Tracing the Behavior of Genetic Algorithms Using Expected Values of Bit and Walsh Products

Joost N. Kok
Department of Computer Science
Leiden University
P.O. Box 9512, 2300 RA Leiden
The Netherlands

Patrik Floréen
Department of Computer Science
University of Helsinki
P.O. Box 26, FIN-00014 University of Helsinki
Finland

Abstract

We consider two methods for tracing genetic algorithms. The first method is based on the expected values of bit products and the second method on the expected values of Walsh products. We treat proportional selection, mutation and uniform and one-point crossover. As applications, we obtain results on stable points and fitness of schemata.

1 INTRODUCTION

In this paper we introduce some methods for examining the fundamental properties of genetic algorithms (Goldberg 1989a, Holland 1992) that work on populations of bit strings, i.e., strings of a fixed length consisting of zeros and ones.

Following Vose and Liepins (1991), 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, uniform crossover, one-point crossover and mutation. 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 with probability p_c (crossover rate) produces an offspring by taking randomly with equal probability as the i-th element one of the two i-th elements in the parents. One-point crossover takes two parents and with crossover rate p_c takes a random crossover point $l \in \{1, \ldots, n-1\}$ and gives as result a bit string consisting of the l first bits of one parent and the $n-l$ last bits of the other parent. In both crossovers, if the operation is not chosen (i.e., with probability $1-p_c$), then the result is (randomly with equal probability) one of the parents. Mutation changes each element of a bit string to the opposite value with probability p_m (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. (Nix and Vose 1992, Vose 1992, Whitley 1992)

We propose two alternative structures for 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 formulae. Hence we think that *these structures are suited for performing analysis of genetic algorithms.*

The first structure is *expected values of bit products.* Given an index set and a bit string, a bit product multiplies those elements of the bit string that are in the index set. This is inspired by a paper by Rabinovich and Wigderson (1991). In that paper, the expected values of bit products are used for obtaining bounds on the rate of convergence. However, 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. On the contrary, we consider here arbitrary fitness functions and distributions, and we treat also mutation and one-point crossover.

The second structure is *expected values of Walsh products.* Walsh products 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 the literature (e.g., Bethke 1981, Goldberg 1989b, Goldberg 1989c), 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.

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.

Often one is not interested in the distribution itself, but in the population mean (expected value) of a *measurement function* ϕ (Altenberg 1994). 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):

$$\phi(x) = \begin{cases} 1 & x \in h \\ 0 & \textit{otherwise} \end{cases} .$$

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 goals of the paper is to show that *the population mean for different measurement functions can be traced using the expected values of the bit and Walsh products.*

For both methods it is important to give an expansion of the fitness function it terms of, respectively, the bit and the Walsh products. This is always possible. In a practical situation, we first calculate the bit or Walsh expansion of the fitness function and the expected values of bit or Walsh products for the initial distribution. Then we use the formulae obtained to trace the behavior over several generations. Note that we do not have to calculate the coefficients again. And as we will see, it is enough to trace the expected values of bit and Walsh products to in effect trace the distribution exactly (including, for instance, all moments of the measurement functions).

As is the case for distributions (Vose 1992), *it is possible with the expected values of bit and Walsh products analytically to find stable distributions*, i.e., distributions that do not change under a number of genetic operators.

We also consider as a special case fitness functions whose values are determined by the number of ones in the string (Goldberg 1992, Srinivas and Patnaik 1993). Consequently, the placement of the ones in the string is irrelevant for the function values, so only symmetric distributions (in which bit strings with an equal number of ones have equal probability) can be applied. This special case is of interest because the formulae 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 in the general case. In our examples illustrating our method, we have used bit strings of length 10 for the general case and bit strings of length 30 for the symmetric case.

We also 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 give some notation. Section 3 discusses distributions. Section 4 is about expected values of bit products, and in Section 5 we continue the topic restricted to symmetric distributions. Section 6 is about expected values of Walsh products. We end with a discussion.

2 NOTATION

In this section we introduce some notation and discuss different representations and operations concerning bit strings. Let ϕ be a measurement function, and let x, y, z be bit strings. A distribution of bit strings of length n associates to each bit string a probability $P(x)$, such that $\sum_x P(x) = 1$. Given a distribution, we can compute the expected value of the measurement function $E[\underline{\phi}] = \sum_x \phi(x)P(x)$.

