ALPS: The Age-Layered Population Structure for Reducing the Problem of Premature Convergence

[Genetic Programming Track]

Gregory S. Hornby
University Affiliated Research Center, UC Santa Cruz
NASA Ames Research Center, Mailstop 269-3
Moffett Field, CA
hornby@email.arc.nasa.gov

ABSTRACT

To reduce the problem of premature convergence we define a new attribute of an individual, its *age*, and propose the Age-Layered Population Structure (ALPS), in which age is used to restrict competition and breeding between members of the population. ALPS differs from a typical EA by segregating individuals into different age-layers by their "age" – a measure of how long the genetic material has been in the population – and by regularly replacing all individuals in the bottom layer with randomly generated ones. The introduction of new, randomly generated individuals at regular intervals results in an EA that is never completely converged and is always looking at new parts of the fitness landscape. By using age to restrict competition and breeding search is able to develop promising young individuals without them being dominated by older ones. We demonstrate the effectiveness of the ALPS algorithm on an antenna design problem in which evolution with ALPS produces antennas more than twice as good as does evolution with two other types of EAs. Further analysis shows that the ALPS model does allow the offspring of newly generated individuals to move the population out of mediocre local-optima to better parts of the fitness landscape.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: General

General Terms

Algorithms, Design

Keywords

Computer-Automated Design, Design, Evolutionary Algorithms, Evolutionary Design, Open-ended design

1. INTRODUCTION

A common problem experienced by practitioners of evolutionary computation (EC) is that they run their evolutionary algorithm (EA) and after a while the fitness of the best solution levels off at some mediocre value and no further improvements are made. What has happened is that the existing genetic material in the population has converged such that with the current individuals in the population the variation operators cannot produce new individuals that will move the population into a better basin of attraction. This has been called the *premature convergence* problem [3, 6, 13, 8].

Perhaps the easiest ways of slowing convergence are to increase the mutation size/rate or the population size. Increasing the mutation rate will keep diversity high and keep the population from converging quickly but it is just as likely to replace good alleles and building blocks as bad ones. Also, if the mutation size is too large then the mutation operator will not create offspring near its parent and be unable to explore narrow fitness peaks. Using a larger population increases the number of generations before it converges but the problem then becomes picking a population size. Too large a population on a simple problem results in search taking much longer than necessary, and on difficult problems the necessary size of the population may not be feasible.

More involved approaches of maintaining the genotypic diversity of the population consist of modifying other parts of the evolutionary algorithm and, over the years, different such methods have been tried. Diversity of the population can be maintained by modifying the replacement strategy, such as with *preselection* [5], *crowding* [6], and *deterministic crowding* [14]. Another approach is the use of *sharing functions*, which modify the fitness of individuals based on their genotypic similarity [7]. Or the population structure can be modified, such as with spatially structured populations in which individuals have a location and are restricted to interacting with their neighbors [17]. While these different methods can work to different degrees, drawbacks are that methods which work through genotypic comparisons are better suited to bit strings then genetic programs and, ultimately, all such methods are limited to discovering solutions that are within the basin(s) of attraction of the initial population.

Breaking out of the basin of attraction of the initial starting population can only be achieved by introducing new, randomly generated individuals into the population. Implicitly, this method of getting around the premature convergence problem is a fairly common practice that is done by running the EA multiple times with different random number seeds. While restarting the EA increases the chances of eventually finding the global optima – at the very least, if the random individual generator can produce every point in the fitness landscape then eventually the global optima will be found in one of the initial populations – deciding how long to run the EA before restarting becomes a challenge. If few generations are used, then the population may not have enough time to climb the fitness peak of the global optima, if a large number of generations are used then much time will be wasted while the population has converged on top of a mediocre fitness peak before the next run is started.

Rather than restarting the EA from scratch with an entirely new population, the alternative is to continuously introduce into the population new, randomly generated individuals. Hu and Goodman proposed such an algorithm and called it the Hierarchical Fair Competition (HFC) model [10]. To protect new individuals and their offspring from competition with the high-fitness, pre-existing individuals, HFC separates the population into different fitness-layers with selection and replacement occurring only within a fitness-layer, similar to the Fitness Uniform Selection (FUSS) EA [12]. The idea behind this is to allow individuals, and their offspring, to develop much like how sports players are brought up through various school and minor league systems before playing in the top tier. Yet HFC has the problem that individuals that have converged to a local optima near the top of a fitness layer prevent newer individuals in different basins of attraction from climbing through that fitness-layer. Variations on the HFC paradigm – Adaptive Hierarchical Fair Competition (AHFC) [11] and Continuous Hierarchical Fair Competition (CHFC) [1] – have been made but these have not been shown to be significantly better than regular HFC.

