Theoretical Computer Science 229 (1999) 143–175

Theoretical Computer Science

www.elsevier.com/locate/tcs

Fourier Analysis of Genetic Algorithms

Walter A. Kosters[a,] *, Joost N. Kok[a], Patrik Floréen[b]

[a] *Leiden Institute of Advanced Computer Science, Universiteit Leiden, P.O. Box 9512, 2300 RA Leiden, The Netherlands*

[b] *Department of Computer Science, University of Helsinki, P.O. Box 26, FIN-00014 University of Helsinki, Finland*

Abstract

We propose a general framework for Fourier analysis in the field of genetic algorithms. We introduce special functions, analogous to sine and cosine for real numbers, that have nice properties with respect to genetic operations such as mutation and crossover. The special functions we introduce are generalizations of bit products and Walsh products. As applications, we trace (both analytically and numerically) the behavior of genetic algorithms, and obtain results on the fitness of schemata. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Fourier analysis; Genetic algorithms; Walsh products

1. Introduction

In this paper we introduce a method for examining the fundamental properties of genetic algorithms (see [7, 11]).

A genetic algorithm works on a multiset of bit strings of fixed length. A fitness function is given which measures the quality of the elements of the multiset. The elements of the multiset are usually called individuals and the multiset is called the population. A genetic algorithm applies genetic operators in order to improve the multiset. Standard genetic operators are proportional selection, uniform crossover, one point crossover and mutation. In this paper we will also consider some non standard operators including one bit mutation and masked crossover.

Following [23] (based on earlier work by Vose), we consider infinite populations, i.e., we view a population as a probability distribution and we see how such a distribution changes under the genetic operators.

Proportional selection means that the probability for an individual to be chosen is proportional to its fitness. Uniform crossover takes two bit strings (parents) and

* Corresponding author.
E-mail addresses: kosters@wi.leidenuniv.nl (W.A. Kosters), joost@wi.leidenuniv.nl (J.N. Kok)

produces an offspring by taking as the ith element randomly with equal probability one of the two ith elements in the parents. One-point crossover takes two n-bit parents, chooses a random crossover point $\ell \in \{1,\dots,n-1\}$ and gives as result a bit string consisting of the ℓ first bits of one parent and the $n-\ell$ last bits of the other parent. Mutation changes each element of a bit string to the opposite value with probability p (mutation rate).

By repeated application of the genetic operators on the distributions, it is possible to trace the distribution from generation to generation, thus simulating genetic algorithms. There is a relationship between the deterministic path of the distributions and models of genetic algorithms with finite population size motivating the tracing of distributions (especially in the work of Vose and others, see [13, 15, 22, 24, 25], and also [16]).

In this paper we propose alternative structures for modelling distributions. These structures are equivalent to distributions in the sense that distributions can be derived from these structures, and vice versa. The application of genetic operators to these structures gives rise to nice formulas. The alternative structures consist of a general framework for Fourier analysis in the field of genetic algorithms. We introduce special functions, analogous to sine and cosine for real numbers, that have nice properties with respect to genetic operations such as mutation and crossover. Instead of tracing distributions, we trace expected values of these special functions.

The special functions are of the following form. For any complex number a and subsets i and x of a finite universe U we define

$$G_{a,i}(x) = a^{\|i\setminus x\|}.$$

Here $\|s\|$ denotes the number of elements of the set s, and $i\setminus x$ consists of those elements in i that are not in x.

The so-called bit products $B_i(x)$ and Walsh products $R_i(x)$ (see for example [6, 8, 9, 14]) are special cases. Given an index set i and a bit string x, a bit product $B_i(x)$ multiplies those elements of the bit string x that are in the index set i. Walsh products $R_i(x)$ are similar to the bit products, but the bit strings are changed into $\{-1,1\}$-strings, and products are taken on the $\{-1,1\}$-strings.

In [17] expected values of bit products are used for obtaining bounds on the rate of convergence. In that paper the following restrictions are made: the distributions need to be symmetric, only the fitness function that counts the number of ones in a bit string is considered, and only proportional selection and uniform crossover are treated. In the literature (e.g., [3, 8, 9]), several applications of Walsh products (but not expected values of Walsh products) can be found in the field of genetic algorithms, for example for the construction of deceptive fitness functions (functions that are difficult for genetic algorithms), for the construction of a number of measures for the state of a population, and for the analysis of the fitness of a schema. Moreover, [21] gives a spectral analysis of schemata and [12] proposes to use so-called Haar functions instead of Walsh functions.

Expected values of bit and Walsh products are similar, but also have their relative merits. For example, expected values of Walsh products are better suited for the analysis of schemata, while it seems that bit products are more useful in establishing bounds on convergence times. Usually, bit products are more intuitive due to their more direct connection to bit strings, and this makes it easier to understand their properties. This paper gives a unifying framework for these different products which has the advantages of both approaches.

Often one is not interested in the distribution itself, but in the population mean (expected value) of a measurement function ϕ (cf. [1]). With a measurement function one observes certain properties of a distribution. A measurement function takes as input a bit string x and yields a numerical value. Examples of measurement functions are

(1) the fitness (population mean is the mean fitness of individuals): $\phi(x) = f(x)$,

(2) the square of the fitness (population mean is the second moment of the fitness of the individuals): $\phi(x) = (f(x))^2$,

(3) elements of schema h (population mean is the probability of the schema): $\phi(x) = 1$ if $x \in h$ and 0 otherwise.

A schema is a string over $\{0, 1, *\}$. The notation $x \in h$ means here that x is an instance of h, i.e., x can be obtained by replacing $*$'s in h by zeros or ones. One of the applications of the paper is to show that the population mean for different measurement functions can be traced using the expected values of the special functions.

We also consider as special cases symmetric distributions and fitness functions whose values are determined by the number of ones in the string (see [10, 20]). These special cases are of interest because the formulas that describe the change in expected values become even simpler, and the time complexity reduces because the number of different expected values to be traced is only linear instead of exponential as it is in the general case. It is then possible to go analytically through a simple genetic algorithm. We also examine the n-queens problem with our method.

We show that the value of a measurement function on a schema in an arbitrary distribution can be calculated from expected values of Walsh products. As instances of this result, we obtain both the uniform and nonuniform Walsh-schema transform.

The rest of the paper is organized as follows. In the next section we discuss the infinite population model. Section 3 presents the Fourier analysis. Section 4 is about expected values of the basic functions. In Section 5 we continue with the tracing of expected values. Section 6 and Section 7 give instantiations of the formulas for Walsh products and bit products, respectively. In Section 8 we discuss symmetric populations, in Section 9 we present the non-uniform Walsh-schema transform, and Section 10 gives several other applications. We end with a discussion.

2. Infinite population model

A genetic algorithm operates on a set of bit-strings (the so-called population) using genetic operations. These genetic operations include selection, mutation, and several types of crossover. We use distributions $P(x)$ to express the probability that a string x occurs at a certain time.A distribution associates to each bit string x a probability $P(x)$

such that $\sum_x P(x)=1$. We are interested in the connection between the distribution just before $(P(x))$ and immediately after $(P'(x))$ a genetic operation. Of course, it is also possible to keep part of the original distribution, leading to the distribution $(1-q)P(x)+qP'(x)$, where q denotes the probability of the genetic operation involved. For crossover, q is usually called the crossover rate.

We identify bit-strings consisting of n bits with subsets of a universe U in the appropriate way: an element of U is in the "set" x if and only if the corresponding bit equals 1. Using the usual binary representation it is also possible to view bit-strings as integers.

First we consider how the distribution changes under the application of the operators. From the definitions of the operators we obtain the following formulas corresponding with the Vose and Liepins model from [23].

For proportional selection, the probability of a bit string is weighted proportional to its fitness: For example, if the fitness of a bit string is twice as good as the average, proportional selection results in doubling the probability. Selection uses a fitness function f, and satisfies

$$P'(x)=\frac{f(x)}{E[f]}P(x).$$

For mutation we have

$$P'(x)=\sum_y p^{\|\mathrm{xor}(x,y)\|}(1-p)^{n-\|\mathrm{xor}(x,y)\|}P(y),$$

where $p\in[0,1]$ is the mutation rate, and $\mathrm{xor}(l,m)=(l\backslash m)\cup(m\backslash l)$ is the symmetric difference of l and m (which is exactly the Hamming distance between l and m).

Next we examine "one-bit mutation", where exactly one bit from the parent is toggled; this bit is chosen randomly. We have

$$P'(x)=\frac{1}{n}\sum_{\substack{y\\ \|\mathrm{xor}(x,y)\|=1}} P(y).$$

Notice that, given x, precisely n sets y occur in this summation.

Uniform crossover can be expressed as follows. If both parents have a 1 in a certain bit-position, the child receives a 1 too (then we are part of the intersection of the two sets); the same holds for a 0. If the two bits from the parents differ, 0 and 1 are equally likely for the child (now we are part of the symmetric difference). Hence

$$P'(x)=\sum_{\substack{y,z\\ y\cap z\subseteq x\subseteq y\cup z}} (1/2)^{\|\mathrm{xor}(y,z)\|}P(y)P(z).$$

As a further example we slightly change uniform crossover: if the bits from the parents are equal, the child receives a zero bit. We then have

$$P'(x)=\sum_{\substack{y,z\\ x\subseteq \mathrm{xor}(y,z)}} (1/2)^{\|\mathrm{xor}(y,z)\|}P(y)P(z).$$

As a generalization we consider "weighted crossover": we let $py_\ell + (1-p)z_\ell$ for $p \in [0,1]$ be the chance that the child bit x_ℓ becomes a one, where y_ℓ and z_ℓ are the corresponding bits from the parents y and z. Note that uniform crossover is the special case $p = 1/2$. We derive

$$P'(x) = \sum_{\substack{y,z \\ y\cap z \subseteq x \subseteq y\cup z}} p^{\|(z\setminus y)\setminus x\|}(1-p)^{\|x\cap(z\setminus y)\|}\, p^{\|x\cap(y\setminus z)\|}(1-p)^{\|(y\setminus z)\setminus x\|} P(y)P(z).$$

For the one-point crossover we take the sum over the crossover points of the probabilities for the corresponding prefix and the corresponding postfix:

$$P'(x) = \frac{1}{n-1}\sum_{l=1}^{n-1}\left\{ \sum_{\substack{y \\ \mathrm{pre}(y,l)=\mathrm{pre}(x,l)}} P(y) \cdot \sum_{\substack{z \\ \mathrm{po}(z,n-l)=\mathrm{po}(x,n-l)}} P(z) \right\},$$

where $\mathrm{pre}(x,l)$ denotes the prefix of length l of bit string x and $\mathrm{po}(x,l)$ denotes the postfix of length l of bit string x.

Hence we can define a number of genetic operators on distributions. If we consider the simple genetic algorithm (proportional selection followed by one-point or uniform crossover and mutation), then we can combine the definitions into a transition matrix. This matrix has a number of interesting symmetries and it is possible to numerically trace the probabilities (see [23, 25]). In the rest of the paper we will propose structures based on Fourier analysis as an alternative to these distributions. These structures seem to be better suited for calculation, especially for constructed fitness functions.