Often, one codes a subset i of $\{1, \ldots, n\}$ by a bit string of length n: the k-th bit is 1 if and only if k is an element of i. A bit string also represents an integer by considering it as a binary number. For example,

$$\{2, 3, 5\} = \underbrace{011010\cdots0}_{n} = 22.$$

Note that in our representation, the higher order bits are to the right. Hence we have three different representations: sets, bit strings, and integers. In this paper we often switch between these representations, and the main advantage is that the different representations have different operations associated. We assume that it is always clear from the context which representation is used.

3 DISTRIBUTIONS

A distribution associates to each bit string x a probability $P(x)$. We label the generation number by a superscript: $(\cdot)^t$ denotes the value before the operation and $(\cdot)^{t+1}$ the value after the operation.

From the definitions of the operations we obtain the following formulae corresponding to the Vose and Liepins model (Vose and Liepins 1991).

Proportional selection:

$$P^{t+1}(x) = \frac{f(x)}{E[\underline{f}]^t} P^t(x).$$

Uniform crossover:

$$P^{t+1}(x) = (1-p_c)P^t(x) + p_c \sum_{y,z} P^t(y)P^t(z) \cdot$$

$$\prod_{i=1}^{n} \begin{cases} 0 & \textit{if } x_i \neq y_i \ \wedge \ x_i \neq z_i \\ 1 & \textit{if } y_i = z_i = x_i \\ \frac{1}{2} & \textit{otherwise} \end{cases} .$$

One-point crossover:

$$P^{t+1}(x) = (1-p_c)P^t(x) + \frac{p_c}{n-1} \cdot$$

$$\sum_{l=1}^{n-1} \left(\sum_{\substack{y \,:\, \textit{prefix}(y,l)= \\ \textit{prefix}(x,l)}} P^t(y) \ \cdot \sum_{\substack{z \,:\, \textit{postfix}(z,n-l)= \\ \textit{postfix}(x,n-l)}} P^t(z) \right),$$

where $\textit{prefix}(x,l)$ means the prefix of length l of bit string x and $\textit{postfix}(x,l)$ means the postfix of length l of bit string x.

Mutation:

$$P^{t+1}(x) = \sum_y p_m^{d_H(x,y)}(1-p_m)^{n-d_H(x,y)} P^t(y),$$

where $d_H(x,y)$ denotes the Hamming distance between x and y.

4 EXPECTED VALUES OF BIT PRODUCTS

We define the bit products as follows:

$$B_0(x) = 1, \text{ and}$$

$$B_i(x) = \prod_{k \in i} x_k \text{ for } i = 1, \ldots, 2^n - 1.$$

Note that a bit string x of length n contains 2^n bit products and that all bit products are either zero or one.

We can express any (measurement) function ϕ as a weighted sum over bit products:

$$\phi(x) = \sum_i \beta_i B_i(x) = \sum_{i \subseteq x} \beta_i.$$

In order to avoid confusion, from now on we use β's for coefficients of a measurement function ϕ in general, and we use b's for the coefficients of the specific measurement function which is the fitness function f.

We show how to choose coefficients β_i. Consider the matrix $\mathcal{B}_n$ of bit products of bit strings of length n, defined by $\mathcal{B}_n(i,j) = B_j(i)$, i.e.,

$$\mathcal{B}_n = \begin{pmatrix} B_0(0) & \cdots & B_{2^n-1}(0) \\ \vdots & & \vdots \\ B_0(2^n-1) & \cdots & B_{2^n-1}(2^n-1) \end{pmatrix}.$$

It can be constructed recursively as follows:

$$\mathcal{B}_0 = (1), \mathcal{B}_{k+1} = \begin{pmatrix} \mathcal{B}_k & 0 \\ \mathcal{B}_k & \mathcal{B}_k \end{pmatrix}.$$

We have

$$\begin{pmatrix} \phi(0) \\ \vdots \\ \phi(2^n-1) \end{pmatrix} = \mathcal{B}_n \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_{2^n-1} \end{pmatrix}.$$

Hence to find the coefficients from the fitness values, we have to show that $\mathcal{B}_n$ has an inverse. It is not difficult to show that the inverse is $(\mathcal{B}_n)^{-1}(i,j) = (-1)^{\|i\|+\|j\|}\mathcal{B}_n(i,j)$, where $\| \ i \ \|$ is the number of elements in the set i, i.e., the number of ones in the bit string i. Hence the coefficients are found as

$$\begin{pmatrix} \beta_0 \\ \vdots \\ \beta_{2^n-1} \end{pmatrix} = (\mathcal{B}_n)^{-1} \begin{pmatrix} \phi(0) \\ \vdots \\ \phi(2^n-1) \end{pmatrix}.$$

