

# Alternative Fitness Functions and Their Effect on the Evolution of Hierarchically-Related Individuals

Rebecca J. Parsons and Tiffani L. Williams  
University of Central Florida  
Department of Computer Science  
Orlando, FL 32816-2362  
{rebecca,williams}@cs.ucf.edu

## Abstract

*The dynamics of genetic algorithms are often used to evolve a single individual that solves a particular problem. However, these same dynamics generally defeat attempts to evolve a cooperating set of individuals that jointly address a problem. This paper describes experiments performed on alternate fitness functions that demonstrate the feasibility of evolving a set of hierarchically-related individuals, each addressing a different aspect of the posed problem. Of particular importance is the ability of the genetic algorithm to evolve these individuals with no a priori information on either the number of levels in the hierarchy or the number of individuals in the hierarchy.*

## 1. When one answer is not enough

Genetic algorithms used as optimizers evolve a population of individuals towards convergence at one individual that represents an answer to the posed problem [6, 2]. The theories about genetic algorithms clearly articulate that the dynamics of the genetic algorithm are well-suited to this task [6, 4]. Unfortunately, there are numerous situations when a single answer is undesirable. A biological example of this need for diversity is the human immune system. Our immune system is an excellent example of a system that is capable of evolving a dynamic set of individuals (antibodies) that cooperate to attain the goal of protecting the body.

This work explores the question of evolving a cooperating set of individuals using a simplified immune system as the exploratory system. We begin with a description of Forrest *et al.*'s work on a genetic algorithm model of immune system components [3], which motivates our work. This section also defines the particular problem being addressed in this work. Section 3 presents our genetic algorithm, including the specifics of the fitness functions used. Section 4

describes our preliminary results. The implications of these results are explored in Section 5. We conclude with a description of the further planned experiments.

## 2. Genetic algorithms and the immune system

The job of the human immune system is to protect the body from foreign matter, called antigens. The immune system contains antibodies that recognize and bind to antigens. There are myriad complexities involved in creating and modifying antibodies to combat a particular invasion.

Forrest *et al.* described two different aspects of the evolution demonstrated by the immune system [3]. Specifically, they first explored how the immune system might evolve *generalist* antibodies, such as those that bind to bacteria. These antibodies bind to a small portion of the invading matter, but this part is common across a broad class of foreign substances. The immune system also includes a suite of antibodies that are specific to a particular antigen. These *specialist* antibodies bind specifically to one antigen and have only random similarities to others.

The experimental environment consists of a given set of fixed length antigens. Forrest *et al.* used separate fitness functions to evolve successfully both specialist and generalist antibodies. They formulated the problem as a matching problem; the match score for an individual, candidate antibody, paired with a particular antigen is simply the number of complementary bit positions in the two individuals. For the generalist problem, the fitness value for an individual was computed by selecting some subset of antigens and calculating the average match score for the individual across the selected set. For the specialist problem, a subset of individuals was selected from the population and one antigen was randomly chosen; the antibody individual with the highest match score against the chosen antigen received that match score as its fitness value.

The work of Forrest *et al.* raises the question of whether

it is possible to evolve simultaneously both generalists and specialists. We answer a more general form of that question. Specifically, the problem addressed here is the following:

Given a hierarchically-related set of antigens each with length  $l$ , can a genetic algorithm evolve a set of antibody individuals some of which match 25% of the positions of all the antigens (generalists), with others matching 50% of the positions of some of the antigens (mid-levels) and all of the positions of one particular antigen (specialists)?

Figure 1 shows the particular antigen set selected for these experiments. The length of the antigen is 64, and there are four distinct antigens. This antigen set, designed based on a Latin square [1], is constructed so that a specialist for one antigen matches none of the other antigens; a mid-level for some set of two antigens matches 50% of those two antigens and none of the other two antigens; and a generalist matches 25% of all the antigens. Thus, this set provides the most separation possible between the antigens. The trees show the three distinct ways that the antigens could be grouped for the mid-level antibodies. As a result, there are a total of 2,944 different antibody individuals that could fill positions in these trees: 2,520 generalists, 420 mid-levels, and 4 specialists.

