Proportionate Prediction With Respect to Mutation
An extension on Hollands Schema Theorem

Ethan J. Eldridge

November 24, 2011

1 Introduction

This small paper is a snippet of something I turned in for my Evolutionary Computation Class in the Fall of 2011, this idea of N-Bit Crossover and the concepts of complements is entirely my own as far as I know. The first difference equations for crossover were worked out in Class by my professor Maggie Eppstein, and were (I believe) originally from Chapter 2 of Genetic Algorithms Revisited: Mathematical Foundations by Goldberg. I was intrigued by Schema Theorem and submitted the material in this short paper as a sort of shout out to make me, an undergraduate, stand out amongst the graduate students taking the course. A basic knowlegde of Schema Theory most definately helps in reading this paper. Wikipedia has a sparse, but rather nice article on the subject and numerous other websites offer resources on the subject as well. Without further ado, read on!

2 Part a: Extension of Difference Equations example

For a two bit schema, the following four difference equations account for the destructive and constructive affects of crossover:

$$P_{11}^{t+1} = P_{11}^{t} \cdot \frac{f_{11}}{\bar{f}} \left(1 - P_c^{'} \frac{f_{00}}{\bar{f}} P_{00}^{t}\right) + P_c^{'} \frac{f_{01} f_{10}}{\bar{f}^2} P_{01}^{t} P_{10}^{t} \tag{1}$$

$$P_{10}^{t+1} = P_{10}^{t} \cdot \frac{f_{10}}{\bar{f}} \left(1 - P_c^{'} \frac{f_{01}}{\bar{f}} P_{01}^{t}\right) + P_c^{'} \frac{f_{00} f_{11}}{\bar{f}^2} P_{00}^{t} P_{11}^{t} \tag{2}$$

$$P_{01}^{t+1} = P_{01}^{t} \cdot \frac{f_{01}}{\bar{f}} \left(1 - P_c^{'} \frac{f_{10}}{\bar{f}} P_{10}^{t}\right) + P_c^{'} \frac{f_{00} f_{11}}{\bar{f}^2} P_{00}^{t} P_{11}^{t} \tag{3}$$

$$P_{00}^{t+1} = P_{00}^{t} \cdot \frac{f_{00}}{\bar{f}} \left(1 - P_c^{'} \frac{f_{11}}{\bar{f}} P_{11}^{t}\right) + P_c^{'} \frac{f_{01} f_{10}}{\bar{f}^2} P_{01}^{t} P_{10}^{t} \tag{4}$$

P_{**}^{t} Stands for the proportion at any a time snapshot t for a schema **
$\bar{f}$ stands for average fitness (Hollands theorem uses Fitness Proportionate Selection)
f_{**} stands for the fitness of a schema
$P_c^{'}$ stands for the probability of crossover

These four equations do not account for the destructive or constructive forms of mutation, let us extend them one step at a time. Firstly, Let us consider how the schema can change,Table 2 shows this aptly. The * indicates that the specified locus is not changed by mutation and the capital M indicates a change in mutation.

	00	01	10	11
$**$	00	01	10	11
$*M$	01	00	11	10
$M*$	10	11	00	01
MM	11	10	01	00

Table 2
Mutation Yield Matrix

The survival of any schema is equal to $(1-P_m)^{o(H)}$[1], in this case, $o(H) = 2$ so we from now on we'll just write $(1-P_m)^2$. This becomes the following equation

$$P_{11}^{t+1} = P_{11}^t(1-P_m)^2 \tag{5}$$

moving onto the constructive properties of mutation, we must consult Table 2 to see that it is not as simple as the survival of the entire schema. For the case where mutation affects both loci (MM) we can see that this is the converse of survival and therefore is $(P_m)^{o(H)} = (P_m)^2$.[2] Lastly, we must take into consideration the two middle rows of Table 2, these two display the probability of one allele staying the same and the other mutating. To compute this probability, we combine the survival term $(1-P_m)$ with the destruction term P_m to attain $(P_m(1-P_m)) = P_m - P_m^2$, with these probabilities define we can now extend the equations into the following:

$$P_{11}^{t+1} = P_{11}^t(1-P_m)^2 + P_{00}^t(P_m)^2 + P_{01}^t(P_m - P_m^2) + P_{10}^t(P_m - P_m^2) \tag{6}$$
$$P_{10}^{t+1} = P_{10}^t(1-P_m)^2 + P_{01}^t(P_m)^2 + P_{00}^t(P_m - P_m^2) + P_{11}^t(P_m - P_m^2) \tag{7}$$
$$P_{01}^{t+1} = P_{01}^t(1-P_m)^2 + P_{10}^t(P_m)^2 + P_{11}^t(P_m - P_m^2) + P_{00}^t(P_m - P_m^2) \tag{8}$$
$$P_{00}^{t+1} = P_{00}^t(1-P_m)^2 + P_{11}^t(P_m)^2 + P_{01}^t(P_m - P_m^2) + P_{10}^t(P_m - P_m^2) \tag{9}$$