We are going to see how the expected values of bit products change under the genetic operators. The expected value of a bit product in a distribution is defined for every $i = 0, \ldots, 2^n-1$ as $E[\underline{B_i}] = \sum_x B_i(x)P(x) = \sum_{x \supseteq i} P(x)$, or, in matrix notation,

$$\begin{pmatrix} E[\underline{B_0}] \\ \vdots \\ E[\underline{B_{2^n-1}}] \end{pmatrix} = \mathcal{B}_n^T \begin{pmatrix} P(0) \\ \vdots \\ P(2^n-1) \end{pmatrix},$$

where $\mathcal{B}_n^T$ is the transpose of the matrix $\mathcal{B}_n$. Note that because $\mathcal{B}_n$ has an inverse, we can derive the probability distribution from the expected values of the bit products:

$$\begin{pmatrix} P(0) \\ \vdots \\ P(2^n-1) \end{pmatrix} = (\mathcal{B}_n^{-1})^T \begin{pmatrix} E[\underline{B_0}] \\ \vdots \\ E[\underline{B_{2^n-1}}] \end{pmatrix}.$$

From the expected values of bit products we can also derive the expected value $E[\underline{\phi}]$ of a measurement function on a distribution:

$$E[\underline{\phi}] = \sum_i \beta_i E[\underline{B_i}].$$

For calculating the variance or standard deviation we need the second moment $\phi'(x) = (f(x))^2$, whose coefficients β' can be expressed in terms of the coefficients of f as $\beta'_k = \sum_{i,j:i\cup j=k} b_i b_j$, and hence

$$E[\phi'] = \sum_k \sum_{i,j:i\cup j=k} b_i b_j E[\underline{B_k}].$$

Next we show how the genetic operators change the expected values of bit products. Note that for all x, $B_{i\cup j}(x) = B_i(x)B_j(x)$, so $E[\underline{B_{i\cup j}}] = E[\underline{B_i B_j}]$.

Proportional selection:

$$E[\underline{B_i}]^{t+1} = \frac{1}{E[\underline{f}]^t} \sum_j b_j E[\underline{B_{i\cup j}}]^t.$$

Uniform crossover:

$$E[\underline{B_i}]^{t+1} = (1-p_c)E[\underline{B_i}]^t +$$
$$p_c(\frac{1}{2})^{\|i\|} \sum_{j\subseteq i} E[\underline{B_j}]^t E[\underline{B_{i\setminus j}}]^t.$$

One-point crossover:

$$E[\underline{B_i}]^{t+1} = (1-p_c)E[\underline{B_i}]^t +$$
$$\frac{p_c}{n-1} \sum_{l=1}^{n-1} E[\underline{B_{i\cap\{1,\ldots,l\}}}]^t E[\underline{B_{i\cap\{l+1,\ldots,n\}}}]^t.$$

Mutation:

$$E[\underline{B_i}]^{t+1} = \sum_{j\subseteq i} (1-2p_m)^{\|j\|} p_m^{\|i\setminus j\|} E[\underline{B_j}]^t.$$

As an example of how the formulae in this paper can be derived, we show how we arrive at the formula for mutation. The idea is to consider every bit in the bit product separately, and multiply the average bit values:

$$E[\underline{B_i}]^{t+1} = \sum_x \prod_{k\in i} (p_m(1-x_k) + (1-p_m)x_k) P^t(x) =$$
$$\sum_x \prod_{k\in i} ((1-2p_m)x_k + p_m) P^t(x) =$$
$$\sum_x \sum_{j\subseteq i} (1-2p_m)^{\|j\|} B_j(x) p_m^{\|i\setminus j\|} P^t(x) =$$
$$\sum_{j\subseteq i} (1-2p_m)^{\|j\|} p_m^{\|i\setminus j\|} E[\underline{B_j}]^t.$$

5 SYMMETRIC DISTRIBUTIONS

In this section we use the symmetry restriction of Rabinovich and Wigderson (1991). A distribution is called symmetric if for all subsets i and j the following condition holds: $\| i \| = \| j \| \Rightarrow P(i) = P(j)$, 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 $\| i \| = \| j \| \Rightarrow E[\underline{B_i}] = E[\underline{B_j}]$. With the symmetry restriction it is sufficient to trace only $n+1$ bit products (or in fact, only n because $E[\underline{B_\emptyset}]$ is always 1):

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

or, in integer representation,

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

Mutation and uniform crossover preserve symmetry, but one-point crossover does not: this can be seen by taking the distribution with $P(000) = P(111) = 1/2$. After one-point crossover the distribution is nonsymmetric, 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 for all subsets i and j we have $\| i \| = \| j \| \Rightarrow f(i) = f(j)$, or, equivalently, $\| i \| = \| j \| \Rightarrow b_i = b_j$.