### 3. Competition and evolving hierarchies

Typical genetic algorithms drive towards a single solution. The critical components of a genetic algorithm include the representation of individuals, the fitness function, the crossover and mutation operators, and the selection mechanism. To design a genetic algorithm for a particular problem, these individual components each have to be addressed. Our representation is a fixed length bit string of length 64, and a population of size 500 is used throughout these experiments. The operators are two-point crossover and single bit mutation, applied at the rate of 0.600 and 0.001 respectively. Selection of individuals for the next generation uses roulette wheel selection, based on their fitness relative to the population mean. The elitist strategy ensures that the best individual is preserved in the population. The genetic algorithm package Genesis is the basis for our implementation [5].

The major difference in this genetic algorithm is in the computation of fitness. The fitness function uses the match score, a bitwise comparison of antigens and antibodies. Each complementary bit position contributes equally to the match score for an antigen-antibody pair. As described earlier, Forrest *et al.* used different fitness functions to evolve generalist and specialist antibodies. The fitness function we use combines these different functions. A major distinguishing feature from the traditional genetic algorithm is

that the fitness of an individual is *dependent* on other individuals in the population. Tournaments are used not for selection but for the evaluation of fitness. Fitness computation proceeds by selecting a set of  $\alpha$  antigens and a set of  $\sigma$  antibodies. Each antibody individual  $s$  in the set is compared to each antigen  $r$  in the antigen set; the score for each individual  $s$  is the average match score across all the antigens. The antibody with the highest resulting score is awarded this score as its fitness value. Multiple sets of antibodies are tested against the same antigen set, constituting a match. Multiple sets of antigens are tested during a generation; each of these is considered a tournament. Figure 2 contains pseudo-code for the fitness computation.

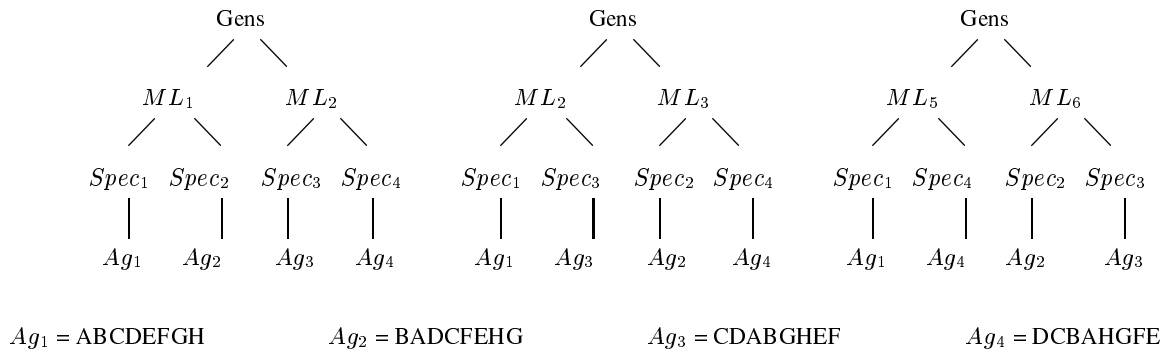
The selection of both antibodies and antigens is without replacement. Therefore, assuming there are no duplicate antigens in the overall set, all antigens in the test set will be distinct. Since there can be multiple representatives of a particular antibody in a population, the set of antibodies may have repeats. Each representative competes for fitness independently, with ties broken randomly.