3. Fourier analysis

We start with the introduction of the basic functions. For any complex number a and subsets i and x of a finite universe U we define

$$G_{a,i}(x) = a^{\|i\setminus x\|}.$$

Here $\|s\|$ denotes the number of elements of the set s, and $i\setminus x$ consists of those elements in i that are not in x.

The so-called bit products $B_i(x)$ and Walsh products $R_i(x)$ are special cases:

$$B_i(x) = \lim_{a\downarrow 0} G_{a,i}(x) = \begin{cases} 1 & \text{if } i \subseteq x, \\ 0 & \text{otherwise,} \end{cases}$$

$$R_i(x) = G_{-1,i}(x) = (-1)^{\|i\setminus x\|}$$

for subsets i and x of U.

To obtain some more feeling for the functions $G_{a,i}$ we list some properties in the next lemma.

Lemma 1. *We have*

$$G_{ab,i}(x) = G_{a,i}(x)G_{b,i}(x); \tag{1}$$

$$G_{a,i_1\cup i_2}(x) = G_{a,i_1}(x)G_{a,i_2}(x) \qquad \text{if } i_1\cap i_2=\emptyset; \tag{2}$$

$$G_{a,i}(x_1\cup x_2) = a^{-\|i\|}G_{a,i}(x_1)G_{a,i}(x_2) \qquad \text{if } i\cap x_1\cap x_2=\emptyset; \tag{3}$$

$$a^{\|i\cap x\|} = a^{\|i\|}G_{1/a,i}(x) \qquad \text{if } a\neq 0; \tag{4}$$

$$\sum_{x\subseteq i} G_{a,i}(x) = (a+1)^{\|i\|}. \tag{5}$$

Proof. We give a one line proof of (5), using $(t+1)^\ell = \sum_{u=0}^{\ell}\binom{\ell}{u}t^u$ with $u=\|x\|$:

$$\sum_{x\subseteq i} G_{a,i}(x) = \sum_{u=0}^{\|i\|}\binom{\|i\|}{u}a^{\|i\|-u} = (a+1)^{\|i\|}.$$

The other equations are also easily proved. $\square$

The next lemma (Lemma 2) uses the following result which is an immediate consequence of Eq. (3):

$$G_{a,i}(x_1\cup x_2) = G_{a,i}(x_1) \quad \text{if } i\cap x_2=\emptyset. \tag{6}$$

From Eq. (5) we conclude (take the derivative with respect to a):

$$\sum_{x\subseteq i}\|i\backslash x\|\; G_{a,i}(x) = \|i\|\; a(a+1)^{\|i\|-1}. \tag{7}$$

This result will be used later on to determine Fourier coefficients for the one-bit mutation operator.

Eq. (4) suggests an alternative definition; sometimes the Walsh products are introduced by

$$R_i^{\text{alternative}}(x) = (-1)^{\|i\cap x\|} = (-1)^{\|i\|}R_i(x),$$

leading to equivalent formulas.

As a first example of the use of these equations we will prove the following lemma, where we use the Kronecker delta $\delta_{l,m}$ defined by

$$\delta_{l,m} = \begin{cases} 1 & \text{if } l=m,\\ 0 & \text{otherwise.}\end{cases}$$

Results may also be presented in equivalent matrix form, see [6].

Lemma 2. *We have*

$$\sum_{t\subseteq U}(-a)^{\|t\|}G_{a,l}(t)G_{a,m}(t) = (-a)^{\|l\|}(1-a)^{\|U\|}\delta_{l,m}$$

for arbitrary subsets l and m of a finite universe U and any nonzero complex number a.

Proof. Indeed, using (4), (1) and (2), (6), (5) and some calculus, we derive (with q any nonzero complex number):

$$\begin{aligned}
&\sum_{t\subseteq U} q^{\|t\|} G_{a,l}(t)G_{a,m}(t)\\
&\quad = q^{\|U\|} \sum_{t\subseteq U} G_{1/q,U}(t)G_{a,l}(t)G_{a,m}(t)\\
&\quad = q^{\|U\|} \sum_{t\subseteq U} G_{a/q,\mathrm{xor}(l,m)}(t)G_{a^2/q,\,l\cap m}(t)G_{1/q,\,U\setminus(l\cup m)}(t)\\
&\quad = q^{\|U\|} \sum_{t_1\subseteq \mathrm{xor}(l,m)} G_{a/q,\mathrm{xor}(l,m)}(t_1) \sum_{t_2\subseteq l\cap m} G_{a^2/q,\,l\cap m}(t_2) \sum_{t_3\subseteq U\setminus(l\cup m)} G_{1/q,\,U\setminus(l\cup m)}(t_3)\\
&\quad = q^{\|U\|}\,(a/q+1)^{\|\mathrm{xor}(l,m)\|}(a^2/q+1)^{\|l\cap m\|}(1/q+1)^{\|U\setminus(l\cup m)\|}\\
&\quad = (a+q)^{\|\mathrm{xor}(l,m)\|}(a^2+q)^{\|l\cap m\|}(1+q)^{\|U\setminus(l\cup m)\|}.
\end{aligned}$$

Now let $q\to -a$, and use

$$\lim_{q\to -a}(a+q)^{\|\mathrm{xor}(l,m)\|}=\delta_{l,m}$$

in order to complete the proof. □

Suppose that a function ϕ on subsets of U satisfies

$$\phi(x)=\sum_{l}\beta_l(\phi)G_{a,l}(x)$$

for all $x\subseteq U$, for certain complex numbers $\beta_l(\phi)$. Here the summation runs over all subsets l of U; in such cases we will omit U. Using Lemma 2 we get

$$\beta_l(\phi)=(-a)^{-\|l\|}(1-a)^{-\|U\|}\sum_t(-a)^{\|t\|}G_{a,l}(t)\phi(t).$$

From the formula for $\beta_l(\phi)$ we infer that the $2^{\|U\|}$ functions $G_{a,i}$ ($i\subseteq U$) are linearly independent if $a\neq 0,1$, and every function on subsets of U can be uniquely expressed as a linear combination of $G_{a,i}$'s. The $\beta_l(\phi)$ ($l\subseteq U$) are the so-called Fourier(-Walsh) coefficients of ϕ with respect to the functions $G_{a,i}$.

In the limiting case where $a\downarrow 0$, leading to the bit products, one has to be very careful. In the sequel most formulas are however easily seen to hold in this case. The functions B_i ($i\subseteq U$) are still linearly independent.

In order to perform calculations with the functions $G_{a,i}$ the following lemma, that generalizes Eq. (5), is useful. In particular, we will use it later on to calculate Fourier coefficients for the various recombination operators.

Lemma 3. *Let i be a finite set, and a,b and c complex numbers. Then*

$$\sum_{x,y\subseteq i} G_{a,i}(x\setminus y)G_{b,i}(x\cap y)G_{c,i}(y\setminus x)=(ab+bc+ca+abc)^{\|i\|}.$$

Proof. Put $u=\|x\backslash y\|, v=\|x\cap y\|$ and $w=\|y\backslash x\|$. Then the sum from the left hand side equals

$$\sum_{v=0}^{\|i\|}\sum_{u=0}^{\|i\|-v}\sum_{w=0}^{\|i\|-u-v}\frac{\|i\|!}{u!v!w!(\|i\|-u-v-w)!}a^{\|i\|-u}b^{\|i\|-v}c^{\|i\|-w}$$

$$=\sum_{v=0}^{\|i\|}\sum_{u=0}^{\|i\|-v}\frac{\|i\|!}{u!v!(\|i\|-u-v)!}a^{\|i\|-u}b^{\|i\|-v}c^{\|i\|}\sum_{w=0}^{\|i\|-u-v}\binom{\|i\|-u-v}{w}c^{-w}$$

$$=\sum_{v=0}^{\|i\|}\sum_{u=0}^{\|i\|-v}\frac{\|i\|!}{u!v!(\|i\|-u-v)!}a^{\|i\|-u}b^{\|i\|-v}(c+1)^{\|i\|-u-v}c^{u+v}$$

$$=\sum_{v=0}^{\|i\|}\frac{\|i\|!}{v!(\|i\|-v)!}a^{\|i\|}b^{\|i\|-v}(c+1)^{\|i\|-v}c^{v}\sum_{u=0}^{\|i\|-v}\binom{\|i\|-v}{u}(c/a(c+1))^{u}$$

$$=\sum_{v=0}^{\|i\|}\binom{\|i\|}{v}(c+a(c+1))^{\|i\|-v}a^{v}b^{\|i\|-v}c^{v}$$

$$=(ab+bc+ca+abc)^{\|i\|},$$

thereby proving the lemma. □

An easy application of Lemma 3, using (3) and (1), yields

$$\sum_{x,y\subseteq i}G_{a,i}(x\backslash y)G_{b,i}(x\cap y)G_{c,i}(y\backslash x)G_{d,i}(x)G_{e,i}(y)G_{f,i}(x\cup y)G_{g,i}(\mathrm{xor}(x,y))$$
$$=(abd+bce+cag+abcdefg)^{\|i\|},$$

for complex numbers a, b, c, d, e, f and g.

4. Expected values

In this section we propose the alternative structure for distributions. The structure is equivalent to distributions in the sense that distributions can be derived from the structure, and vice versa. The application of genetic operators to the structure gives rise to nice formulas.

We propose to use expected values of the functions $G_{a,i}$ for a fixed a:

$$E[G_{a,i}]=\sum_{x}G_{a,i}(x)P(x),\quad i\subseteq U.$$

Notice that $E[G_{a,\emptyset}]=1$ always holds. If $P(x)=1/2^n$ for every string x, so all strings are equally likely (remember that $\|U\|=n$), it is easy to see that

$$E[G_{a,i}]=\left(\frac{a+1}{2}\right)^{\|i\|}.$$

We can easily express the probabilities in terms of the expected values $E[G_{a,i}]$ using the Fourier coefficients of a function $\phi_y(x)$. In particular, taking

$$\phi_y(x) = \delta_{x,y} \ (x, y \subseteq U)$$

we find

$$E[\phi_y] = \sum_x \phi_y(x)P(x) = P(y),$$

while on the other hand

$$E[\phi_y] = \sum_l \beta_l(\phi_y)E[G_{a,l}]$$

with

$$\beta_l(\phi_y) = (-1)^{\|l\|+\|y\|}(1-a)^{-\|U\|}G_{a,y}(l).$$

Hence

$$P(y) = \sum_l (-1)^{\|l\|+\|y\|}(1-a)^{-\|U\|}G_{a,y}(l)E[G_{a,l}].$$

5. Tracing expected values

In this section we examine how the expected values $E[G_{a,i}]$ change under the genetic operators. By $E'[\phi]$ we denote the expected value of ϕ immediately after the genetic operation we are interested in.