The expected value of a measurement function ϕ can now be simplified to

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

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

$$E[\underline{f}] = \sum_{i=0}^{n} \binom{n}{i} b_{2^i-1} E[\underline{B_{2^i-1}}].$$

and, for $\phi'(x) = (f(x))^2$, we have

$$E[\underline{\phi'}] = \sum_{i=0}^{n} \binom{n}{i} b_{2^i-1}.$$

$$\sum_{j=i}^{n} \binom{n-i}{n-j} E[\underline{B_{2^j-1}}]^t \sum_{k=0}^{i} \binom{i}{k} b_{2^{j-k}-1}.$$

Proportional selection:

$$E[\underline{B_{2^i-1}}]^{t+1} = \frac{1}{E[\underline{f}]^t}.$$

$$\sum_{j=i}^{n} \binom{n-i}{n-j} E[\underline{B_{2^j-1}}]^t \sum_{k=0}^{i} \binom{i}{k} b_{2^{j-k}-1}.$$

Uniform crossover:

$$E[\underline{B_{2^i-1}}]^{t+1} = (1-p_c)E[\underline{B_{2^i-1}}]^t +$$

$$p_c(\frac{1}{2})^i \sum_{j=0}^{i} \binom{i}{j} E[\underline{B_{2^j-1}}]^t E[\underline{B_{2^{i-j}-1}}]^t.$$

Mutation:

$$E[\underline{B_{2^i-1}}]^{t+1} = \sum_{j=0}^{i} \binom{i}{j} (1-2p_m)^j p_m^{i-j} E[B_{2^j-1}]^t.$$

In our graphs we have used a generational Simple Genetic Algorithm (SGA) (Goldberg 1989a) based on repeating the following sequence after initializing the distribution: select by proportional selection from the distribution an intermediate distribution, then perform crossover on the intermediate distribution with crossover rate p_c, and finally perform mutation with mutation rate p_m, resulting in the next distribution.

In Figure 1 we follow the expected value $E[\underline{f}]$ using the fitness function that counts the number of ones in a bit string. We start from an initial distribution in which only bit strings with one 1-bit have non-zero (equal) probability. In Figures 2 and 3 we vary the mutation and crossover rates. From the figures we see that high mutation rates are good in the beginning, when there are much 0-bits, and low mutation rates are good for the fine-tuning. We also see that for our function, higher crossover rates are better. In Figure 4 we computed with the Maple system stable expected values of bit products for different mutation rates and string lengths. We solved a system of $n+1$ equations, where the equations are obtained by replacing the $(\cdot)^{t+1}$ superscripts by $(\cdot)^t$. In this way we get exact, analytical values, from which we obtain the expected fitness values. Note that convergence to optimum is possible only if $p_m = 0$ in accordance with Rudolph (1994).

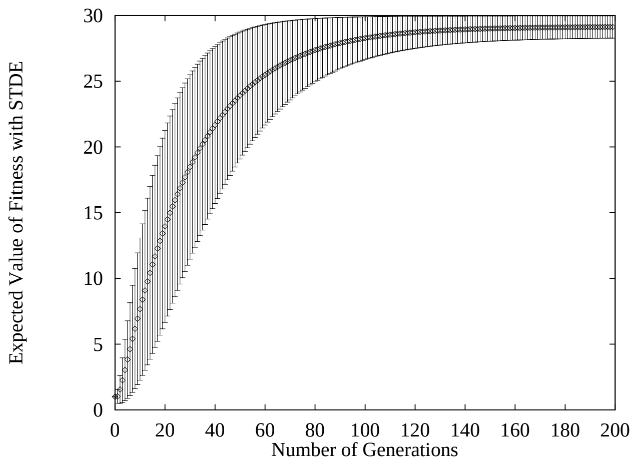

Figure 1: Expected value with error bars denoting ± 1 standard deviation of fitness of SGA with $n = 30$, $p_c = 1.0$, uniform crossover, $p_m = 0.001$, $f(x) = \| x \|$.