For these experiments, we vary  $\sigma$  from 1 to 30. Since the antigen suite contains four individuals,  $\alpha$  varies from one to four. We deterministically choose the antigen sets. The number of antigens in the test set,  $\alpha$ , determines  $\delta$ . If  $\alpha = 2$ ,  $\delta = 6$  since there are six distinct subsets containing two antigens. The fitness trials are still random, since the antibody sets are chosen randomly. To increase the probability that every individual gets some chance for fitness,  $\sigma = 1500$  (three times the size of the population). All runs were stopped at 300 generations.

### 4. Results

The experiments discussed in this section were performed multiple times with different random seeds. As the trends remained consistent, we show the results for the first run of each experiment. Table 1 summarizes the results for the experiments; the full tables appear in Section 7. Each entry in the table either indicates that all covering individuals evolved in the experiment, denoted by †, or identifies the source of the failure. For example, looking at the  $\alpha = 2$  line, three different failure codes appear. For  $\sigma = 1$ , there were no specialists in the final population, shown by the failure code of 2. For  $\sigma = 2 - 4$ , the complete suite of specialists did not evolve. For the rest of the values of  $\sigma$ , generalists were missing and there were missing specialists.

We perform two broad classes of experiments. In the first set, we use a fixed (static) value for  $\alpha$ . For the second set, we vary the settings to see if some combination of values might be effective. As there are a huge number of possible combinations, we start with three.



**Figure 1. Hierarchical Trees**

$\mathcal{F}(int \alpha, \sigma, \delta, \gamma)$   
Initialize all fitness values  $f_i$  to 0.  
**for** (i = 1 to  $\delta$ )  
  Choose a set  $G = \{g_i | 1 \leq i \leq \alpha\}$  of  $\alpha$  antigens without replacement  
  **for** (j = 1 to  $\gamma$ )  
    Choose a set  $B$  of  $\sigma$  antibodies without replacement  
    **for** (each  $b \in B$ )  
      
$$M_b = \sum_{k=1}^{\alpha} \sum_{m=1}^l b[m] \neq g[m]$$
      **endfor**  
      Let  $f_{b_q} = f_{b_q} + M_{b_q}$  for  $b_q \in B$  such that  $b_q$  has the highest match score.  
    **endfor**  
  **endfor**  
**endfor**

**Figure 2. Fitness Function Computation.**  $l$  = length of the individuals;  $\alpha$  = number of selected antigens;  $\sigma$  = number of selected antibodies;  $\delta$  = number of tournaments;  $\gamma$  = number of matches per tournament. Ties are broken randomly.

Exp	$\sigma$											
	1	2	3	4	5	6	7	8	9	10	15	30
$\alpha = 1$	3	3	‡	3	‡	‡	‡	‡	‡	‡	1	1
$\alpha = 2$	2	3	3	3	4	4	4	4	4	4	4	4
$\alpha = 3$	2	2	3	‡	‡	‡	‡	‡	‡	‡	1	1
$\alpha = 4$	3	2	2	2	2	2	2	2	2	2	2	2
25%	2	3	3	3	3	‡	3	‡	5	‡	1	‡
2&4	2	3	3	3	3	3	3	4	4	4	4	4
1&3	2	2	2	‡	‡	‡	‡	‡	‡	‡	1	1

**Table 1. Summary of Results:** ‡ = success, failure codes: 1 = no generalists, 2 = no specialists, 3 = insufficient specialists, 4 = both insufficient specialists and no generalists, and 5 = insufficient midlevels.

#### 4.1. Static configuration

We vary the  $\sigma$  value from 1 to 10, and then include 15 and 30. Several general trends are immediately apparent. Generalists appear immediately in the population, increase their position in the population, and then die out rapidly as  $\sigma$  increases. Mid-level antibodies also tend to appear immediately, increase in representation, and then decline to a smaller, but still significant level. An issue with the mid-level antibodies is ensuring that the trees are balanced. Specifically, we declare success only if both sides of all three trees are represented in the mid-level individuals of the population. Specialists sometimes do not emerge until a larger  $\sigma$  value, but they tend to dominate the population as  $\sigma$  climbs. This trend holds except for  $\alpha$  values of two and four.

