) \sum_{\bar{b}(m)} \sum_{b(m)} |\Phi(\beta_i^1) - \Phi(\beta_i^0)| \\
&+ \sum_{m\in\mathcal{M}_1} p_c(m) \sum_{b(m)} \sum_{\bar{b}(m)} |\Phi(\beta_i^1) - \Phi(\beta_i^0)| \\
&= \sum_{m\in\mathcal{M}_0} p_c(m) \sum_b |\Phi(\beta_i^1) - \Phi(\beta_i^0)| + \sum_{m\in\mathcal{M}_1} p_c(m) \sum_b |\Phi(\beta_i^1) - \Phi(\beta_i^0)| \\
&= \sum_{m\in\mathcal{M}} p_c(m) \sum_b |\Phi(\beta_i^1) - \Phi(\beta_i^0)| = \sum_b |\Phi(\beta_i^1) - \Phi(\beta_i^0)| \underbrace{\sum_{m\in\mathcal{M}} p_c(m)}_{=1} \\
&= \sum_b |\Phi(\beta_i^1) - \Phi(\beta_i^0)|
\end{aligned}
$$

□

So, for example, for strings of length $\ell = 4$ and for the direction x_3 we have

$$\sum_{x_1x_2x_4} |\Phi'(x_1x_21x_4) - \Phi'(x_1x_20x_4)| \leq \sum_{x_1x_2x_4} |\Phi(x_1x_21x_4) - \Phi(x_1x_20x_4)|,$$

with equality holding only in fairly restricted cases.

We can now generalise the previous theorem

Theorem 2. *$\forall i$ and for any subset $S \subset \{1, \cdots, \ell\}$ such that $i \notin S$, under the assumption of infinite populations, the application of any type of homologous recombination on a distribution of individuals $\Phi(x)$ where $x \in \{0,1\}^\ell$, produces a new distribution $\Phi'(x)$ where*

$$\sum |\Phi'(x_1 \cdots x_{i-1}\, 1\, x_{i+1} \cdots x_\ell) - \Phi'(x_1 \cdots x_{i-1}\, 0\, x_{i+1} \cdots x_\ell)| \leq$$
$$\sum |\Phi(x_1 \cdots x_{i-1}\, 1\, x_{i+1} \cdots x_\ell) - \Phi(x_1 \cdots x_{i-1}\, 0\, x_{i+1} \cdots x_\ell)|$$

where $x_j = *$ for all $j \in S$ and the sums are over the index set $b = \{x_k\}_{k \notin S \setminus \{i\}}$.

Proof. We proceed as in the previous theorem. Again, for each crossover mask m we divide the set of summation variables b into two: $b(m) = \{x_j \in b | m_j = 1\}$ and $\bar{b}(m) = \{x_j \in b | m_j = 0\}$. Also, we define $\beta_i^1 = x_1 \cdots x_{i-1}\, 1\, x_{i+1} \cdots x_\ell$ and $\beta_i^0 = x_1 \cdots x_{i-1}\, 0\, x_{i+1} \cdots x_\ell$, with the only difference is that these are now schemata. With this minimal changes in the interpretation of symbols, the proof for this theorem involves *exactly* the same formal calculations as the proof for the previous theorem. We don't repeat them for brevity. □

So, for example, for strings of length $\ell = 4$ and for the direction x_2 we have

$$\sum_{x_1x_4} |\Phi'(x_1 1 * x_4) - \Phi'(x_1 0 * x_4)| \leq \sum_{x_1x_4} |\Phi(x_1 1 * x_4) - \Phi(x_1 0 * x_4)|.$$

For the same direction we also have

$$\sum_{x_1} |\Phi'(x_1 1 * *) - \Phi'(x_1 0 * *)| \leq \sum_{x_1} |\Phi(x_1 1 * *) - \Phi(x_1 0 * *)|$$

and

$$\sum_{x_4} |\Phi'(*1 * x_4) - \Phi'(*0 * x_4)| \leq \sum_{x_4} |\Phi(*1 * x_4) - \Phi(*0 * x_4)|.$$

For both theorems we should expect a strict inequality to hold except for a small set of pathological cases. For example, a sufficient condition for the strict inequality to apply in Theorem 1 is that there exists an assignment for $x_1 \cdots x_{i-1}, x_{i+1} \cdots x_\ell$ such that there exist two crossover masks m_1 and m_2 with $p_c(m_i) > 0$ for which $\text{sign}\big(\Phi(\Gamma(\beta_i^1, m_1)) - \Phi(\Gamma(\beta_i^0, m_1))\big) \neq \text{sign}\big(\Phi(\Gamma(\beta_i^1, m_2)) - \Phi(\Gamma(\beta_i^0, m_2))\big)$. A (pathological) case where equality holds is, for example, the case of order-1 schemata, since homologous crossover preserves order-1 schema frequencies (as one can verify by applying the exact schema theorem to order one schemata). E.g., for $\ell = 3$ and the x_2 direction we have

$$|\Phi'(*1*) - \Phi'(*0*)| = |\Phi(*1*) - \Phi(*0*)|.$$

5 How GAs really work

Based on the results of the previous sections we are now in a position to provide some new intuitions on how genetic algorithms perform their search.