6 EXPECTED VALUES OF WALSH PRODUCTS

Walsh products are similar to bit products, in which 0 is changed into -1. Formally:

$$R_0(x) = 1, \text{ and}$$

$$R_i(x) = \prod_{k \in i} (2x_k - 1) \text{ for } i = 1, \ldots, 2^n - 1.$$

We can express any (measurement) function ϕ also as a weighted sum over Walsh products:

$$\phi(x) = \sum_i \rho_i R_i(x).$$

The coefficients ρ_i are called the Walsh coefficients. Consider the Walsh matrix $\mathcal{R}_n$, defined by

$$\mathcal{R}_n = \begin{pmatrix} R_0(0) & \cdots & R_{2^n-1}(0) \\ \vdots & & \vdots \\ R_0(2^n-1) & \cdots & R_{2^n-1}(2^n-1) \end{pmatrix}.$$

It can be constructed recursively as follows:

$$\mathcal{R}_0 = (1), \mathcal{R}_{k+1} = \begin{pmatrix} \mathcal{R}_k & -\mathcal{R}_k \\ \mathcal{R}_k & \mathcal{R}_k \end{pmatrix}.$$

The Walsh matrices are similar to so-called Hadamard matrices. The main difference is in the order of rows and columns. We have

$$\begin{pmatrix} \phi(0) \\ \vdots \\ \phi(2^n-1) \end{pmatrix} = \mathcal{R}_n \begin{pmatrix} \rho_0 \\ \vdots \\ \rho_{2^n-1} \end{pmatrix}.$$

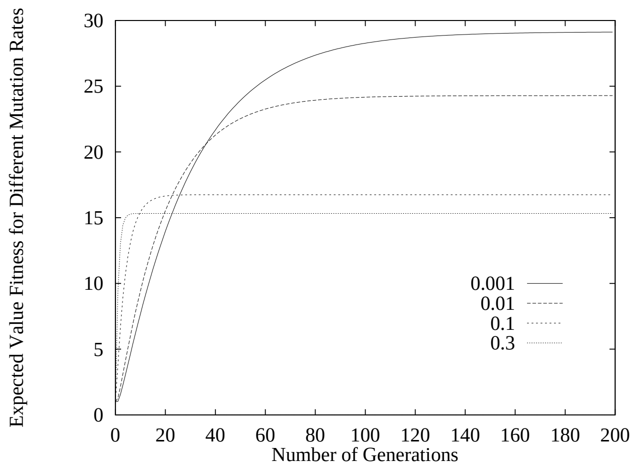

Figure 2: Expected value of fitness in SGA with $n = 30$, $p_c = 1.0$, uniform crossover, different mutation rates, $f(x) = \| x \|$.

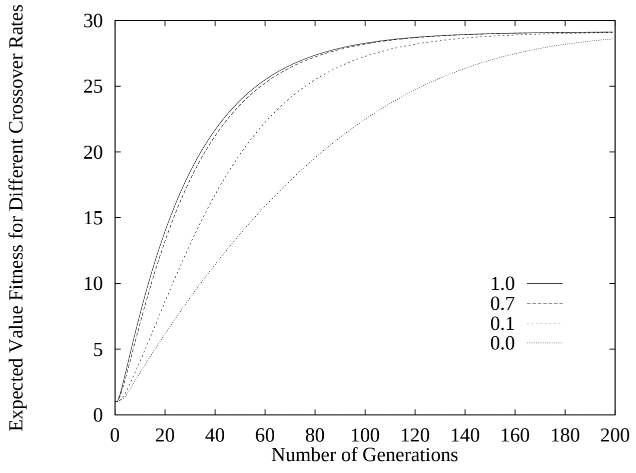

Figure 3: Expected value of fitness in SGA with $n = 30$, different crossover rates, uniform crossover, $p_m = 0.001$, $f(x) = \| x \|$.

It is not difficult to show that $\mathcal{R}_n$ has the inverse $(\mathcal{R}_n)^{-1} = 2^{-n}\mathcal{R}_n^T$. Hence

$$\begin{pmatrix} \rho_0 \\ \vdots \\ \rho_{2^n-1} \end{pmatrix} = \frac{1}{2^n}\mathcal{R}_n^T \begin{pmatrix} \phi(0) \\ \vdots \\ \phi(2^n-1) \end{pmatrix}.$$

We are going to trace the expected values of Walsh products. The expected value of a Walsh product is defined for every $i = 0, \ldots, 2^n - 1$ as $E[\underline{R_i}] = \sum_x R_i(x)P(x)$, or, in matrix notation,

$$\begin{pmatrix} E[\underline{R_0}] \\ \vdots \\ E[\underline{R_{2^n-1}}] \end{pmatrix} = \mathcal{R}_n^T \begin{pmatrix} P(0) \\ \vdots \\ P(2^n-1) \end{pmatrix}.$$