As a first example, the operator that changes every set into its complement ($i \mapsto U\backslash i$, the complement operator) has the property $P'(x) = P(U\backslash x)$, whence

$$\begin{aligned} E'[G_{a,i}] &= \sum_x G_{a,i}(x)P'(x) = \sum_x G_{a,i}(x)P(U\backslash x) = \sum_x G_{a,i}(U\backslash x)P(x) \\ &= \sum_x a^{\|i\cap x\|}P(x) = a^{\|i\|}\sum_x G_{1/a,i}(x)P(x) = a^{\|i\|}E[G_{1/a,i}]. \end{aligned}$$

For proportional selection first note that $E[f]$ can be computed using the Fourier coefficients $\beta_l(f)$ of f by

$$E[f] = \sum_l \beta_l(f)E[G_{a,l}],$$

whereas $E'[G_{a,i}]$ (the expected value after proportional selection) follows from

$$E'[G_{a,i}] = \frac{1}{E[f]}\sum_l \beta_l(f\,G_{a,i})E[G_{a,l}];$$

here we used the Fourier coefficients $\beta_l(f\,G_{a,i})$ of $x \mapsto f(x)G_{a,i}(x)$.

How does mutation affect the expected value of the functions $G_{a,i}$? The answer is provided by the following theorem. (Recall that p is the mutation rate.)

Theorem 4 (Mutation). *We have*

$$E'[G_{a,i}] = \sum_{j \subseteq i} (1-2p)^{\|j\|}(p(a+1))^{\|i\setminus j\|} E[G_{a,j}].$$

Proof. We compute

$$\begin{aligned} E'[G_{a,i}] &= \sum_x G_{a,i}(x)P'(x) \\ &= \sum_x \sum_y G_{a,i}(x) p^{\|\mathrm{xor}(x,y)\|}(1-p)^{n-\|\mathrm{xor}(x,y)\|}P(y) \\ &= \sum_y \left\{ \sum_x G_{a,i}(x) p^{\|\mathrm{xor}(x,y)\|}(1-p)^{n-\|\mathrm{xor}(x,y)\|} \right\} P(y). \end{aligned}$$

Let

$$\phi(y) = \sum_x G_{a,i}(x) p^{\|\mathrm{xor}(x,y)\|}(1-p)^{n-\|\mathrm{xor}(x,y)\|};$$

note that $\phi(y)$ is the expected value of $G_{a,i}$ if $P(x) = \delta_{x,y}$.
Then, using (1), (2), (3) and (5) from Section 3, and

$$\|x\setminus y\| = n - \|y\| - \|(U\setminus y)\setminus x\|,$$

we proceed as in the example of the operator that changes every set into its complement, and see that (with $\alpha = p/(1-p)$)

$$\begin{aligned} \phi(y) &= (1-p)^n \sum_x G_{a,i}(x)\alpha^{\|y\setminus x\| + \|x\setminus y\|} \\ &= \alpha^{n-\|y\|}(1-p)^n \sum_x G_{a,i}(x)G_{\alpha,y}(x)G_{1/\alpha,U\setminus y}(x) \\ &= (a(1-p)+p)^{\|i\setminus y\|}(ap+(1-p))^{\|i\cap y\|}. \end{aligned}$$

We introduce the Fourier coefficients $\beta_l(\phi)$ of ϕ:

$$E'[G_{a,i}] = \sum_y \phi(y)P(y) = \sum_y \sum_l \beta_l(\phi)G_{a,l}(y)P(y) = \sum_l \beta_l(\phi)E[G_{a,l}].$$

And we continue, with $r = a(1-p)+p$ and $s = ap + (1-p)$:

$$\begin{aligned} \beta_l(\phi) &= \lim_{q\to -a} q^{-\|l\|}(1+q)^{-n} \sum_t q^{\|t\|} G_{a,l}(t) r^{\|i\setminus t\|} s^{\|i\cap t\|} \\ &= \lim_{q\to -a} q^{n-\|l\|}(1+q)^{-n} s^{\|i\|} \sum_t G_{1/q,U}(t)G_{a,l}(t)G_{r,i}(t)G_{s,U\setminus i}(t)G_{1/s,U}(t) \\ &= \lim_{q\to -a} q^{n-\|l\|}(1+q)^{-n} s^{\|i\|} \sum_{t_1\subseteq U\setminus(l\cup i)} G_{1/q,U\setminus(l\cup i)}(t_1) \sum_{t_2\subseteq l\setminus i} G_{a/q,l\setminus i}(t_2) \\ &\qquad \sum_{t_3\subseteq i\cap l} G_{ar/qs,i\cap l}(t_3) \sum_{t_4\subseteq i\setminus l} G_{r/qs,i\setminus l}(t_4) \\ &= \lim_{q\to -a} (1+q)^{-\|l\cup i\|}(1+a/q)^{\|l\setminus i\|}(ra/q+s)^{\|i\cap l\|}(r+qs)^{\|i\setminus l\|} \end{aligned}$$

$$= \begin{cases} (1-a)^{-\|i\|}(s-r)^{\|l\|}(r-as)^{\|i\setminus l\|} & \text{if } l\subseteq i, \\ 0 & \text{otherwise} \end{cases}$$

$$= \begin{cases} (1-2p)^{\|l\|}(p(a+1))^{\|i\setminus l\|} & \text{if } l\subseteq i, \\ 0 & \text{otherwise} \end{cases}$$

and the theorem follows. □

If we start from a distribution P with $P(x_0)=1$ for some fixed string x_0 (and all other sets/strings have probability 0), one mutation with $p=1/2$ results in

$$E'[G_{a,i}]=\left(\frac{a+1}{2}\right)^{\|i\|},$$

so afterwards all strings are equally likely — as expected.

The line of argument used in the proof applies to general one parent genetic operators. Indeed, suppose that

$$P'(x)=\sum_y I(x,y)P(y)$$

for some function $I(x,y)$, denoting the probability that parent y generates child x. Then

$$E'[G_{a,i}]=\sum_j \left\{(1-a)^{-n}\sum_{t,x}(-a)^{\|t\|-\|j\|}G_{a,j}(t)G_{a,i}(x)I(x,t)\right\}E[G_{a,j}].$$

For instance, the mutation operator that only changes zero bits (with probability p), leads to

$$E'[G_{a,i}]=\sum_{j\subseteq i}(1-p)^{\|j\|}p^{\|i\setminus j\|}E[G_{a,j}].$$

Intermediate results are

$$P'(x)=\sum_{y\subseteq x}p^{\|x\setminus y\|}(1-p)^{n-\|x\|}P(y)$$

and $\phi(y)=r^{\|i\setminus y\|}$ (cf. the proof of Theorem 4).

As a corollary of the theorem above we mention a result concerning the operator that changes every set into its complement ($i\mapsto U\setminus i$, see also above); this corresponds with $p=1$, leading to

$$E'[G_{a,i}]=\sum_{j\subseteq i}(-1)^{\|j\|}(a+1)^{\|i\setminus j\|}E[G_{a,j}].$$

Next we examine “one-bit mutation”, where exactly one bit from the parent is toggled; this bit is chosen randomly. We get:

Theorem 5 (One-bit mutation). *We have*

$$E'[G_{a,i}] = (1 - 2\,\|i\|/n)E[G_{a,i}] + (1+a)/n \sum_{\substack{j \subseteq i \\ \|i \setminus j\| = 1}} E[G_{a,j}].$$

Proof. Similar to the proof of Theorem 4, we put

$$E'[G_{a,i}] = \frac{1}{n} \sum_y \phi(y) P(y)$$

with

$$\phi(y) = \sum_{\substack{x \\ \|\mathrm{xor}(x,y)\| = 1}} G_{a,i}(x) = (n + (a-1)\,\|i\| + (1/a - a)\,\|i \setminus y\|\,) G_{a,i}(y).$$

Again we compute the Fourier coefficients $\beta_l(\phi)$ of ϕ, leaving the details to the reader:

$$\begin{aligned}
\beta_l(\phi) &= \lim_{q \to -a} q^{-\|l\|}(1+q)^{-n} \sum_t q^{\|t\|} G_{a,l}(t) \\
&\qquad (n + (a-1)\,\|i\| + (1/a - a)\,\|i \setminus t\|\,) G_{a,i}(t) \\
&= (n + (a-1)\,\|i\|)\delta_{i,l} + (1/a - a) \lim_{q \to -a} q^{-\|l\|}(1+q)^{-\|i \cup l\|}(a+q)^{\|l \setminus i\|} \\
&\qquad \Big\{ (a+q)^{\|i \setminus l\|}\,\|i \cap l\|\, a^2 (a^2+q)^{\|i \cap l\| - 1} \\
&\qquad\quad + (a^2+q)^{\|i \cap l\|}\,\|i \setminus l\|\, a(a+q)^{\|i \setminus l\| - 1} \Big\} \\
&= \begin{cases} n - 2\,\|i\| & \text{if } i = l, \\ 1 + a & \text{if } l \subseteq i \text{ and } \|i \setminus l\| = 1, \\ 0 & \text{otherwise,} \end{cases}
\end{aligned}$$

where in particular Eq. (7) from Section 3 was used. □

Now we pay attention to two-parent operators. For uniform crossover we get:

Theorem 6 (Uniform crossover). *We have*

$$E'[G_{a,i}] = (1/2)^{\|i\|} \sum_{j \subseteq i} E[G_{a,j}] E[G_{a,\, i \setminus j}].$$

Proof. We compute

$$E'[G_{a,i}] = \sum_{y,z} \sum_{\substack{x \\ y \cap z \subseteq x \subseteq y \cup z}} G_{a,i}(x)(1/2)^{\|\mathrm{xor}(y,z)\|} P(y) P(z).$$

Let

$$\psi(y,z) = (1/2)^{\|\mathrm{xor}(y,z)\|} \sum_{\substack{x \\ y \cap z \subseteq x \subseteq y \cup z}} G_{a,i}(x);$$

we continue:

$$\begin{aligned}\psi(y,z) &= (1/2)^{\|\mathrm{xor}(y,z)\|} a^{\|i\setminus(y\cup z)\|} \sum_{x'\subseteq \mathrm{xor}(y,z)} G_{a,i\cap\mathrm{xor}(y,z)}(x') \\ &= (1/2)^{\|\mathrm{xor}(y,z)\|} a^{\|i\setminus(y\cup z)\|}\, 2^{\|\mathrm{xor}(y,z)\setminus i\|} \sum_{x''\subseteq i\cap \mathrm{xor}(y,z)} G_{a,i\cap\mathrm{xor}(y,z)}(x'') \\ &= ((a+1)/2)^{\|i\cap\mathrm{xor}(y,z)\|} a^{\|i\setminus(y\cup z)\|} \\ &= ((a+1)/2)^{\|i\|} G_{2/(a+1),\,i}(\mathrm{xor}(y,z)) G_{a,i}(y\cup z).\end{aligned}$$

We now have to compute the Fourier coefficients $\beta_{l,m}(\psi)$ of ψ with respect to the functions $G_{a,l}(y)G_{a,m}(z)$ of two variables

$$\psi(y,z) = \sum_{l,m} \beta_{l,m}(\psi) G_{a,l}(y) G_{a,m}(z),$$

and then arrive at

$$E'[G_{a,i}] = \sum_{l,m} \beta_{l,m}(\psi) E[G_{a,l}] E[G_{a,m}].$$

We compute, using (1), (2), (3), (4) and Lemma 3 from Section 3:

$$\begin{aligned}\beta_{l,m}(\psi) &= \lim_{q\to -a} q^{2n-\|l\|-\|m\|}(1+q)^{-2n}((a+1)/2)^{\|i\|} \\ &\qquad \sum_{t_1,t_2} G_{1/q,U}(t_1)G_{1/q,U}(t_2)G_{a,l}(t_1)G_{a,m}(t_2) \\ &\qquad\quad G_{2/(a+1),i}(\mathrm{xor}(t_1,t_2))G_{a,i}(t_1\cup t_2) \\ &= 2^{-\|i\cap\mathrm{xor}(l,m)\|} \lim_{q\to -a} q^{-\|l\|-\|m\|}(a+q)^{\|l\setminus i\|+\|m\setminus i\|+\|i\setminus \mathrm{xor}(l,m)\|} \\ &\qquad (1+q)^{\|i\setminus(l\cup m)\|+\|\mathrm{xor}(l,m)\setminus i\|-2\|i\cup l\cup m\|} \\ &\qquad\quad (a^2+q)^{\|i\cap l\cap m\|}(q(1-a)^2+2(a+q)^2)^{\|i\cap\mathrm{xor}(l,m)\|} \\ &= \begin{cases} (1/2)^{\|i\|}\delta_{m,i\setminus l} & \text{if } l\subseteq i, \\ 0 & \text{otherwise}\end{cases}\end{aligned}$$

since $(l\setminus i)\cup(m\setminus i)\cup(i\setminus\mathrm{xor}(l,m))$ has to be the empty set in order to give a non-zero contribution. □

Notice that

$$\sum_{j\subseteq i} E[G_{a,j}]E[G_{a,i\setminus j}] = \sum_{\substack{l,m\subseteq i \\ l\cup m=i,\, l\cap m=\emptyset}} E[G_{a,l}]E[G_{a,m}],$$

showing more symmetry. Furthermore, for singletons i (sets consisting of one element) we have $E'[G_{a,i}] = E[G_{a,i}]$.