Initially, if the population is spread over a large region, the GA will perform low-pass filtering at very low cut-off frequency. So, only major, low-frequency features of the landscape can be perceived. The GA will, for example, look for broad areas where the average fitness is high. As the population starts converging the landscape starts refocusing. This is the result of the fact that homologous crossover can only explore the now-much-smaller convex hull of the population. The GA can now perform a finer optimisation within that region. Finally, when the population is very converged, the cut-off frequency is very high and, so, crossover becomes an all-pass filter, thereby revealing all of the structure of the fitness function, although limited to the region currently being explored.

From this qualitative description it seems obvious that a GA with intense crossover can easily be deceived in the early phases of runs. If the global optimum is in a region dominated by high frequency components, the GA will initially be unable to perceive that part of the landscape correctly. So, if the low-frequency components of the landscape induce the algorithm to move the search in an area not containing the global optimum, the algorithm will never find it, being unable to undo its incorrect early decisions.

6 The OneMix problem class

As a further test for our theoretical and qualitative explanations, we decided to design and test a fitness function having exactly these features.

The function is a mixture of the OneMax problem and a ZeroMax problem. Like these it is a function of unitation, u, which represents the number of 1s in a string. For unitation values bigger than $\ell/2$ our new function is just OneMax. For lower unitation values, it is OneMax if u is odd, a scaled version of ZeroMax, otherwise. The new function, which we call *OneMix*, is formally defined as

$$f(u) = \begin{cases} (1+a)(\ell/2-u)+\ell/2 & \text{if } u \text{ is even} \\ & \text{and } u < \ell/2 \\ u & \text{otherwise,} \end{cases}$$

where $a > 0$. With this constraint we ensure that the global optimum is the string $00\cdots0$.

Figure 5 shows the OneMix function for $\ell = 100$ and $a = 0.6$, i.e., the fitness of the global optimum is 1.3ℓ while the fitness of the deceptive attractor is ℓ. Note the high-frequency components in the area containing the global optimum. Also, note that the average (low-pass filtered) fitness in that area is *lower* than in the basin of attraction of the deceptive attractor $11\cdots1$.

In order to assess the behaviour of a GA on this class of functions, we performed a number of both simulations and empirical runs. These are described in the following sections.

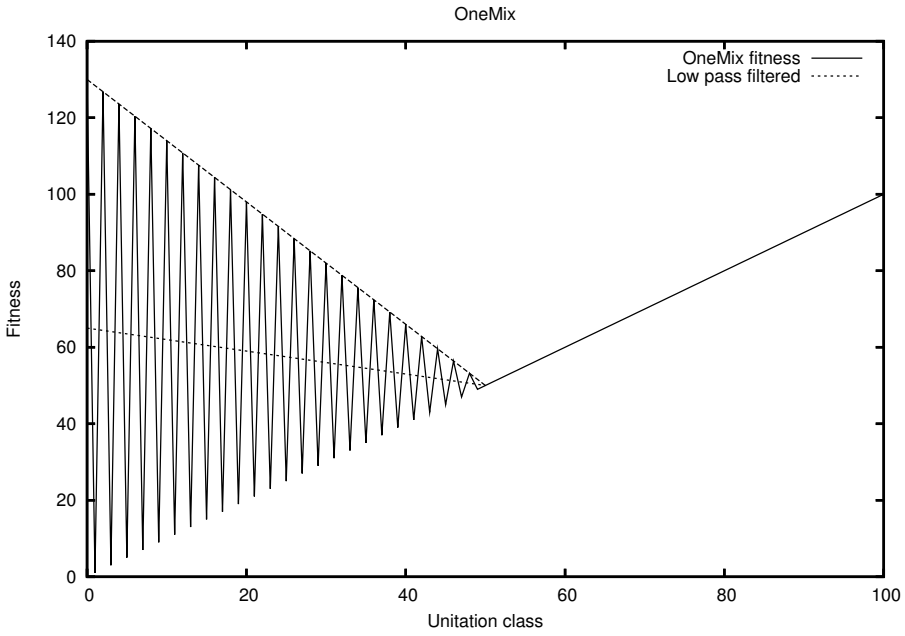

Figure 5: OneMix fitness function for $\ell = 100$.