Hence, given the expected values of the Walsh products, we can retrieve the probability distribution:

$$\begin{pmatrix} P(0) \\ \vdots \\ P(2^n-1) \end{pmatrix} = \frac{1}{2^n}\mathcal{R}_n \begin{pmatrix} E[\underline{R_0}] \\ \vdots \\ E[\underline{R_{2^n-1}}] \end{pmatrix}.$$

From the expected values of Walsh products we can derive the expected value $E[\underline{\phi}]$ of a measurement function on a distribution:

$$E[\underline{\phi}] = \sum_i \rho_i E[\underline{R_i}].$$

(We will denote Walsh coefficients of the fitness function f by r's.) Again, the Walsh coefficients of the measurement function $\phi'(x) = (f(x))^2$ can be expressed in terms of the Walsh coefficients of f as $r'_j = \sum_i r_i r_{xor(i,j)}$, and hence

$$E[\underline{\phi'}] = \sum_{i,j} r_i r_{xor(i,j)} E[\underline{R_j}].$$

Here $xor(i,j)$ is an operation on bit strings that applies xor bitwise. In terms of sets $xor(i,j) = (i \cup j) \setminus (i \cap j)$.

Proportional selection:

$$E[\underline{R_i}]^{t+1} = \frac{1}{E[\underline{f}]^t} \sum_j r_{xor(i,j)} E[\underline{R_j}].$$

Uniform crossover:

$$E[\underline{R_i}]^{t+1} = (1-p_c)E[\underline{R_i}]^t +$$
$$p_c(\frac{1}{2})^{\|i\|} \sum_{j \subseteq i} E[\underline{R_j}]^t E[\underline{R_{i \setminus j}}]^t.$$

One-point crossover:

$$E[\underline{R_i}]^{t+1} = (1-p_c)E[\underline{R_i}]^t +$$
$$\frac{p_c}{n-1} \sum_{l=1}^{n-1} E[\underline{R_{i\cap\{1,\ldots,l\}}}]^t E[\underline{R_{i\cap\{l+1,\ldots,n\}}}]^t.$$

Mutation:

$$E[\underline{R_i}]^{t+1} = (1-2p_m)^{\|i\|} E[\underline{R_i}]^t.$$

From the formulae it is not difficult to see that we can exchange the order of mutation and crossover: it does

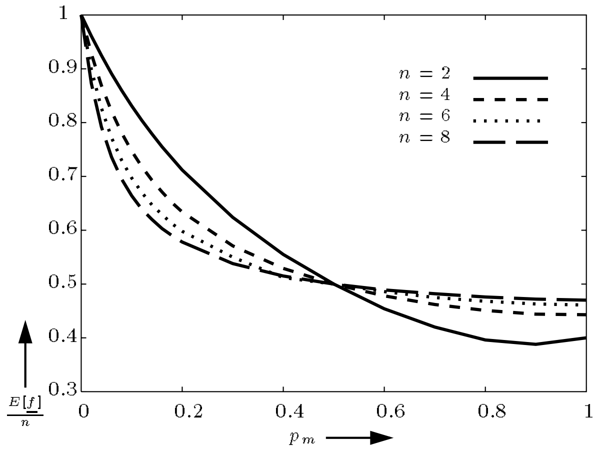

Figure 4: Expected fitness values in stable distributions for SGA with different string lengths, different mutation rates, $p_c = 1.0$, uniform crossover, $f(x) = \parallel x \parallel$.

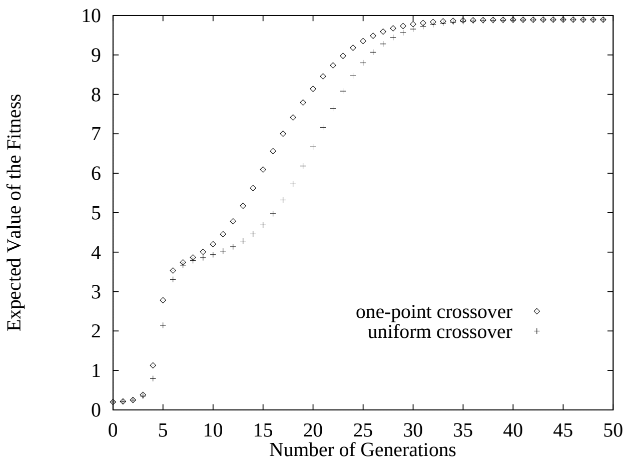

Figure 5: Expected fitness of SGA with $n = 9$, $p_c = 0.8$, $p_m = 0.01$, and f as described in the text. The initial distribution consists of only the string 100000000.

not matter for infinite populations whether we do first mutation and then crossover, or the other way around.