To better integrate new, randomly generated individuals we define a new attribute of an individual, its *age*, and propose the Age-Layered Population Structure (ALPS), in which age is used as attribute to restrict competition and breeding between members of the population. An individual's age is a measure how long its genetic material has been evolving. Randomly created individuals starting with an age of 0, with its is age increased by one for each generation in which it is used to produce an offspring, and individuals created through mutation or recombination start with an age of 1 plus the age of its oldest parent(s). The population is then separated into multiple layers, with each layer having a maximum allowable age for individuals to be in it, and selection, breeding, and replacement is restricted to adjacent layers. By structuring the population so that individuals only compete against other individuals of similar ages, individuals will only cluster about a local optima as long as it has the best fitness for similarly aged individuals thereby allowing other newly-discovered basins of attraction to be explored.

To demonstrate the effectiveness of ALPS it is compared against HFC and a standard EA on an antenna design problem. The results show that ALPS significantly outperforms both a standard EA and HFC by a large margin and also that by using age to restrict competition and breeding the genetic material of newly generated random individuals is able to evolve up new fitness peaks and move the population to new and better parts of the fitness landscape.

The rest of this paper is organized as follows. First, a definition of age and the ALPS paradigm is described in detail. The next section is a description of the experimental setup for comparing ALPS against a canonical EA and against a variant of HFC. This is followed by the results of the experiments and then a discussion analyzing the behavior of the ALPS algorithm in comparison with the other two. Finally, we close with a short section on combining ALPS with other diversity maintenance techniques and then summarize the conclusions of this work.

2. THE ALPS PARADIGM

As with HFC, the Age-Layered Population Structure differs from traditional evolutionary algorithms by regularly introducing new, randomly generated individuals in the population and by segregating individuals in the population and restricting which other individuals they compete and mate with. The difference between ALPS and HFC is that ALPS uses *age* as the attribute on which the competition and mating restrictions are based whereas HFC uses *fitness*. We define an individual's age as a count of how long its genetic material has been evolving inside the population, as measured by the number of generations in which it has been used as a parent. Thus with ALPS, the population consists of a sequence of layers, with an increasing upper-limit on the maximum age of individuals which that layer can contain. Evolution of individuals proceeds much like a typical EA, except two exceptions. First, individuals are restricted to only breeding with individuals in their own layer or from the layer immediately before them. Second, the bottom layer is replaced with randomly generated individuals at regular intervals. We now describe this algorithm in more detail.

The age-measure that we define for the ALPS-EA is a count of how many generations in which the individual's genotypic material has been evolving inside the population. New individuals, that are randomly generated, start with an initial age of 0 since their genetic material has just been introduced into the population. Individuals that are created through variation, such as by mutation or recombination, take the age value of their oldest parent plus 1 since their genetic material comes from their parents and has now been in the population for one more generation than their parents. Each generation in which an individual is used as a parent to create an offspring its age is increases by 1 since its genetic material has been used in evolution in another generation. Even if an individual is selected to reproduce multiple times in one generation its age is still only increased by 1 so that good individuals that reproduce a lot are not penalized for being more fit than similarly aged individuals.

To restrict competition and breeding among individuals the population keeps individuals in a number of age-layers, somewhat similar to the island-model EA [17]. Each age-layer in the population has a maximum age limit for individuals in it, except for the last layer which can have individuals of any age. Different systems can be used for setting these values, such as by using linearly, polynomialy or exponentially increasing limits (see table 1). To keep the size of the population and number of layers manageable, and since there is generally little need to segregate individuals which are within a few "generations" of each other, these values are then multiplied by an age-gap parameter. Thus with a polynomial aging scheme and an age-gap of 20 the maximum

Table 1: Different systems for setting the age-limits for each age-layer.