7 Evolved kernels for OneMix

To test whether it is possible to model the behaviour of a GA solving the OneMix problem as a single hill-climber searching a modified (low-pass filtered) landscape, we used again the technique presented in [2] and briefly summarised in Section 3. That is we used genetic programming to evolve kernels such that a gradient-based hill-climber could track the center of mass of the GA as closely as possible.

We considered two settings: a GA with $1/\ell$ mutation and a GA with uniform crossover. In both cases the population size was 1000, string length was $\ell = 200$ and we used tournament selection with tournament size 2. The genetic algorithm was a steady state GA using negative tournaments for replacement. We ran both algorithms for 40 generations and starting from 5 different (random) initial conditions. Figure 6 shows a plot of the OneMix function (for $\ell = 200$ and $a = 0.6$), together with the low-pass filtered version of it evolved by GP. As shown in this figure and also in Figure 7, a hill-climber following the gradient of the evolved landscape is able to track the center of mass of the GA population very closely for about 15 generations, failing only when the GA population loses diversity. The differential of the evolved kernel is expression $\frac{u}{11+u}$.

For comparison, Figure 8 shows a run of a GA without crossover (but with $1/\ell$ mutation) with the corresponding trajectory of a hill-climber following the gradients of an evolved landscape. Here again we were able to evolve a landscape such that a hill-climber provides a good approximation of the behaviour of a GA. Figure 9 shows the evolved kernel and the result of convolving it with the OneMix landscape. The differential of the evolved kernel is expression $\frac{u}{0.1448 \times u^2 - 20}$.

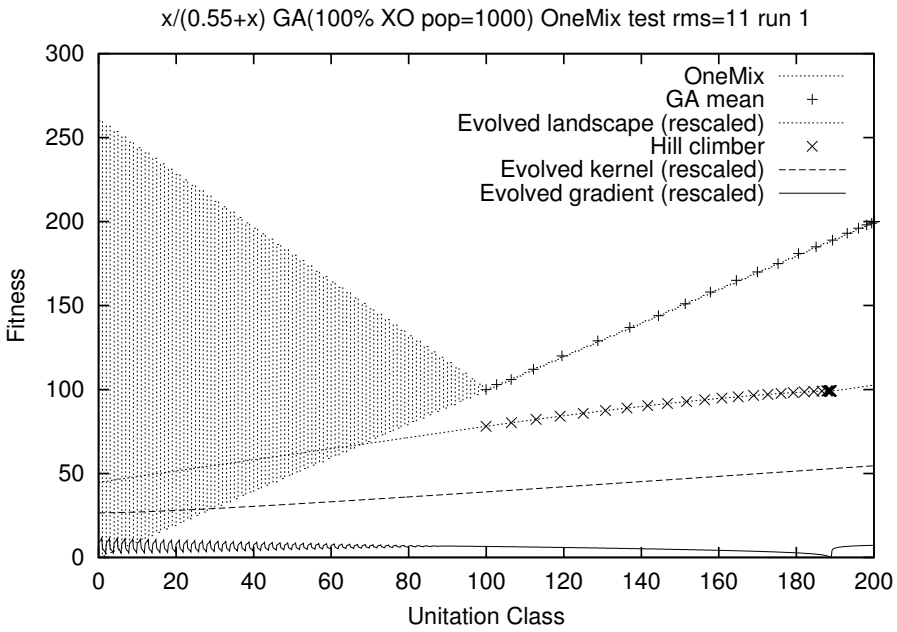

Figure 6: Plot of the OneMix function ($\ell = 200, a = 0.6$), together with the low-pass filtered version of it seen by a single gradient-based hill-climber behaving like a GA with uniform crossover. The center of mass of the GA population and the trajectory of the hill-climber are also shown.