For symmetric distributions, we can also derive corresponding formulae.

Using the Walsh products we can trace the expected value and the variance of the fitness. In Figure 5 we took fitness function $f(x) = \frac{1}{5} \parallel x \parallel + 3B_{\{1,2,3\}}(x) + 3B_{\{4,5,6\}}(x) + 3B_{\{7,8,9\}}(x)$ and an initial distribution in which string 100000000 has probability one and compared the two crossover operations. We see that the SGA discovers the blocks of ones step by step.

In Figure 6 we took 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 73 for 0011111100 was too small, but a fitness value of 74 was enough for getting the mean to approach the optimum value.

As an 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 $d(1*0*11) = \{1, 3, 5, 6\}$.

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

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

This is defined, because for all $k \in d(h)$ we have $h_k \in \{0, 1\}$.

The value of a measurement function 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 defined by $P(h) = \sum_{x \in h} P(x)$. Our goal is to express $\phi(h)$ in terms of expected values of Walsh products. From

$$\sum_{x \in h} \phi(x) P(x) = \frac{1}{2^{o(h)}} \sum_{j \subseteq d(h)} R_j(h) \sum_i E[\underline{R_i}] \rho_{xor(i,j)}$$

we get $P(h)$ as a special case by taking $\phi(x) = 1$, if $x \in h$, and $\phi(x) = 0$, otherwise. Then we have $\rho_i = R_i(h)/2^{o(h)}$, if $i \subseteq d(h)$, and $\rho_i = 0$, otherwise. Hence

$$P(h) = \frac{1}{2^{o(h)}} \sum_{i \subseteq d(h)} R_i(h) E[\underline{R_i}].$$

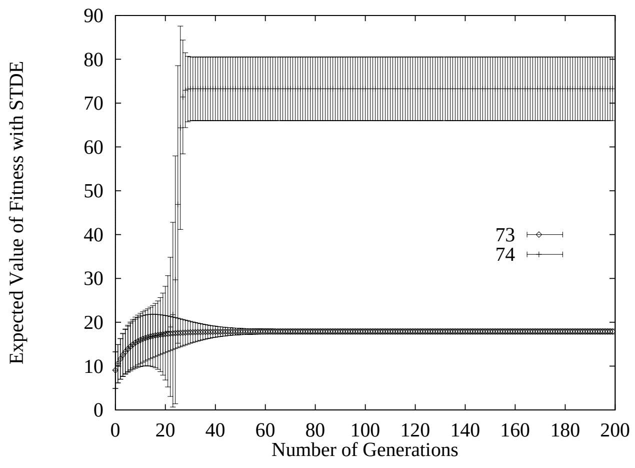

Figure 6: Expected value with error bars denoting ± 1 standard deviation of fitness of SGA with $n = 10$, $p_c = 0.8$, $p_m = 0.001$, one-point crossover, and f deceptive fitness functions described in the text with optimum fitness of 73 and 74. The initial distribution consists of all strings having equal probability.

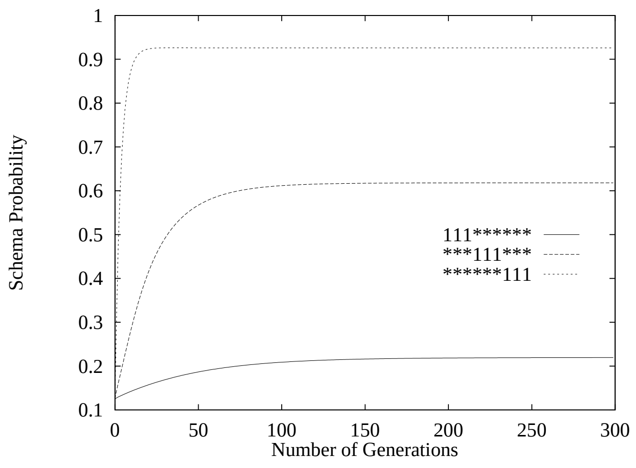

Figure 7: Probability of three different schemata for SGA with $n = 9$, p_c=0.8, p_m=0.01, one-point crossover, and $f(x) = x^2$. The initial distribution consists of all strings having equal probability.