For “weighted crossover” one may proceed as above, putting $r = a(1-p)+p$ and $s = ap+(1-p)$:

$$\psi(y,z) = a^{\|i\setminus(y\cup z)\|} r^{\|i\cap(y\setminus z)\|} s^{\|i\cap(z\setminus y)\|},$$

finally leading to

$$E'[G_{a,i}] = \sum_{j\subseteq i} p^{\|j\|} E[G_{a,j}](1-p)^{\|i\setminus j\|} E[G_{a,i\setminus j}].$$

For the slightly changed uniform crossover (if the bits from the parents are equal, the child receives a zero bit) we find, analogous to ordinary uniform crossover, denoting E^{00} instead of E':

$$E^{00}[G_{a,i}] = (a-1)^{-\|i\|} \sum_{l,m\subseteq i} a^{2\|i\setminus(l\cup m)\|}(-(a+1)/2)^{\|\mathrm{xor}(l,m)\|} E[G_{a,l}]E[G_{a,m}].$$

When we let the child inherit a one if the bits from the parents coincide, we arrive at (denoting E^{11} instead of E'):

$$E^{11}[G_{a,i}] = (1-a)^{-\|i\|} \sum_{l,m\subseteq i} (-(a+1)/2)^{\|\mathrm{xor}(l,m)\|} E[G_{a,l}]E[G_{a,m}].$$

We call this operation "one-one-crossover".

When we let the child get a one if both parents have a zero, and a zero if they both have a one, direct computation yields

$$E'[G_{a,i}] = \sum_{\substack{l,m\subseteq i \\ l\cap m=\emptyset}} (-1/2)^{\|l\|+\|m\|}(a+1)^{\|i\setminus(l\cup m)\|} E[G_{a,l}]E[G_{a,m}].$$

This formula is also easily proved using Theorem 6, followed by an application of the operator that changes every set into its complement. Note that in this case the order of the two operators may be interchanged.

Finally we examine "masked crossover". We fix a subset w of U, the "mask". Given two parents, outside w we let the child x be equal to the first parent y, inside w to the second parent z: $x=(y\setminus w)\cup(z\cap w)$. The computation is straightforward, using

$$\psi(y,z) = G_{a,i}((y\setminus w)\cup(z\cap w)) = G_{a,i\setminus w}(y)G_{a,i\cap w}(z),$$

and immediately leads to

$$E'[G_{a,i}] = E[G_{a,i\setminus w}]E[G_{a,i\cap w}].$$

As a special case we mention one-point crossover, where the child inherits the first k bits from the first parent, and the last $n-k$ bits from the second parent; k is chosen randomly from $\{1,2,\ldots,n-1\}$. We easily deduce:

$$E'[G_{a,i}] = \frac{1}{n-1}\sum_{k=1}^{n-1} E[G_{a,i\cap 1^k 0^{n-k}}]\cdot E[G_{a,i\cap 0^k 1^{n-k}}].$$

Here we defined $b^k c^\ell$ as the string consisting of k b's followed by ℓ c's.

6. Walsh products

In this section we instantiate the formulas for the Walsh products: we take a equal to -1.

First, given the expected values $E[R_l]$ of Walsh products, we can retrieve the probabilities as follows:

$$P(y) = \frac{1}{2^n} \sum_l R_l(y) E[R_l] = \frac{1}{2^n} \sum_l (-1)^{\|l \wedge y\|} E[R_l].$$

For the Fourier coefficients $\beta_l(\phi)$ we have

$$\beta_l(\phi) = \frac{1}{2^n} \sum_t G_{a,l}(t)\phi(t) = \frac{1}{2^n} \sum_t (-1)^{\|l \wedge t\|} \phi(t).$$

Squaring $f(x) = \sum_l \beta_l(f) R_l(x)$ one shows that the Fourier coefficients of the measurement function $(f(x))^2$ for the Walsh products can be expressed in terms of those of f as

$$\beta_l(f^2) = \sum_i \beta_i(f)\beta_{\mathrm{xor}(i,l)}(f),$$

and hence

$$E[f^2] = \sum_{i,l} \beta_i(f)\beta_{\mathrm{xor}(i,l)}(f) E[R_l].$$

This second moment is useful for the computation of the variance $E[f^2] - E[f]^2$.

We list the results for the different operators.

- *Complement operator:*

$$E'[R_i] = (-1)^{\|i\|} E[R_i].$$

- *Proportional selection:*

it is easy to check that for the Walsh products

$$\beta_l(f R_i) = \beta_{\mathrm{xor}(l,i)}(f).$$

Hence

$$E'[R_i] = \frac{1}{E[f]} \sum_l \beta_{\mathrm{xor}(l,i)}(f) E[R_i],$$

where as usual

$$E[f] = \sum_i \beta_i(f) E[R_i].$$

- *Mutation:*

$$E'[R_i] = (1 - 2p)^{\|i\|} E[R_i].$$

Hence, if we consider the Walsh products as a basis for functions on strings with n bits, then mutation acts as a diagonal matrix.

- *One-bit mutation:*

$$E'[R_i] = (1 - 2\,\|i\|/n) E[R_i].$$

Again the Walsh products establish a diagonal matrix.

- *One-one-crossover:*

$$E^{11}[R_i]=(1/2)^{\|i\|}\sum_{j\subseteq i}E[R_j]^2.$$

Finally note that the formulas for uniform and one-point crossover can be copied, for instance that for uniform crossover:

$$E'[R_i]=(1/2)^{\|i\|}\sum_{j\subseteq i}E[R_j]E[R_{i\setminus j}].$$

7. Bit products

In this section we instantiate the formulas for the bit products: we let $a\downarrow 0$.

First, it is easy to retrieve the probabilities once we know the expected values of the bit products:

$$P(y)=\sum_{l\supseteq y}(-1)^{\|l\setminus y\|}E[B_l].$$

The Fourier coefficients of a function ϕ can be expressed as follows:

$$\begin{aligned}\beta_l(\phi)&=(-a)^{-\|l\|}(1-a)^{-\|U\|}\sum_t(-a)^{\|t\|}G_{a,l}(t)\phi(t)\\&=(1-a)^{-\|U\|}\sum_t(-1)^{\|t\|-\|l\|}a^{\|t\|-\|l\|+\|l\setminus t\|}\phi(t)\\&=(1-a)^{-\|U\|}\sum_t(-1)^{\|t\|-\|l\|}a^{\|t\setminus l\|}\phi(t).\end{aligned}$$

By taking the limit $a\downarrow 0$ we obtain

$$\beta_l(\phi)=\sum_{t\subseteq l}(-1)^{\|t\|-\|l\|}\phi(t).$$

The Fourier coefficients of $(f(x))^2$ can be expressed in terms of those of f as

$$\beta_l(f^2)=\sum_{\substack{i,j\\ i\cup j=l}}\beta_i(f)\beta_j(f),$$

and hence

$$E[f^2]=\sum_l\sum_{\substack{i,j\\ i\cup j=l}}\beta_i(f)\beta_j(f)E[B_l].$$

Next we show how the genetic operators change the expected values of bit products.

- *Proportional selection:*

$$E'[B_i]=\frac{1}{E[f]}\sum_j\beta_j(f)E[B_{i\cup j}].$$

Note that for all x we have $B_{i\cup j}(x)=B_i(x)B_j(x)$, so $E[B_{i\cup j}]=E[B_iB_j]$.

- *Mutation:*

$$E'[B_i] = \sum_{j \subseteq i} (1-2p)^{\|j\|} p^{\|i \setminus j\|} E[B_j].$$

Again the formulas for uniform and one-point crossover hold without change.

8. Schemata

As a further application of the expected values of Walsh products we derive a nonuniform Walsh-schema transform for arbitrary measurement functions. For a schema h, let $o(h)$ be the number of defined (non-$*$) elements, and let $d(h)$ be the positions of the defined elements. For example, $o(1*0*11) = 4$ and $d(1*0*11) = \{1,3,5,6\}$. The predicate $x \in h$ is true if and only if x is an instance of the schema h, that is, x can be obtained by replacing the $*$'s in the schema by zeros or ones.

In the proofs we frequently use the following properties of xor: for all $i, j \in U$:

$$i = \text{xor}(\text{xor}(i,j), j)$$

and, for any $j \in U$:

$$\{\text{xor}(i,j)\colon\ i \in U\} = U.$$

For a schema h and a subset $i \subseteq d(h)$ we define

$$R_i(h) = \prod_{k \in i} (2h_k - 1).$$

This is well-defined, because for all $k \in d(h)$ we have $h_k \in \{0,1\}$. Alternatively, one can define the extension $\text{ext}(h)$ of h, which is obtained by replacing all $*$'s by zeros. Then

$$R_i(h) = (-1)^{\|i \setminus \text{ext}(h)\|} = R_i(\text{ext}(h)),$$

thereby showing that if h happens to contain no $*$'s, the value of $R_i(h)$ coincides with the value of the Walsh product as it was defined earlier.