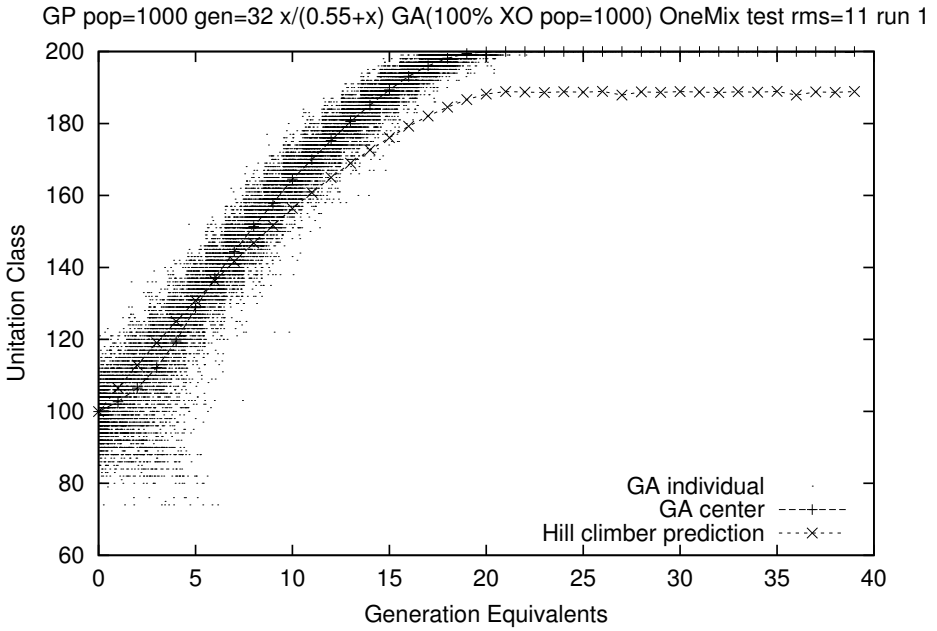

Figure 7: One typical run of a GA with uniform crossover on the OneMix function together with the corresponding trajectory of the hill-climber.

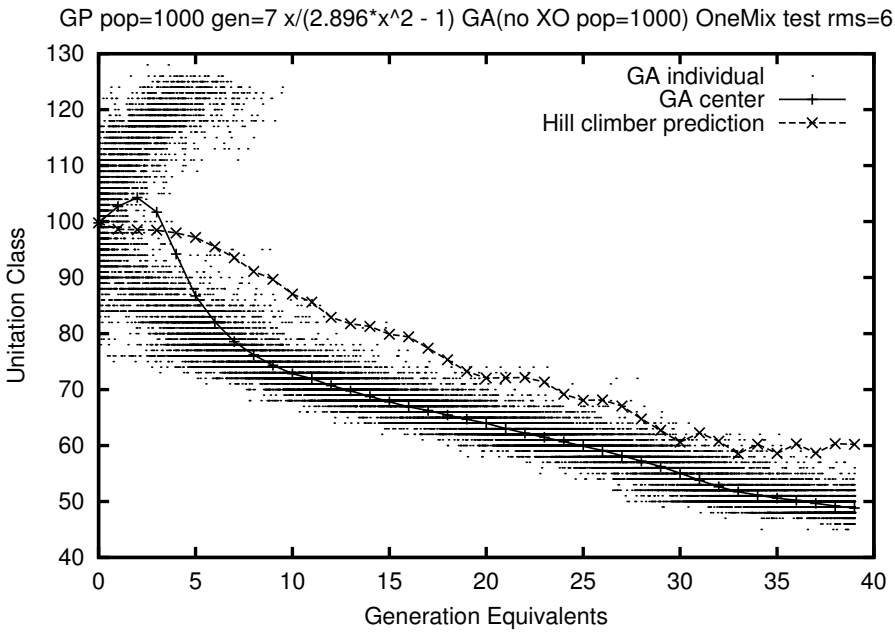

Figure 8: One typical run of a GA with mutation but without crossover on the OneMix function together with the trajectory of the hill-climber.

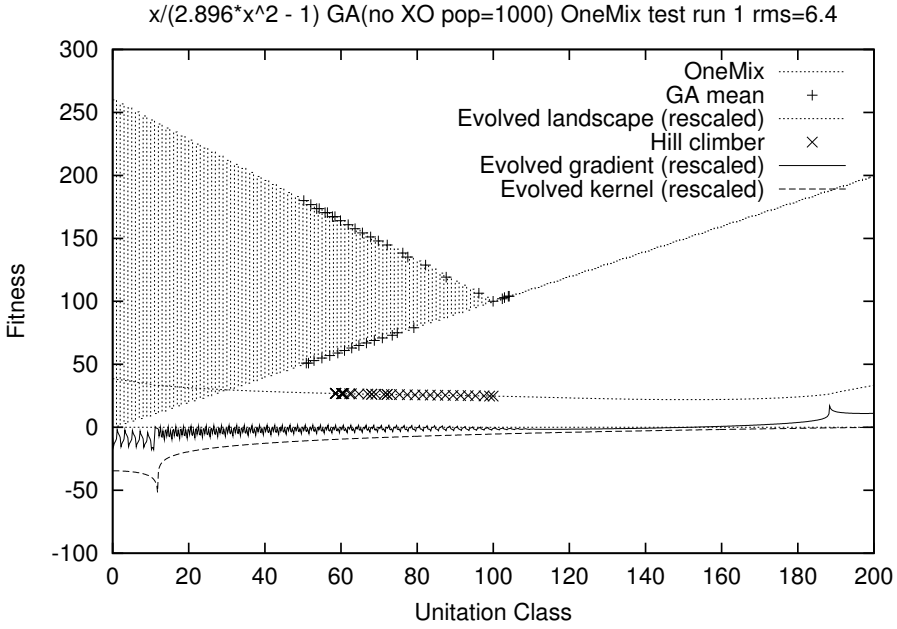

Figure 9: Plot of the OneMix function ($\ell = 200, a = 0.6$), together with the low-pass filtered version of it seen by a single gradient-based hill-climber behaving like a GA with mutation (no crossover). The center of mass of the GA population and the trajectory of the hill-climber are also shown.