The complete suite of antibodies never evolves when  $\sigma$  is set to either two or four. In both these cases, the problem is evolving the complete suite of specialists. With  $\alpha = 4$  we have no specialists at all. With  $\alpha = 2$ , only one half of the tree evolves the appropriate specialists.

Hence, the successful runs are those with  $\alpha$  set to either one or three. The middle range of  $\sigma$  values are successful; in this the range the specialists have had time to appear and the generalists have not yet died away.

#### 4.2. Variable configurations

We choose three different combinations for the variable trials to present here. First, we cycle through the  $\alpha$  values evenly, giving 25% of the trials to each of the possible  $\alpha$  values. The second trial focuses only on the two bad choices from the static tests, allocating 50% of the trials to each of the  $\alpha$  values two and four. The third trial allocates the trials equally to the  $\alpha$  values one and three. Other combinations including either two or four have been generally unsuccessful (results not shown).

For the first configuration, we have successful evolution for  $\sigma$  values of 6, 8, 10, and 30. This configuration has the same general trends as the earlier successful trials with a couple of differences. The specialists take longer to get established in the population and are not necessarily stable, resulting in the missing successes of 5 and 7. The distribution of the mid-levels is also sensitive. Finally, while the generalists die off as is the trend from before, there are some generalists at  $\sigma$  of 30, which is unusual.

The second combination demonstrates that the deficiencies of the values two and four are not complementary, since there are no successes in this trial. The essential problem is still the missing specialists.

For the final combination, the pattern holds for  $\alpha$  values of one and three from the static trials. Successful evolution occurs at the middle values of  $\sigma$ , beginning when the specialists arrived and ending when the generalists died out.

### 5. Analysis

To understand these results, it is instructive to discuss the intuition behind this fitness computation. A generalist antibody receives a score of at most 16 when matched against any antigen, while the correct specialist receives a score of 64 when matched against the appropriate antigen. Thus, any competition between a generalist and the correct specialist against one antigen will be won by the specialist. In all but the case of  $\alpha = 4$ , then, the generalist is bound to lose if the correct specialist is present. The probability of the correct specialist being in the match increases as the size of the match ( $\sigma$ ) increases. Therefore, the die-off of the generalists is not unexpected.

Mid-level antibodies are present most of the time, although they are not always well-distributed among the various possible trees. A mid-level antibody can receive a fitness of 32 in a match against either of its two antigens.

Thus, the correct mid-level also defeats a generalist if both are present in a match. Indeed, until the value of  $\alpha$  reaches four, generalists are always at a fitness disadvantage. The domination of the generalists when  $\alpha$  is four makes sense, then, particularly when one considers that there are far more generalists available in the search space than specialists and mid-levels combined. The slow introduction of specialists is also consistent with intuition, since there are far fewer specialists in the search space (4 versus 2,520 generalists).

The most counter-intuitive result is the total failure of the system for  $\alpha = 2$ . While an explanation exists for the failure at  $\alpha = 4$ , it does not appear that the values of  $\alpha = 2$  and  $\alpha = 3$  should differ so radically. Generalists die out far more rapidly and only half of the specialists evolve. The combined experiments showed that the addition of a  $\alpha$  value of two seriously hurt the performance of the genetic algorithm. The instability of the successes in the first variable experiment demonstrates part of this effect.

## 6. Genetic algorithms and hierarchical evolution

The problem described here and that is present in the immune system is a hierarchical learning problem. While specialized concepts exist to describe each object in the object set, there are also sensible and desirable general concepts represented that summarize the characteristics of subsets of the objects. This sample problem provides the opportunity to understand the conditions required to evolve concepts at various levels of generality. We construct an object set that has three levels of concepts, moving from completely specialized to completely general. The results indicate that a genetic algorithm is capable of reliably evolving all levels of this hierarchy in a single run, based only on the degree to which the concept matches some subset of the object set. The genetic algorithm evolves these solutions with no guidance as to either the number of levels or the number of concepts represented by the object set.