The value of a measurement function ϕ on a schema h is defined by

$$\phi(h) = \frac{\sum_{x \in h} \phi(x)P(x)}{P(h)},$$

where the probability of a schema is given by $P(h) = \sum_{x \in h} P(x)$. Our goal is to express $\phi(h)$ in terms of expected values of Walsh products. First note that

$$P(h) = \frac{1}{2^n} \sum_{x \in h} \sum_{i} R_i(x) E[R_i] = \frac{1}{2^{o(h)}} \sum_{i \subseteq d(h)} R_i(h) E[R_i],$$

and more general

$$\begin{aligned}
\sum_{x\in h}\phi(x)P(x) &= \frac{1}{2^n}\sum_{x\in h}\phi(x)\sum_i R_i(x)E[R_i]\\
&= \frac{1}{2^n}\sum_i E[R_i]\sum_{x\in h}\phi(x)R_i(x)\\
&= \frac{1}{2^n}\sum_i E[R_i]\sum_{x\in h}\left\{\sum_j \beta_j(\phi)R_j(x)\right\}R_i(x)\\
&= \frac{1}{2^n}\sum_i E[R_i]\sum_{x\in h}\sum_j \beta_{\mathrm{xor}(i,j)}(\phi)R_j(x)\\
&= \frac{1}{2^n}\sum_i E[R_i]\sum_j \beta_{\mathrm{xor}(i,j)}(\phi)\sum_{x\in h}R_j(x)\\
&= \frac{1}{2^n}\sum_i E[R_i]\sum_{j\subseteq d(h)}\beta_{\mathrm{xor}(i,j)}(\phi)R_j(h)2^{n-o(h)}\\
&= \frac{1}{2^{o(h)}}\sum_{j\subseteq d(h)}R_j(h)\sum_i E[R_i]\beta_{\mathrm{xor}(i,j)}(\phi).
\end{aligned}$$

Consequently, in order to trace the probability of a schema h, we only need the expected values of Walsh products consisting of defined elements of the schema. These expected values of Walsh products are weighted by constants $R_i(h)$ that only depend on the schema. Now we can derive

$$\begin{aligned}
\phi(h) &= \frac{\sum_{x\in h}\phi(x)P(x)}{P(h)}\\
&= \frac{\sum_{j\subseteq d(h)}R_j(h)\sum_i \beta_i(\phi)E[R_{\mathrm{xor}(i,j)}]}{\sum_{i\subseteq d(h)}R_i(h)E[R_i]}\\
&= \sum_{j\subseteq d(h)}\left\{\frac{\sum_i \beta_i(\phi)E[R_{\mathrm{xor}(i,j)}]}{\sum_{i\subseteq d(h)}R_i(h)E[R_i]}\right\}R_j(h)\\
&= \sum_{j\subseteq d(h)} w_j R_j(h),
\end{aligned}$$

where the definition of the w_j's is obvious.

This is the nonuniform Walsh-schema transform for arbitrary measurement functions. Next we show that the uniform Walsh-schema transform of Goldberg from [8] and the nonuniform Walsh-schema transform of Bridges and Goldberg from [4] are instances of this.

(1) Take ϕ to be the fitness function f, fix a schema h and take the following distribution:

$$P(x) = \begin{cases} 1/2^{n-o(h)} & \text{if } x\in h,\\ 0 & \text{otherwise}\end{cases}$$

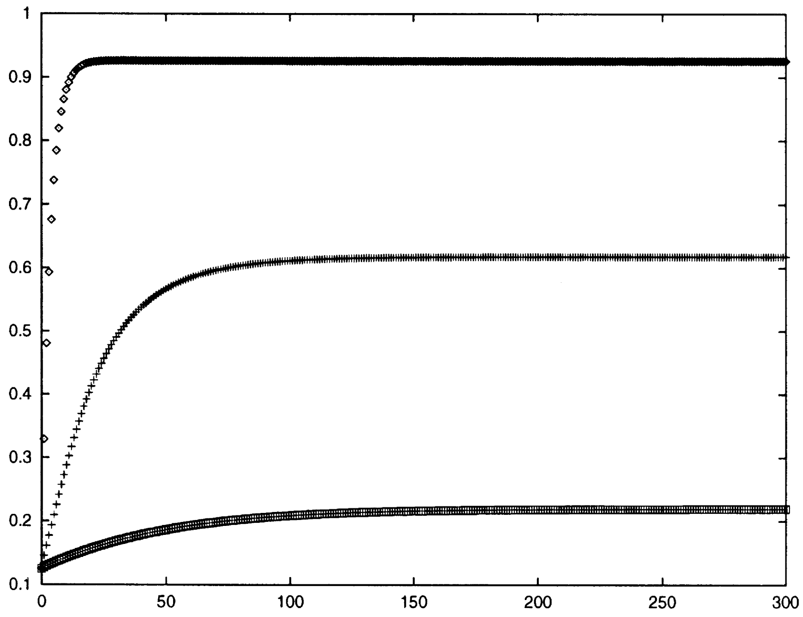

Fig. 1. Schema probabilities as a function of the number of generations, for three schemata in the case of $f(x)=x^2$.

(and hence $P(h)=1$). The $E[R_i]$ are easy to find for this distribution: $E[R_i] = R_i(h)$, if $i \subseteq d(h)$, and $E[R_i]=0$, otherwise. This gives the uniform Walsh-schema transform:

$$\begin{aligned} f(h) &= \frac{1}{2^{n-o(h)}} \sum_{x\in h} f(x) \\ &= \frac{1}{2^{n-o(h)}} \sum_{x\in h} \sum_{i} \beta_i(f) R_i(x) \\ &= \sum_{i\subseteq d(h)} \beta_i(f) R_i(h). \end{aligned}$$

(2) If we take $\phi = f$, then the nonuniform Walsh-schema transform follows directly: substitute $\beta_i(f)$ (the Fourier coefficients of the fitness function f) for the $\beta_i(\phi)$. The advantage of our formulation is that we obtain more insight in the structure of the coefficients in the transform (in [4] they are defined as the Fourier coefficients of the proportion weighted fitness function $\phi(h) = f(h)P(h)2^{o(h)}$).

In Fig. 1 we examine the fitness function $f(x)=x^2$ and three different schemata: $111******$ (top), $***111***$ (middle) and $******111$ (bottom). It shows the probability of these three schemata for the simple genetic algorithm with $n=9$, crossover rate 0.8, mutation rate 0.01 and one-point crossover. In the initial distribution all strings have equal probability.

9. Symmetric populations

In this section we impose the symmetry restriction of Rabinovich and Wigderson (see [17]). We only use bit products. A distribution is called symmetric if the following condition holds:

$$\forall i,j: \|i\| = \|j\| \Rightarrow P(i) = P(j),$$

i.e., the probability of a bit string depends only on the number of ones in the string. This symmetry condition is equivalent to the condition

$$\forall i,j: \|i\| = \|j\| \Rightarrow E[B_i] = E[B_j].$$

We can easily see that the first condition implies the second one. Clearly,

$$E[B_i] = \sum_{x \supseteq i} P(x) = \sum_{k=\|i\|}^{n} \binom{n}{k} P(2^k - 1).$$

So if $\|i\| = \|j\|$, we get

$$E[B_i] = \sum_{k=\|j\|}^{n} \binom{n}{k} P(2^k - 1) = E[B_j].$$

We now show that the second condition implies the first one. Consider a permutation Ψ that maps x to y. Then we derive

$$\begin{aligned} P(x) &= \sum_{l \supseteq x} (-1)^{\|l \setminus x\|} E[B_l] \\ &= \sum_{\Psi(l) \supseteq \Psi(x)} (-1)^{\|\Psi(l) \setminus \Psi(x)\|} E[B_{\Psi(l)}] \\ &= \sum_{l' \supseteq \Psi(x)} (-1)^{\|l' \setminus \Psi(x)\|} E[B_{l'}] \\ &= P(\Psi(x)) = P(y). \end{aligned}$$

With this symmetry restriction it is sufficient to trace only $n+1$ bit products (or in fact only n, because $E[B_\emptyset]$ is always 1):

$$E[B_i], \quad i = \emptyset, \{1\}, \{1,2\}, \ldots, \{1,2,\ldots,n\},$$

or, in integer representation,

$$E[B_{2^l-1}], \quad l = 0, \ldots, n.$$

Here every integer is obtained by interpreting the reverse of the corresponding bit string as its binary representation.

Mutation and uniform crossover preserve symmetry. This can be understood in the following way. Let Ψ be a permutation operator. Suppose $\|x\| = \|y\|$ and $P(x) = P(y)$.

• *Mutation*: Note that $\|x\| = \|y\|$ implies that there is some permutation Ψ such that $y = \Psi(x)$. By applying this permutation throughout the original mutation formula we get

$$P'(x) = \sum_{\Psi(z)} p^{\|\mathrm{xor}(y,\Psi(z))\|}(1-p)^{n-\|\mathrm{xor}(y,\Psi(z))\|}P(\Psi(z)).$$

When z runs over all possible values, $\Psi(z)$ also runs over all possible values, but in a different order. We can replace $\Psi(z)$ with z', and hence

$$P'(x) = \sum_{z'} p^{\|\mathrm{xor}(x,z')\|}(1-p)^{n-\|\mathrm{xor}(x,z')\|}P(z') = P'(y).$$

- *Uniform crossover*:

$$P'(x) = \sum_{\substack{u,z \\ u\cap z\subseteq x\subseteq u\cup z}} (1/2)^{\|\mathrm{xor}(u,z)\|}P(u)P(z).$$

Again, using the permutation trick, we write $y = \Psi(x)$ and get

$$P'(x) = \sum_{\substack{\Psi(u),\,\Psi(z) \\ \Psi(u)\cap\Psi(z)\subseteq\Psi(x)\subseteq\Psi(u)\cup\Psi(z)}} (1/2)^{\|\mathrm{xor}\,(\Psi(u),\Psi(z))\|}P(\Psi(u))P(\Psi(z)).$$

In the same way as for mutation, both $\Psi(u)$ and $\Psi(z)$ run over all possible values, so we obtain the same conclusion: $P'(x) = P'(y)$.

However, one-point crossover does not preserve symmetry in general: this is easily seen by taking the distribution with $P(000) = P(111) = 1/2$. After one-point crossover the distribution is not symmetric anymore, because $P(100) = 1/8$, but $P(010) = 0$.

In order to preserve symmetric distributions under proportional selection, we have to put a restriction on the fitness function. A necessary and sufficient condition is that we have

$$\forall i,j : \|i\| = \|j\| \Rightarrow f(i) = f(j).$$

This can be seen by noting that $P(i) = P(j)$ (with $P(i) \neq 0$) implies that

$$P'(i) = \frac{f(i)}{E[f]}P(i) = \frac{f(j)}{E[f]}P(j) = P'(j)$$
$$\Leftrightarrow\ f(i) = f(j).$$

An equivalent condition is

$$\forall i,j : \|i\| = \|j\| \Rightarrow \beta_i(f) = \beta_j(f).$$

The proof proceeds as follows. First we see that the second condition implies the first one:

$$f(i) = \sum_{k\subseteq i} \beta_k(f) = \sum_{l=0}^{\|i\|} \binom{\|i\|}{l} \beta_{2^l-1}(f)$$
$$= \sum_{l=0}^{\|j\|} \binom{\|j\|}{l} \beta_{2^l-1}(f) = \sum_{k\subseteq j} \beta_k(f) = f(j).$$

For the other direction, with a permutation Ψ that maps i to j we get

$$\beta_i(f) = \sum_{t\subseteq i} (-1)^{\|t\|-\|i\|} f(t)$$

$$= \sum_{\Psi(t)\subseteq\Psi(i)} (-1)^{\|\Psi(t)\| - \|\Psi(i)\|} f(\Psi(t))$$

$$= \sum_{t\subseteq j} (-1)^{\|t\| - \|j\|} f(t) = \beta_j(f).$$

The expected value of a measurement function ϕ can for bit products be simplified to

$$E[\phi] = \sum_{i=0}^{n} \left\{ \sum_{\substack{j \\ \|j\|=i}} \beta_j(\phi) \right\} E[B_{2^i-1}].$$

Hence, if we take $\phi(x) = f(x)$, then

$$E[f] = \sum_{i=0}^{n} \binom{n}{i} \beta_{2^i-1}(f) E[B_{2^i-1}],$$

whereas for f^2 we have

$$E[f^2] = \sum_{i=0}^{n} \binom{n}{i} \beta_{2^i-1}(f) \sum_{j=i}^{n} \binom{n-i}{n-j} E[B_{2^j-1}] \sum_{k=0}^{i} \binom{i}{k} \beta_{2^{j-k}-1}(f).$$

For the genetic operators we get the following formulas.