8 Infinite population model

Normally exact probabilistic models of genetic algorithms are huge, due to the number of degrees of freedom implied in even small populations and short bit strings. However, for fitness functions of unitation and GAs that use mutation and uniform crossover, a more manageable infinite population model has been developed in [8]. This model assumes that all strings in each unitation class are equally likely. This condition will be met by a random initial (infinite) population. The model model also requires the assumption that the infinite population model trajectory converges to a fixed point where strings in each unitation class are equally likely. The model describes how the distribution of strings in each unitation class varies from one generation to the next. By iterating the transition matrices of the model, one can then obtain the exact time evolution of behaviour for the GA (under the assumption of infinite population) for relatively large string lengths and numbers of generations.

To study the behaviour of a GA on the OneMix function (with $a = 0.6$), we used an infinite-population unitation model with strings of length $\ell = 50$ and truncation selection, with truncation fraction 1/6 (i.e., only the best 1/6 of the population was saved). After selection we used either uniform crossover (applied with 100% probability) or standard mutation with mutation rate $1/\ell$. (The same results could have been obtained by averaging GA runs with very large populations.)

In the simulations with uniform crossover with rate 1, the population went right (to the smooth side), while with mutation only it went left (to the zigzag side) of the OneMix function. This is exactly what we predicted previously.

What is perhaps even more interesting are plots of the population after one round of truncation selection, after truncation selection and crossover, and after truncation selection and mutation. These are shown in Figure 10. As one can see, selection tries to amplify high-frequency components (particularly peaks) in the fitness function, while both mutation and crossover have a smoothing effect. However, the smoothing effect of crossover is much stronger and is such to completely mask the high-fitness points on the left of the unitation landscape.

9 Empirical GA runs with One-Mix

To provide a final corroboration to the infinite population model results and to our theoretical results on the low-pass filtering behaviour of crossover, we performed a series of empirical runs with a variety of different GAs, different values of ℓ, and different operators and probabilities of applications of the operators.

Starting with the case $\ell = 6$, fitness proportionate selection, uniform crossover, random initialisation, and $a = 0.6$, Figure 11 shows that with a low crossover rate, in 1000 independent runs, 25% of the populations converged to the optimal string 000000, while only about 18% converged to the deceptive attractor 111111. The remaining runs converged to other strings, due to the small population and the absence of mutation. The effect is

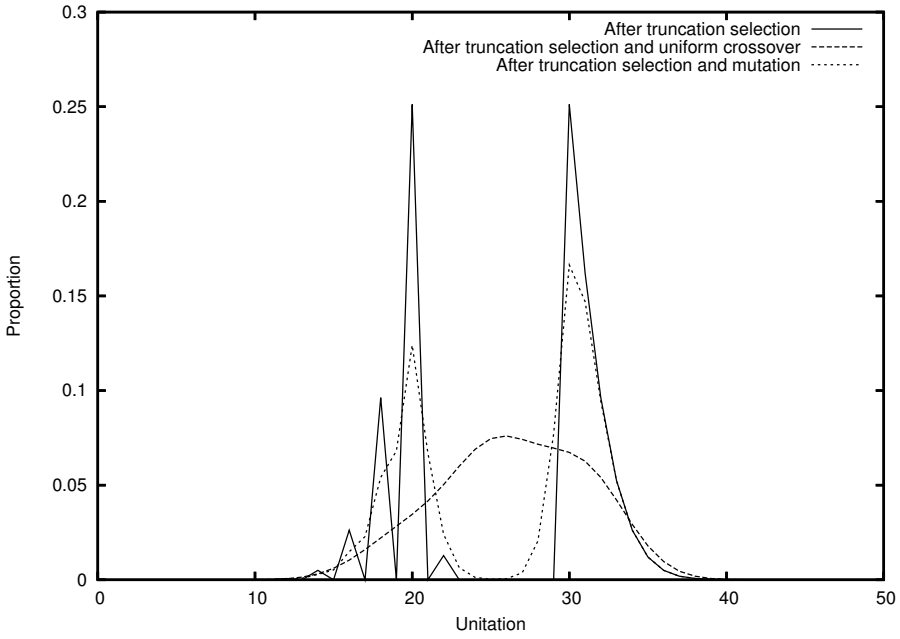

Figure 10: Distribution of unitation after truncation selection both alone and followed by either crossover or mutation.

greatly amplified if one uses larger populations, as shown in Figure 12.