We can generalize this two bit expression using a concept I've termed as a half compliment, which is essentially the middle of rows of Table 2, or when only a single locus is mutated. Our general expression using half and full compliments of a two bit schema is

$$P_{**}^{t+1} = P_{**}^t(1-P_m)^2 + P_{MM}^t(P_m)^2 + P_{M*}^t(P_m - P_m^2) + P_{*M}^t(P_m - P_m^2) \tag{10}$$

Which can be simplified to

$$P_{**}^{t+1} = P_{**}^t(1-P_m)^2 + P_{MM}^t(P_m)^2 + (P_m - P_m^2)(P_{M*}^t + P_{*M}^t) \tag{11}$$

2.1 n-Bit mutation equations

Equations for other length representations could be easily devised using the same logic used to create equations such as 11 for any n-bit problems. We could work this equation as a multiplication of two vectors, one containing the the terms $(A+B)^{o(H)}$ where A represents $(1-P_m)$ and B represents P_m, this could then be expanded out without grouping terms, with each term stored in one element of the vector. Then, the other vector would contain the number of schema for every possible schema. To find the terms the following equation would suffice

$$\sum_{i=0,s}^{o(H)} P_{s+i} M_{i_{10}} \tag{12}$$

Where P_{s+i} stands for the Proportion of the schema matching the i^{th} term in M, and $M_{i_{10}}$ stands for the i^{th} term of the expansion of the binomial $(A+B)^{o(H)}$. This simple method can be used to build up any equation for any schema of n size. One of the tricky parts is the mutation vector, this is more difficult because instead of just being able to count upwards from 0 with i, we must start from whatever schema we're finding the equation for and then add on i with some care, this is discussed more below. In matlab,

```
f=1;
for i=0:n
     j=0:nchoosek(n,i)-1
          M(1,f)=(A^(n-i)*(B^i));
          f=f+1;
     end
end
M = circshift(M,[0 bin2dec(num2str((2.^S-1)));]);
P = Pop.*M;
```

Where M is a one dimensional matrix that will contain the terms of the binomial expansion and P is the returned terms of the difference equation for the mutation prediction, so summing that list amounts to the number of that schema at time $t+1$. The function takes the number of each schema as an argument and converts it into a proportion, as well as which schema to compute the equation for. In the matlab code, this schema is the variable `S` and is a matrix of length n. Instead of attempting to match the members to their respective mutation value, we can simplify things greatly by

[1] logically because the survival of a schema means that mutation did not disrupt any loci, so the probability of not disrupting a loci is $(1-P_m)$, and because each loci is independent of another, the number of fixed bits in the schema is the number of times we apply the chances of not disrupting the schema, hence $(1-P_m)^{o(H)}$.

[2] Note this is NOT $1-(1-P_m)^2$ because the probability of mutation for each loci is independent of each other, and therefore is $(1-(1-P_m))^2$ if we were to represent it as the converse of survival.

simply shifting the mutation vector by the schema's decimal value. Because the expansion terms were computed starting from 0 to n, then by shifting by some i we end up with the terms matching their respective probabilities. Because the shift is circular, the complementary term ends up exactly where it needs to be. Using this implementation, an equation representing a proportion of a schema after mutation can be created easily. The code for this implementation is on the next page.

3 Code for N-Bit Mutation in Matlab

```
%n bit mutation predictor by Ethan Eldridge
function [summation,terms] =  nBitMutPred(Pop,S,Pm)
% P is the matrix containing the number of each of the schemas (fully
% defined so L = o(H), this matrix should be 2^n long
% S is the schema you are finding the equation for

%Binomial Expansion of (A+B)^n
n = length(S);
A = (1-Pm);
B = Pm;
M = zeros(1,2^n); %M has the same number of terms as schemata we have
P = zeros(1,2^n);
%to compute the terms of the expansion, we use a loop with  variable to
%raise A and B to the right powers, and then a nested loop that uses the
%binomial coefficient as it's index to store the terms in the right place
%in the vector

f=1;
for i=0:n
    for j=0:nchoosek(n,i)-1
        M(1,f)=(A^(n-i))*(B^i);
        f=f+1;
    end
end

%Shift the mutation vector to have the proper schema at it's start point
%that big expression just changes the binary schema into its decimal value
M = circshift(M,[0 bin2dec(num2str((2.^S-1)));]);
%Change Population numbers into actual frequences
Pop = Pop/length(Pop);

%next use the passed in Pops and mutliply to get the terms of the
%difference equation

terms = Pop.*M;
summation = sum(terms);
```