- *Proportional selection*:

$$E'[B_{2^i-1}] = \frac{1}{E[f]} \sum_{l} \beta_l(f) E[B_{i\cup l}]$$

$$= \frac{1}{E[f]} \sum_{j=i}^{n} E[B_{2^j-1}] \sum_{\substack{l \\ \|i\cup l\|=j}} \beta_l(f),$$

where in $i \cup l$ we consider the integer i as a set—as described above. Furthermore,

$$\sum_{\substack{l \\ \|i\cup l\|=j}} \beta_l(f) = \sum_{k=0}^{i} \beta_{2^{j-k}-1}(f) \binom{i}{k} \binom{n-i}{n-j}.$$

As a result we get

$$E'[B_{2^i-1}] = \frac{1}{E[f]} \sum_{j=i}^{n} \binom{n-i}{n-j} E[B_{2^j-1}] \sum_{k=0}^{i} \binom{i}{k} \beta_{2^{j-k}-1}(f).$$

- *Mutation*:

$$E'[B_{2^i-1}] = \sum_{j=0}^{i} \binom{i}{j} (1-2p)^j p^{i-j} E[B_{2^j-1}].$$

- *Uniform crossover*:

$$E'[B_{2^i-1}] = (1/2)^i \sum_{j=0}^{i} \binom{i}{j} E[B_{2^j-1}] E[B_{2^{i-j}-1}].$$

The advantage of using symmetric populations in case of the counting ones fitness function is that we can trace larger strings. In Figs. 2–4 we give examples of the

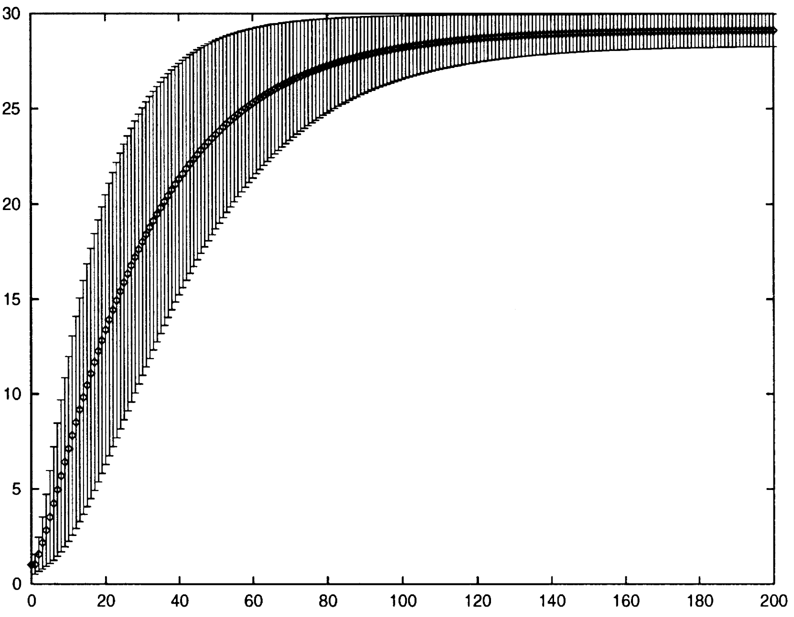

Fig. 2. Expected fitness value with error bars denoting ± 1 standard deviation in case of the simple genetic algorithm for counting ones, with $n = 30$, crossover rate 1.0, uniform crossover and mutation rate 0.001.

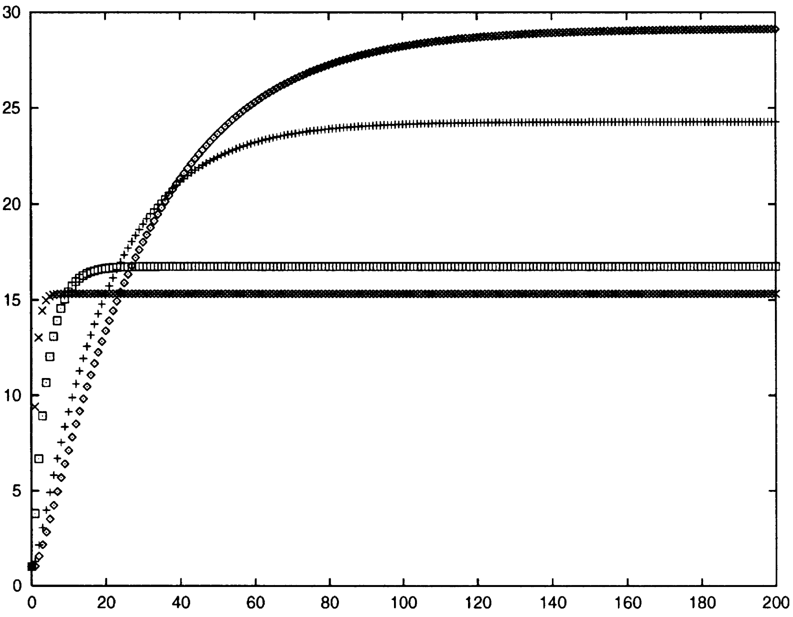

Fig. 3. Expected value of fitness in the simple genetic algorithm for counting ones, with $n = 30$, crossover rate 1.0, uniform crossover and different mutation rates (from top to bottom: 0.001, 0.01, 0.1, 0.3).

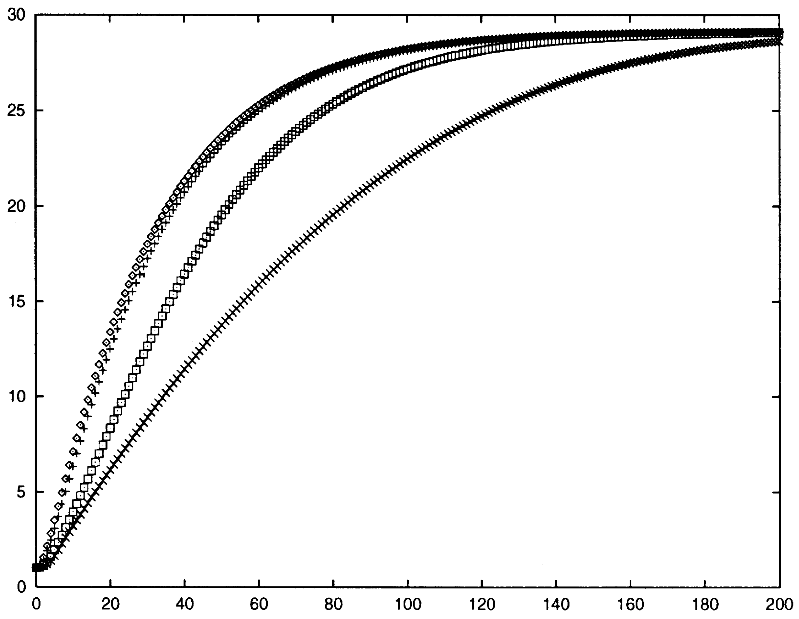

Fig. 4. Expected value of fitness in the simple genetic algorithm for counting ones, with $n = 30$, different crossover rates (from top to bottom: 1.0, 0.7, 0.1, 0.0), uniform crossover and mutation rate 0.001.

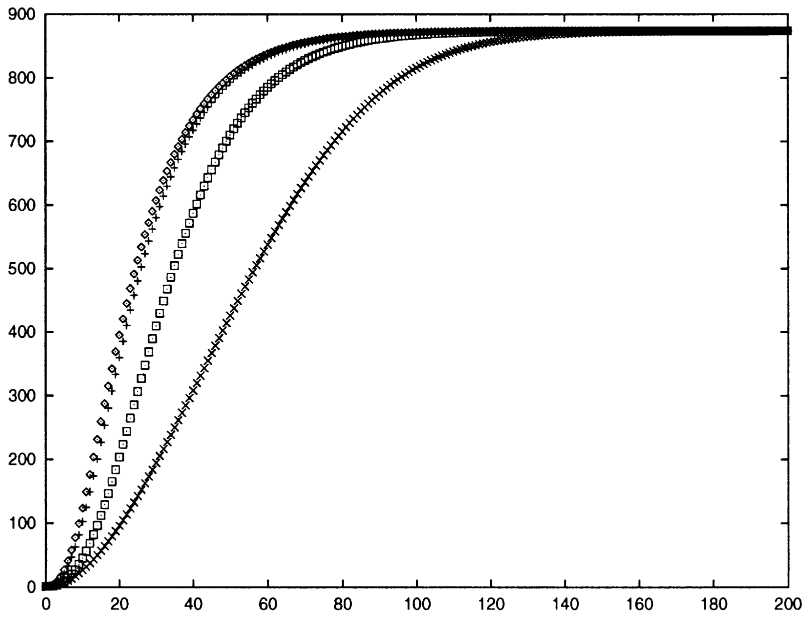

Fig. 5. Expected value of fitness in the simple genetic algorithm for the square of counting ones, with $n = 30$, different crossover rates (from top to bottom: 1.0, 0.7, 0.1, 0.0), uniform crossover and mutation rate 0.001.

numerical tracing of the simple genetic algorithm (proportional selection followed by uniform crossover and mutation) on counting ones, i.e., $f(x) = \|x\|$. Fig. 5 shows some results for the square of counting ones ($f(x) = \|x\|^2$).

10. Applications

We examine several fitness functions, and compute the quantities proposed in the previous sections. We use expected values of the functions $G_{a,i}$ in order to trace the behavior of genetic algorithms. In the case of simple fitness functions it is sometimes possible to give an analytical treatment of the behavior. If the fitness function is more complicated, one should turn to the situation of small n, and use formula manipulation or numerical approximations.

10.1. Tracing using bit and Walsh products

We consider the fitness function

$$f(x) = \tfrac{1}{5}\|x\| + 3B_{\{1,2,3\}}(x) + 3B_{\{4,5,6\}}(x) + 3B_{\{7,8,9\}}(x).$$

We take as initial distribution the one in which string 100000000 has probability one, and compare uniform and one-point crossover.

We see in Fig. 6 that the simple genetic algorithm discovers the blocks of ones step by step. We plotted the expected fitness of the simple genetic algorithm with $n = 9$, crossover rate 0.8 and mutation rate 0.01.