The situation is completely different if the crossover rate in increased to $p_c = 1.0$, as shown in Figures 13 and 14.

Both of the above results are in substantial agreement with the infinite population model. In fact, in both cases, the uniform population consisting of copies of the 000000 string and the uniform population consisting of copies of the 111111 string are stable fixed points of the infinite population model[1]. In the first case with the lower crossover rate, an infinite population corresponding to a random initial population is close to the boundary between the basin of attraction of these fixed points. In the second case with a higher crossover rate, this initial population is in the basin of attraction of the uniform population consisting of copies of the 111111 string.

The same qualitative behaviour was obtained in runs with $\ell = 200$ and tournament (with tournament size 6), two-point crossover (applied with 100% probability) and mutation with a mutation rate of $1/\ell$, as illustrated in Figure 15. Note that if crossover is absent mutation always searches the region of the global optimum reaching it most of the times. In the presence of crossover, however, the algorithm finds the global optimum only in about 1/3 of the runs, converging to the deceptive attractor in all other runs.

[1]The stability of these fixed points can be determined exactly as shown in [6]. In fact, the spectrum of the differential can be computed without iterating the model.

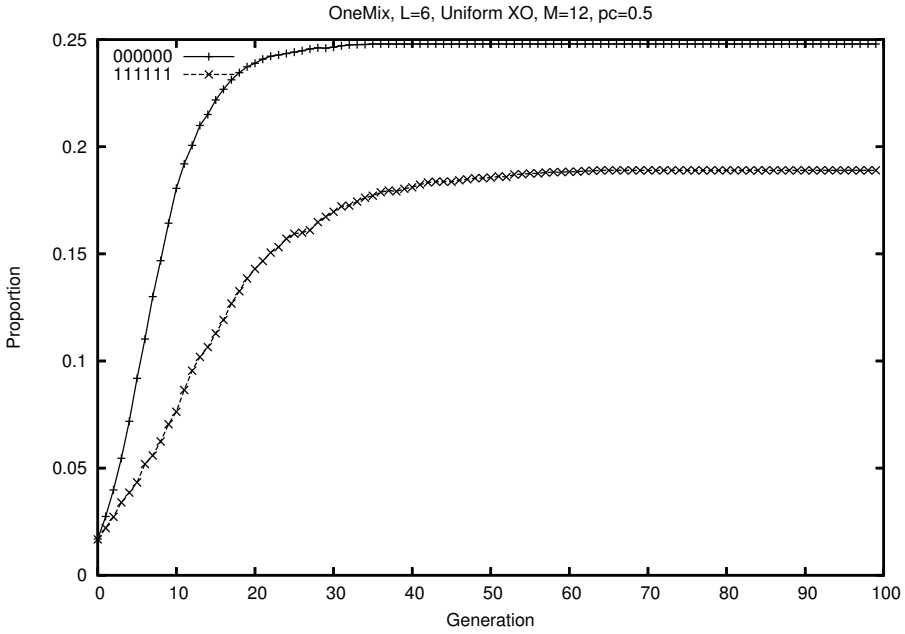

Figure 11: Behaviour of a GA on a OneMix problem of length $\ell = 6$, with uniform crossover applied with probability $p_c = 0.5$ and a population size $M = 12$ (no mutation). Averages of 1000 runs.

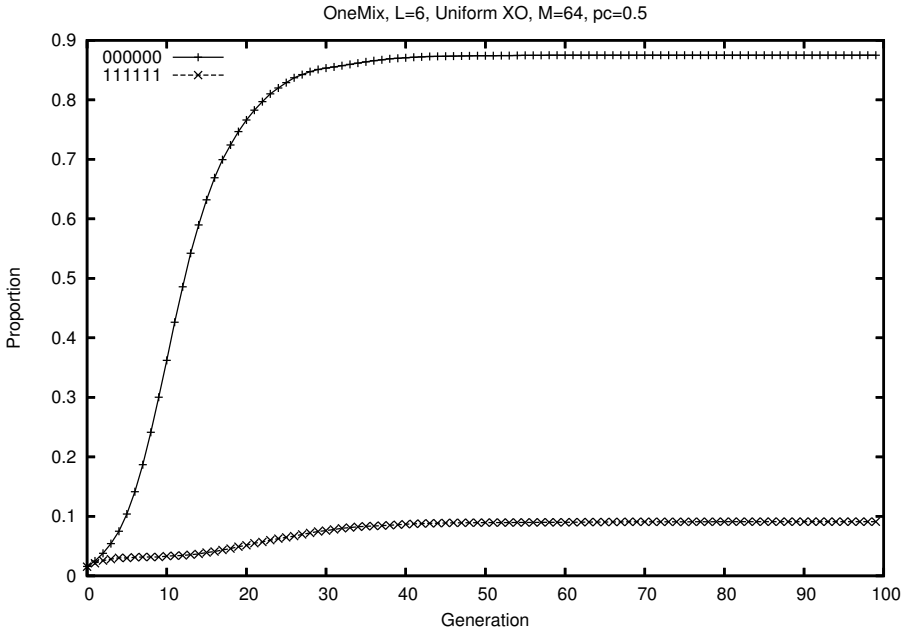

Figure 12: Behaviour of a GA on a OneMix problem of length $\ell = 6$, with uniform crossover applied with probability $p_c = 0.5$ and a population size $M = 64$ (no mutation). Averages of 1000 runs.