In the experiments performed on this object set, the complete set of concepts consistently evolves with competition for fitness between four to six individuals and when the performance in a competition involves matching against either one or three antigens. This moderate value for the size of the competition represents the value for which both generalists and specialists evolve; mid-level generalists exist throughout the range of competition sizes. Specialists are unable to evolve when performance is measured against all four antigens, although specialists perform well when matched against three antigens. For large competitions, generalists dominate when matched against all four antigens. The behavior described scales from a problem size of 32 bits (results not shown).

These experiments both establish the ability of genetic

algorithms to evolve hierarchical concepts and confirm some of our intuitions about the conditions for this evolution to occur. However, some of the results pose intriguing questions. For example, the genetic algorithm evolves only one half of the tree when matched against two antigens ( $\alpha = 2$ ). One possible explanation for this behavior is genetic drift. The previous results of Forrest *et al.* indicate that at larger values of  $\sigma$  the genetic algorithm should be able to maintain multiple peaks of solutions at various levels of fitness. However, the genetic algorithm continues to neglect one side of the tree even at the larger values of  $\sigma$ . Another explanation is that the population size is too small to support the required number of peaks. This explanation is not sufficient, however, since the genetic algorithm can maintain these peaks under other fitness conditions. Additional experiments are required to understand how the peaks are developed and maintained when  $\alpha = 2$  and how this differs from the cases of  $\alpha = 1, 3$ .

We will attempt to answer these questions through further experiments and further analysis of the underlying mechanics of this form of genetic algorithm. Obvious additional experiments include larger populations sizes and related antigen trees to study the effect of hamming distance amongst the concepts on the ability to evolve the concepts. Our initial trials with alternative antigen sets have been encouraging. We also intend to introduce different mixes of antigen presentations and to vary the method of presenting antigens for competitions to better characterize the dynamics of the system.

The genetic algorithm presents a different approach to hierarchical evolution. The genetic algorithm evolves, based only on the degree to which a concept corresponds to the object, a hierarchy of descriptions ranging from a general characterization of all objects to specific concepts that uniquely identify an object. The mid-levels partition the set of objects based on differing characteristics. The genetic algorithm finds the different consistent partitions of the object set over a fairly wide range of parameter settings. These preliminary results indicate that the genetic algorithm can be used in unsupervised settings when experimental clusterings of objects are desired. The ability of the genetic algorithm to evolve, simultaneously, different consistent clusterings is a useful addition to the suite of conceptual clustering techniques.

## 7. Appendix: tables of results

This section documents the complete results for the different experiments. All differences between the runs are noted in the caption for the separate tables. Entries in each table consist of the number of individuals (Indiv), the number of different types (Type), and the standard deviation (Stdev) of the number of individuals per type. The match

**Table 2. Static  $\alpha$ ,  $\alpha = 1$**

		$\sigma$											
		1	2	three	4	5	6	7	8	9	10	15	30
Gen	Indiv	80	110	149‡	90‡	54‡	52‡	14‡	12‡	6‡	3‡	0	0
	Type	76	90	140	87	52	51	14	12	6	3	0	0
	Stdev	0.22	0.58	0.25	0.18	0.19	0.14	0	0	0	0	0	0
ML	Indiv	19	121	143	320	391	325	296	207	236	211	282	194
	Type	13	66	103	112	164	174	163	129	137	105	132	108
	Stdev	0.88	1.26	0.73	2.35	2.03	1.20	1.08	0.91	1.10	1.30	1.43	0.97
Spec	Indiv	1	4	4	6	26	62	139	210	235	254	318	339
	Type	1	1	4	3	4	4	4	4	4	4	4	4
	Stdev	0	0	0	1.73	5	6.45	2.99	6.66	4.5	7.72	4.43	4.11

threshold is 63 bits. ‡ represents hierarchically evolved antibodies. For the variable results, each experiment used  $\alpha$  the indicated percentage of time.