Next we consider the following type of deceptive fitness functions: the best string is 1100000011 with a high fitness value; for other strings x the fitness is $3(x_3 + x_4 + x_5 + x_6 + x_7 + x_8)$. In other words, for each 1-bit in a position where there is a zero in the best string, the fitness is increased by 3. The genetic algorithm is tempted to search in the direction of $**111111**$, and hence it is difficult for it to find the best string. If the fitness value of the optimum is small, the proportional selection is too weak to enforce a high enough probability for the optimum string: the mean does not approach the optimum value. On the other hand, if the fitness value is high enough, the mean approaches the optimum value. Note that the standard deviation remains high because any mutation from the optimum string results in a very different fitness value. In our experiments with this type of fitness functions it turned out that a fitness value of 45 for 1100000011 was too small, but a fitness value of 46 was enough for getting the mean to approach the optimum value. It would be interesting to give an analytical treatment of this phenomenon. See Fig. 7 in which the expected value with error bars denoting ± 1 standard deviation of fitness of a simple genetic algorithm with $n = 10$, crossover rate 0.8 and mutation rate 0.001, and one-point crossover is plotted. In the initial distribution all strings have equal probability.

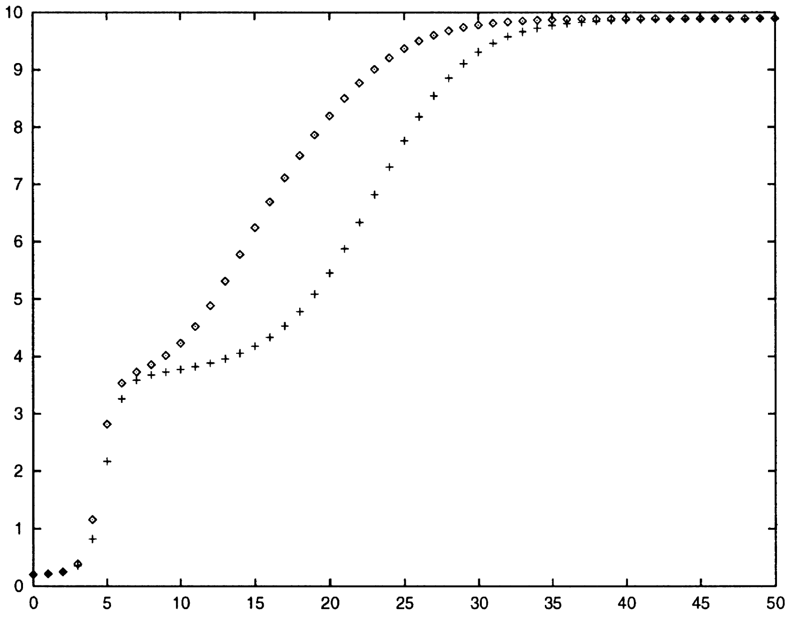

Fig. 6. Discovering blocks of ones, different crossover operators; top is uniform crossover, bottom is one-point crossover.

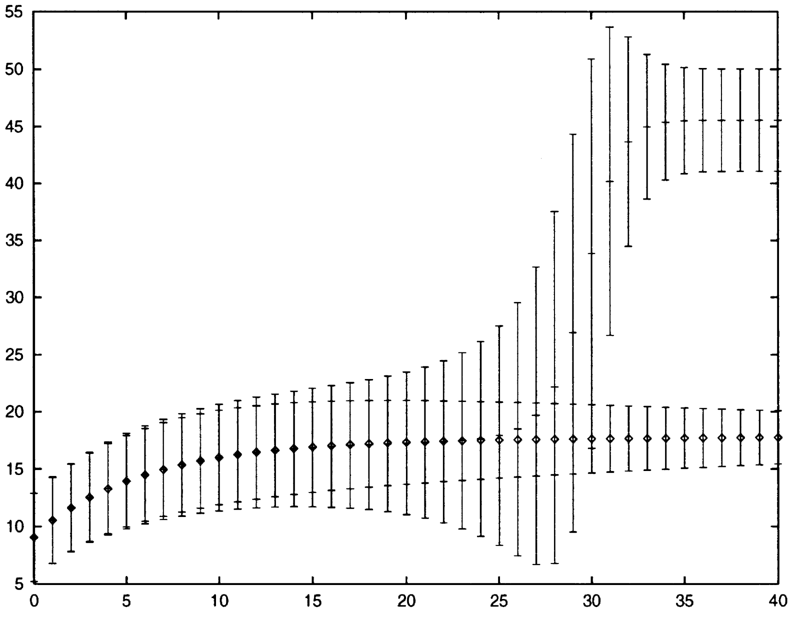

Fig. 7. Deceptive fitness function, for $f(1100000011) = 45$ (bottom) and 46 (top).

10.2. A simple fitness function

We introduce the fitness function f with $f(x) = b^{\|x\|}$, for $b > 1$ fixed. Note that the function only depends on the number of ones in string x. As an illustration we shall now a give a detailed analysis using the methods described above.

Suppose that we have a function $t(a)$ with

$$E[G_{a,i}] = t(a)^{\|i\|}$$

for all $i \subseteq U$. (As we saw before, if all strings x are equally likely, we are in this situation with $t(a) = (a+1)/2$.) In the sequel we put $t = t(a)$. We compute

$$E[f] = \left(\frac{t(b-1) + 1 - ab}{1-a} \right)^n .$$

The situation where $E[G_{a,i}] = 1$ for all i corresponds with the global maximum of f, achieved by the string $x = 11\dots1$, or rather the set U itself.

Now it is easy to describe the effect of the genetic operators. We state the results (without proof). For selection we have

$$E'[G_{a,i}] = \left(\frac{a(1-t) + b(t-a)}{1 - t + b(t-a)} \right)^{\|i\|},$$

for mutation

$$E'[G_{a,i}] = (t(1-2p) + p(a+1))^{\|i\|},$$

for uniform or masked crossover

$$E'[G_{a,i}] = E[G_{a,i}],$$

and for one-one-crossover

$$E'[G_{a,i}] = \left(\frac{t^2 - t(a+1) + 1}{1-a} \right)^{\|i\|} .$$

Note that the formula for uniform crossover implies that this operator does not have any contribution, even if it is applied to part of the population. A simple genetic algorithm, consisting of a repetition of the sequence selection, crossover and mutation, can now be analyzed. One repetition would give (with $a = -1$)

$$E'[R_i] = \left((1-2p) \frac{t - 1 + b(t+1)}{1 - t + b(t+1)} \right)^{\|i\|},$$

so only selection "increases t", i.e., gives improvement.

Using the formulas one can show that repeated selection has the following property: beginning with $t = (a+1)/2$, after $k \geqslant 0$ selection steps (denoted by a superscript (k)) we have:

$$E^{(k)}[G_{a,i}] = \left(\frac{b^k + a}{b^k + 1} \right)^{\|i\|} \to 1 \quad (k \to \infty),$$

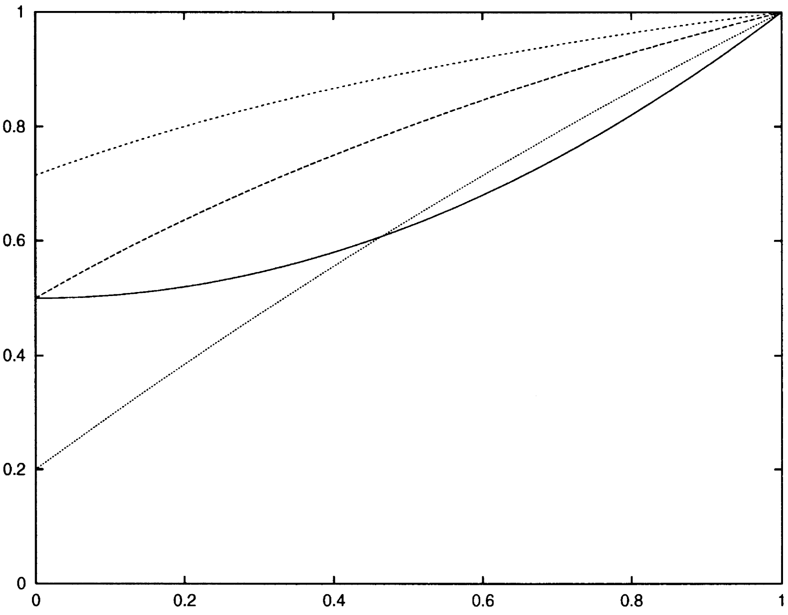

Fig. 8. Functions related to genetic operations, see text.

and, since $f = b^n G_{1/b,U}$ (or using the formula for $E[f]$ mentioned above), we infer

$$E^{(k)}[f] = \left(\frac{b^{k+1}+1}{b^k+1}\right)^n \to b^n \quad (k \to \infty).$$

In fact, it can also be shown that the distribution P after $k \geqslant 1$ selection steps (again denoted by a superscript (k)) satisfies

$$P^{(k)}(x) = \frac{b^{k\|x\|}}{(b^k+1)^n} \to \delta_{x,U} \quad (k \to \infty);$$

this distribution is symmetric, i.e., only depending on $\|x\|$. If we have $a<1$, $E^{(k)}[G_{a,i}] \uparrow 1$ for $k \to \infty$, otherwise $E^{(k)}[G_{a,i}] \downarrow 1$.

Again beginning with $t = (a+1)/2$, and using one one-one-crossover step, we obtain

$$E[G_{a,i}] = \left(\frac{a+3}{4}\right)^{\|i\|},$$

which for $b<3$ is better than the result for one selection step. If $b>3$, one selection step is superior. For $b = 3$ both steps end up in the same result.

In Fig. 8 we see plots of $t \mapsto (t^2+1)/2$ (the only concave function, related to one-one-crossover) and $t \mapsto (t(b+1)+b-1)/(t(b-1)+b+1)$ for $b = 1.5$, 3 and 6 respectively, related to selection. The graph for $b = 1.5$ starts at 0.2, for $b = 3$ it starts at 0.5, and for $b = 6$ it starts at 5/7. Here we put $a = -1$, corresponding with the Walsh products. It is also interesting to consider $t \mapsto p + (1-p)t$, corresponding with the mutation operator that only mutates zero bits.

Since all functions involved are increasing with respect to t, in this example a greedy strategy is optimal: always use the operator that produces the best one step improvement. A second one-one-crossover results in

$$E[G_{a,i}] = \left(\frac{3a+13}{16}\right)^{\|i\|}.$$

This shows that, if $13/9 < b < 3$, it is best to use one one-one-crossover, and from then on use selection. If however $b < 13/9$, a second one-one-crossover helps. Further computations show that a third one is convenient if $b < 217/169$. In general, in order to ensure a minimal number of genetic operations a finite number h of one-one-crossovers should be followed by selections if $b < 3; h$ is the minimal number of iterations $t \mapsto (t^2+1)/2$ (starting from 0) necessary to overtake $(-b+\sqrt{4b-3})/(b-1)$, the intersection of the two functions involved. In this example we see that selection on its own is capable of convergence to the optimum. Operators such as one-one-crossover and special mutations can be used to speed up the convergence rate.