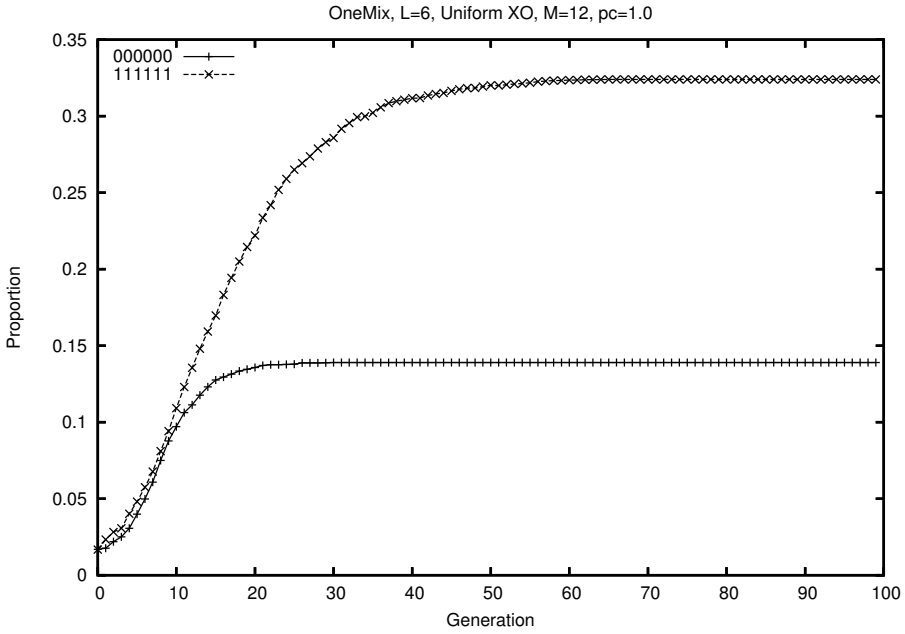

Figure 13: Behaviour of a GA on a OneMix problem of length $\ell = 6$, with uniform crossover applied with probability $p_c = 1.0$ and a population size $M = 12$ (no mutation). Averages of 1000 runs.

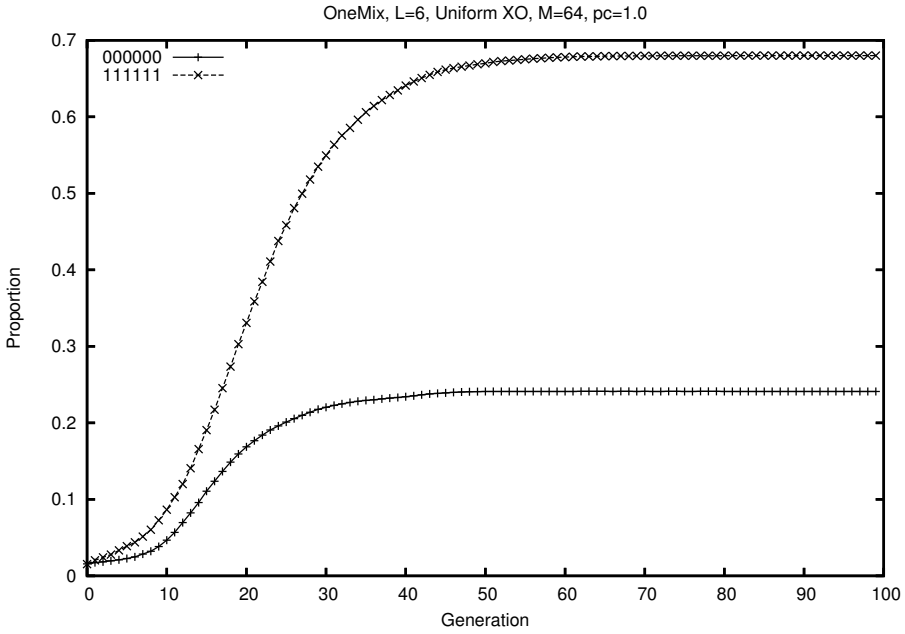

Figure 14: Behaviour of a GA on a OneMix problem of length $\ell = 6$, with uniform crossover applied with probability $p_c = 1.0$ and a population size $M = 64$ (no mutation). Averages of 1000 runs.