Consequently, in order to trace the probability of a schema h, we need only the expected values of Walsh products consisting of defined elements of the schema. The expected values of Walsh products are weighted by constants $R_i(h)$ that only depend on the schema. In Figure 7 we took the fitness function $f(x) = x^2$ and three different schemata.

Now we can derive

$$\phi(h) = \sum_i \rho_i \left(\frac{\sum_{j \subseteq d(h)} R_j(h) E[\underline{R_{xor(i,j)}}]}{\sum_{j \subseteq d(h)} R_j(h) E[\underline{R_j}]} \right) =$$

$$\sum_{j \subseteq d(h)} \left(\frac{\sum_i \rho_i E[\underline{R_{xor(i,j)}}]}{\sum_{l \subseteq d(h)} R_l(h) E[\underline{R_l}]} \right) R_j(h) = \sum_{j \subseteq d(h)} w_j R_j(h).$$

This is the nonuniform Walsh-schema transform for arbitrary measurement functions. Next we show that the uniform Walsh-schema transform of Goldberg (1989b) and the nonuniform Walsh-schema transform of Bridges and Goldberg (1991) are instances of this.

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

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

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

$$f(h) = \sum_{i \subseteq d(h)} r_i R_i(h).$$

2. If we take $\phi = f$, then the nonuniform Walsh-schema transform follows directly: substitute r_i (the Walsh coefficients of the fitness function f) for the ρ_i. The advantage of our formulation is that we get more insight in the structure of the coefficients in the transform (in Bridges and Goldberg (1991) they are defined as the Walsh coefficients of the proportion weighted fitness function $\phi(h) = f(h)P(h)2^{o(h)}$).

7 DISCUSSION

We gave two methods for tracing measurement functions describing properties of genetic algorithms. We hope that these methods can give new tools for algorithmic analysis of genetic algorithms. The methods can be extended to other selection procedures and other genetic operations such as multi-parent crossover, or for instance, to genetic algorithms with a dynamically changing mutation rate.

Future work includes an analysis of our matrices for establishing bounds on the convergence times and for obtaining more insight in stable distributions.

Acknowledgements

The authors would like to thank M. Rasch for contributions to the work. The authors would also like to thank the anonymous referees and T. Bäck for many helpful comments.

References

L. Altenberg (1994). The evolution of evolvability in genetic programming. In K. E. Kinnear Jr (ed.), *Advances in Genetic Programming*, 47-74.

A. D. Bethke (1981). *Genetic Algorithms as Function Optimizers.* PhD thesis, University of Michigan. *Dissertation Abstracts International* **41**(9), 3503B, University Microfilms No. 8106101.

C. L. Bridges and D. E. Goldberg (1991). The nonuniform Walsh-schema transform. In G. E. Rawlins (ed.), *Foundations of Genetic Algorithms (FOGA)*, 13-22.

D. E. Goldberg (1989a). *Genetic Algorithms in Search, Optimization & Machine Learning.* Reading, Mass.: Addison-Wesley.

D. E. Goldberg (1989b). Genetic algorithms and Walsh functions: Part I, A gentle introduction. *Complex Systems* **3**(2):129-152.

D. E. Goldberg (1989c). Genetic algorithms and Walsh functions: Part II, Deception and its analysis. *Complex Systems* **3**(2):153-171.

D. E. Goldberg (1992). Construction of high-order deceptive functions using low-order Walsh coefficients. *Annals of Mathematics and Artificial Intelligence* **5**(1):35-48.

J. H. Holland (1992). *Adaptation in Natural and Artificial Systems.* Cambridge, Mass.: MIT Press.

A. E. Nix and M. D. Vose (1992). Modeling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence* **5**(1):79-88.

Y. Rabinovich and A. Wigderson (1991). An analysis of a simple genetic algorithm. In *Proc. 4th International Conference on Genetic Algorithms*, 215-221.

G. Rudolph (1994). Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks* **5**(1):96-101.

M. Srinivas and L. M. Patnaik (1993). Binomially distributed populations for modelling GAs. In *Proc. 5th International Conference on Genetic Algorithms (ICGA93)*, 138-145.

M. D. Vose (1992). Modeling simple genetic algorithms. In D. Whitley (ed.), *Foundations of Genetic Algorithms 2 (FOGA2)*, 63-73.

M. D. Vose and G. E. Liepins (1991). Punctuated equilibria in genetic search. *Complex Systems* **5**(1):31-44.

D. Whitley (1992). An executable model of a simple genetic algorithm. In D. Whitley (ed.), *Foundations of Genetic Algorithms 2 (FOGA2)*, 45-62.