## References

- [1] G. E. Box, W. G. Hunter, and J. S. Hunter. *Statistics for Experimenters, An Introduction to Design, Data Analysis and Model Building*. John Wiley and Sons, 1978.
- [2] S. Forrest. Genetic algorithms: Principles of natural selection applied to computation. *Science*, 261:872–878, 1993.
- [3] S. Forrest, B. Javornik, R. Smith, and A. Perelson. Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1, 1994.
- [4] D. Goldberg. *Genetic Algorithms in Search Optimization, and Machine Learning*. Addison Wesley, 1989.
- [5] J. J. Grefenstette. Genesis: A system for using genetic search procedures. pages 161–166, 1984.
- [6] J. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan, USA, 1975.



**Table 6. Variable  $\alpha$ , Even 25% Mix**

		$\sigma$											
		1	2	3	4	5	6	7	8	9	10	15	30
Gen	Indiv	93	132	224	134	92	47‡	55	15‡	18	13‡	0	6‡
	Type	89	116	209	128	91	47	54	15	18	13	0	6
	Stdev	0.21	0.44	0.28	0.21	0.11	0	0.14	0	0	0	0	0
ML	Indiv	17	83	88	182	280	290	220	277	308	301	291	260
	Type	15	43	75	113	126	150	136	156	175	162	148	116
	Stdev	0.52	1.33	0.45	1.22	1.73	1.28	1.09	1.17	1.24	1.24	1.49	1.69
Spec	Indiv	0	3	3	10	5	45	74	125	141	161	230	267
	Type	0	1	1	3	3	4	3	4	4	4	4	4
	Stdev	0	0	0	2.08	0.58	8.06	19.66	20.10	37.35	14.01	7.85	2.87

**Table 7. Variable  $\alpha$ , Even 50% Mix with 2 and 4**

		$\sigma$											
		1	2	3	4	5	6	7	8	9	10	15	30
Gen	Indiv	127	200	241	186	52	36	15	3	0	6	0	0
	Type	113	168	215	180	47	36	15	3	0	6	0	0
	Stdev	0.38	0.50	0.35	0.18	0.31	0	0	0	0	0	0	0
ML	Indiv	18	104	86	188	95	84	73	55	47	52	51	37
	Type	16	59	70	96	63	55	34	22	28	20	23	19
	Stdev	0.34	1.38	0.49	1.20	1.06	1.20	2.15	1.90	1.25	1.57	1.98	1.18
Spec	Indiv	0	3	2	12	51	95	136	150	153	160	196	229
	Type	0	1	2	2	2	2	2	2	2	2	2	2
	Stdev	0	0	0	4.24	13.44	31.82	2.83	18.38	0.71	1.41	7.07	6.36

**Table 8. Variable  $\alpha$ , Even 50% Mix with 1 and 3**

		$\sigma$											
		1	2	3	4	5	6	7	8	9	10	15	30
Gen	Indiv	96	192	192	75‡	35‡	33‡	24‡	3‡	1‡	9‡	0	0
	Type	83	180	180	70	32	33	24	3	1	9	0	0
	Stdev	0.40	0.77	0.29	0.26	0.30	0	0	0	0	0	0	0
ML	Indiv	16	123	123	275	402	296	237	286	241	257	248	224
	Type	13	92	92	137	147	152	142	156	140	137	126	113
	Stdev	0.44	2.14	0.67	1.33	2.29	1.29	1.04	1.24	1.09	1.27	1.35	1.30
Spec	Indiv	0	0	0	7	23	64	162	201	227	258	311	351
	Type	0	0	0	4	4	4	4	4	4	4	4	4
	Stdev	0	0	0	0.50	3.77	2	3.79	3.10	3.10	1.91	3.77	10.50