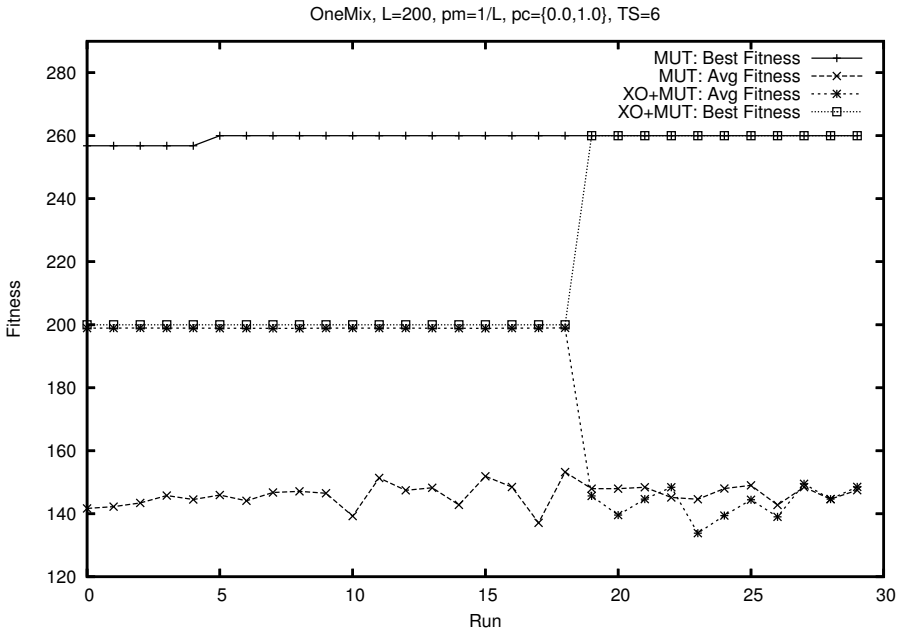

Figure 15: Behaviour of a GA on a OneMix problem of length $\ell = 200$ in the presence of $1/\ell$ mutation with and without two-point crossover applied with 100% probability, in 30 independent runs with each setting.

10 Conclusions

GAs and other population-based algorithms integrate the information provided by individuals. As a result they can be thought as a single, macroscopic individual moving on a deformed (smoothed) landscape.

In this paper, we have shown this empirically and started to build a theoretical basis for the low-pass behaviour of genetic algorithms with strong crossover.

The interpretation of crossover as low-pass filtering has allowed us to create new, simple but non-contrived landscapes, the family of OneMix functions, where crossover does not help and for which we understand why.

Naturally, we could use these ideas to develop landscapes where crossover would do very well. This will be the subject of future research.

Acknowledgement

Supported by EPSRC XPS grant GR/T11234/01. We would like to thank Darren Croft, Iain Couzin, Chris Stephens, Michael Vose and Thomas Jansen for useful discussions. We would also like to thank Schloss Dagstuhl International Conference and Research Center for Computer Science, where a significant portion of this research has cooperatively been carried out.

References

[1] I. D. Couzin, J. Krause, N. R. Franks, , and S. A. Levin. Effective leadership and decision-making in animal groups on the move. *Nature*, 433:513–516, 2005.

[2] W. B. Langdon and R. Poli. Finding effective landscapes for PSOs via kernels. In *IEEE World Conference on Computational Intelligence*, 2006. submitted.

[3] C. R. Stephens and J. Mora Vargas. Effective fitness as an alternative paradigm for evolutionary computation I: General formalism. *Genetic Programming and Evolvable Machines*, 1(4):363–378, October 2000.

[4] C. R. Stephens and J. Mora Vargas. Effective fitness as an alternative paradigm for evolutionary computation II: Examples and applications. *Genetic Programming and Evolvable Machines*, 2(1):7–32, March 2001.

[5] C. R. Stephens and H. Waelbroeck. Schemata evolution and building blocks. *Evolutionary Computation*, 7(2):109–124, 1999.

[6] M. D. Vose and A. H. Wright. Stability of vertex fixed points and applications. In L. D. Whitley and M. D. Vose, editors, *Foundations of genetic algorithms 3*, pages 103–113, San Mateo, 1995. Morgan Kaufmann.

[7] Michael Vose. Modeling simple genetic algorithms. In *FOGA-92, Foundations of Genetic Algorithms*, Vail, Colorado, 24–29 July 1992.

[8] Alden H. Wright and J. Neal Richter. Strong recombination, weak selection, and mutation. Department of Computer Science, University of Montana, Missoula, MT 59812 USA, http://www.cs.umt.edu/u/wright/papers/bistability2006.pdf, submitted to GECCO 2006.