10.3. *A more complicated fitness function*

As a generalization of the previous example we now define f_j for fixed $j \subseteq U$ and $b > 1$ by

$$f_j(x) = b^{\|x \cap j\| - \|x \setminus j\|};$$

taking $j = U$ results in the fitness function f defined above. Straightforward computation yields that after $k \geqslant 0$ selection steps:

$$E^{(k)}[G_{a,i}] = \left(\frac{b^k+a}{b^k+1}\right)^{\|i \cap j\|} \left(\frac{b^{-k}+a}{b^{-k}+1}\right)^{\|i \setminus j\|} \to a^{\|i \setminus j\|} \quad (k \to \infty)$$

and

$$P^{(k)}(x) = \frac{(f_j(x))^k}{(b^k+1)^{\|j\|}(b^{-k}+1)^{n-\|j\|}} \to \delta_{x,j} \quad (k \to \infty).$$

The situation where $E[G_{a,i}] = a^{\|i \setminus j\|}$ for all i corresponds with the global maximum $x = j$, but the distribution is in general not symmetric anymore.

A starting point for a deeper analysis might be the following observation. Suppose that we have $t = t(a)$ and $r = r(a)$ with

$$E[G_{a,i}] = t^{\|i \cap j\|}\, r^{\|i \setminus j\|}$$

for all $i \subseteq U$. Then crossover again satisfies

$$E'[G_{a,i}] = E[G_{a,i}].$$

10.4. *The n-queens problem*

As a further example we examine the well-known n-queens problem. Here we have to place n queens on a chessboard with n rows and n columns in such a way that no

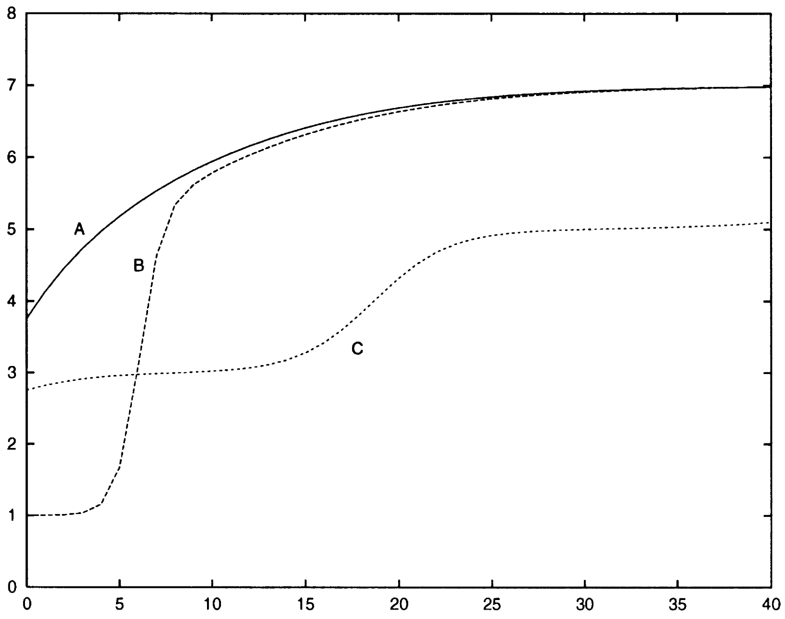

Fig. 9. Expected value of fitness in case of the 4-queens problem, for three different initial populations.

queen directly attacks another one. Arrangements of n queens on such a chessboard (with exactly one queen in every row) are easily encoded as strings with $n\lceil \lg(n)\rceil$ bits: concatenate the binary representations of the column number of these queens.

As a fitness function $Q(x)$ we propose $1 + n(n-1)/2$ minus the number of pairs of queens that attack one another. Clearly $Q(x) \in \{1, 2, \ldots, 1 + n(n-1)/2\}$. For the situation $n = 4$ we tried a simple genetic algorithm, consisting of a repetition of the triad selection-crossover-mutation. Uniform crossover was applied with probability $p_{\text{cross}} \in [0,1]$, whereas the mutation rate was $p_{\text{mut}} \in [0,1]$. The maximal value 7 of $Q(x)$ is attained for $x = 114$ (in binary 01110010) and $x = 141$ (in binary 10001101), corresponding with the two correct solutions.

The behavior of the algorithm is easily traced using the Walsh products. In Fig. 9 we plot $E[Q]$ as a function of time. Here $p_{\text{cross}} = 0.001$ and $p_{\text{mut}} = 0.0$. It appears that the initial distribution is of great importance. In situation A all strings have equal probability in the initial population, in situation B the strings $00\ldots0$ and $11\ldots1$ have probability 0.5 (and all other strings have probability 0), and in situation C all strings x with $\|x\| = 1$ have the same probability (and the other strings have probability 0).

11. Conclusions and further work

We gave a general framework for the Fourier analysis of genetic algorithms that work on distributions. We showed that the expected values of the basic functions are an interesting alternative to distributions. We derived formulas for the genetic operators, and showed that our approach has a number of interesting applications. The basic functions are especially useful to construct fitness functions, that then can be analyzed analytically or numerically, or with a package like Maple (e.g., [5]).

This gives us a way to move on from the counting ones function. As an example, local symmetry might be defined as follows. The fitness is the sum over the fitness values of the bits. For each bit, we assume a neighborhood of bits. The fitness value of the bit is determined by the number of one bits in this neighborhood. (This definition is inspired by the definition of the so-called NK-landscapes, see [2].) This kind of functions is easy to construct using the basic functions, and can be analyzed by the framework.

It is of interest to see what the general framework adds to using only expected values of Walsh products or bit products. The basic functions give us a new degree of freedom. Consider for example Theorem 4. We can choose a such that the formulas become nice: in this case take $(1-2p)=p(a+1)$, i.e., $a=(1-3p)/p$. So depending on the mutation rate we can choose our basic functions. This degree of freedom is very useful for the construction of fitness functions.

There are a number of arguments against using distributions. Due to the infinite population, elements can not get lost due to the finite sampling within a population. However, as can be seen from one of our examples (Fig. 7), there are many cases in which the expected value of the fitness does not converge to the optimum. Distributions are only a limit case of the standard genetic algorithms, and results on distributions can give only bounds on results for finite population genetic algorithms. There is however a relationship between the deterministic path of the distributions and models of genetic algorithms with finite population size motivating the tracing of distributions (cf. [15, 22, 25]).

One might still claim that distributions are too far away from the finite population genetic algorithms. An interesting “solution” is to move the genetic algorithm towards the distributions. For example, one can take a finite population, model this by a distribution (i.e., most probabilities will be zero) and compute a new distribution (or some part of it) using the formulas described in this paper, and sample this distribution to obtain a new population.

The framework we propose can be extended in several ways. A theoretical framework for the extension to larger alphabets (not only $\{0,1\}$) is studied in [13], based on earlier work of Vose. One of the goals of the present paper is to bring several existing tools together and for this we use Fourier analysis. In [13] it is shown that Fourier analysis can be used for the extension to larger alphabets. This gives further evidence that Fourier analysis can play an important role in foundational studies about genetic algorithms.

A general framework for random heuristic search is presented in [24]. It is shown that a simple genetic algorithm is an instance of this framework. Moreover, results are presented about the convergence of random heuristic search using a Lyapunov function, and different interpretations of the expected transitions between populations are discussed. It would be interesting to see whether our Fourier analysis can be applied to other instances of this framework and whether the results about convergence and interpretation can be used in our context too.

The approach we presented is just one way to work on the foundations of genetic algorithms. There is much more work on the foundations of genetic algorithms, and also in the wider field of evolutionary computation, see for example [2, 19]. There are many interesting results (also for the finite case), for example based on Markov chains, cf. [15, 18, 19].

Acknowledgements

We would like to thank Jeannette de Graaf, Adriaan Schippers and the anonymous referees for their helpful comments.

References

[1] L. Altenberg, The evolution of evolvability in genetic programming, in: K.E. Kinnear, Jr. (Ed.), Advances in Genetic Programming, MIT Press, Cambridge, MA, 1994, pp. 47–74.
[2] T. Bäck, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, Institute of Physics Publishing, Bristol, and Oxford University Press, Oxford, 1997.
[3] A.D. Bethke, Genetic algorithms as function optimizers, Ph.D. Thesis, University of Michigan, 1981.
[4] C.L. Bridges, D.E. Goldberg, The nonuniform Walsh-schema transform, in: G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1991, pp. 13–22.
[5] B.W. Char, K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan S.M. Watt, First Leaves: A Tutorial Introduction to Maple V, Springer, New York, 1992.
[6] P. Floréen, J.N. Kok, Tracing the behavior of genetic algorithms using expected values of bit and Walsh products, in: Proc. 6th Internat. Conf. on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1995, pp. 201–208.
[7] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[8] D.E. Goldberg, Genetic algorithms and Walsh functions: Part I, A gentle introduction, Complex Systems 3 (1989) 129–152.
[9] D.E. Goldberg, Genetic algorithms and Walsh functions: Part II, Deception and its analysis, Complex Systems 3 (1989) 153–171.
[10] D.E. Goldberg, Construction of high-order deceptive functions using low-order Walsh coefficients, Ann. Math. Artif. Intell. 5 (1992) 35–47.
[11] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, 1992.
[12] S. Khuri, Transform methods, in: T. Bäck, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, Institute of Physics Publishing, Bristol, and Oxford University Press, Oxford, 1997, B2.6: 1–10.
[13] G.J. Koehler, S. Bhattacharyya, M.D. Vose, General cardinality genetic algorithms, Evolutionary Computation 5 (1998) 439–459.
[14] G.E. Liepins, M.D. Vose, Polynomials, basis sets, and deceptiveness in genetic algorithms, Complex Systems 5 (1991) 45–61.
[15] A.E. Nix, M.D. Vose, Modeling genetic algorithms with Markov chains, Ann. Math. Artif. Intell. 5 (1992) 79–88.
[16] X. Qi, F. Palmieri, Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space; Part I: Basic properties of selection and mutation, IEEE Trans. Neural Networks 5 (1994) 102–119.
[17] Y. Rabinovich, A. Wigderson, An analysis of a simple genetic algorithm, in: Proc. 4th Internat. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1991, pp. 215–221.

[18] G. Rudolph, Convergence analysis of canonical genetic algorithms, IEEE Trans. Neural Networks 5 (1994) 96–101.

[19] G. Rudolph, Convergence properties of evolutionary algorithms, Ph.D. Thesis, University of Dortmund, 1996.

[20] M. Srinivas, L.M. Patnaik, Binomially distributed populations for modelling GAs, in: Proc. 5th Internat. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1993, pp. 138–145.

[21] K. Takabatake, H. Miyauchi, S. Okada, Schema spectral analysis, in: Proc. 1996 IEEE Internat. Conf. on Evolutionary Computation, IEEE, New York, 1996, pp. 176–181.

[22] M.D. Vose, Modeling simple genetic algorithms. in: L.D. Whitley (Ed.), Foundations of Genetic Algorithms, vol. 2, Morgan Kaufmann, San Mateo, CA, 1993, pp. 63–73.

[23] M.D. Vose, G.E. Liepins, Punctuated equilibria in genetic search, Complex Systems 5 (1991) 31–44.

[24] M.D. Vose, A.H. Wright, Simple genetic algorithms with linear fitness, Evolutionary Computation 2 (1995) 347–368.

[25] D. Whitley, An executable model of a simple genetic algorithm, in: L.D. Whitley (Ed.), Foundations of Genetic Algorithms, vol. 2, Morgan Kaufmann, San Mateo, CA, 1993, pp. 45–62.