	Max age in layer						
Aging-scheme	0	1	2	3	4	5	6
Linear	1	2	3	4	5	6	7
Fibonacci	1	2	3	5	8	13	21
Polynomial (n^2)	1	2	4	9	16	25	49
Exponential (2^n)	1	2	4	8	16	32	64

.

ages for the layers are: 20, 40, 80, 180, 320, ... This allows some generations of evolutionary search to occur in the first layer with the randomly generated individuals, allowing the population can find and move into basins of attraction, before being pushed into the next layer and results in a good separation in age-limits between subsequent layers. Alternatively, an ALPS-EA can be run with an infinite number of age-layers, with new layers created as needed, with the expectation that the user will halt evolution before memory is exhausted.

With these age-layers, evolution with the ALPS model is similar to a canonical EA with a few exceptions. First, individuals can breed only with individuals from their own layer or from the previous one. Thus for layer 0, parents are selected from individuals only in layer 0; for layer 1 parents are selected from individuals in layers 0 and 1; for layer 2 parents are selected from individuals in layers 1 and 2; and for layer n, parents are selected from layers $n-1$ and n. By selecting parents individuals from both the current layer and the previous layer, offspring of individuals are able to pass from one layer to the next in a smooth way. Second, at regular intervals all individuals in the first layer, layer 0, are replaced with randomly generated individuals. This happens at every `AGE-GAP` generations. Thus with an age-gap value of 7, new individuals are created in the first layer at generations 0, 7, 14, 21, ... Finally, individuals are only present in a layer when evolution has proceeded for as many generations as the age limit of the previous layer. For example, with an age-gap value of 10 and an exponential aging system, only layer 0 is active for the first 10 generations at which point layer 1 is then used, layer 2 is used starting at generation 20, layer 3 is used starting at generation 40, layer 4 is used starting at generation 90, and so on. When a generation becomes open for use, its individuals are created through offspring from parents selected from the previous layer.

By increasing an individual's age each generation in which it is used as a parent, it will eventually age its way up the layers and, likely, out of the population so that, unlike with HFC, an individual will not stay in a layer forever. Except for the last layer in which case an individual is only guaranteed to stay forever if it is the global optimum, otherwise it will eventually be replaced as better individuals are evolved.

3. EXPERIMENTAL SETUP

To determine the effectiveness of the ALPS paradigm, experiments are performed using it as well as a canonical EA and HFC. A canonical EA is used so that a performance comparison can be made between it and the ALPS-EA to show how well ALPS performs against the style of EA used by the majority of practitioners in the field of evolutionary computation. Runs with HFC provide a useful comparison because it also regularly generates new, randomly generated individuals in its first layer and by comparing ALPS against HFC the relative advantage of using age, versus fitness, to segregate and manage individuals in different layers can be inferred. The details of the three different EAs will now be described.

3.1 Configuration of the Different EAs

The particular ALPS implementation that is used for these experiments is as follows. The population is made up of 10 age-layers with each layer having 100 individuals, resulting in a total population size of 1000 individuals. A polynomial aging scheme is used with an `age-gap` parameter of 20 generations. Tournament selection is used with a tournament size of 7 and an elitism of 3 in each layer (and an overall elitism of 30). New individuals are created with an equal probability of mutation or recombination.

The canonical EA is a generational EA with a population size of 1000 individuals and uses tournament selection with a tournament size of 7 and elitism of 30. New individuals are created with an equal probability of mutation or recombination.

The HFC model that is used is adaptive HFC (AHFC), in which the fitness thresholds for each layer are adjusted every fixed number of generations. Our implementation uses 10 fitness-layers, with 100 individuals in each layer for a total population size of 1000 individuals. Parents are selected using tournament selection, with a tournament size of 7. In addition, an elitism of 3 is used to copy the best individuals of each layer from one generation to the next. Fitness thresholds are adjusted every 20 generation.

3.2 Representation

As will be described in the following section, the problem domain in which the experiments will be performed is that of antenna design. Antennas are encoded with an open-ended representation with which the nodes of the genotype are antenna-construction operators that specify how to construct the antenna. Constructing an antenna begins with a feedwire of length 1mm coming up out of the ground-plane and operators are executed starting with the root node down to the leaf node. In constructing an antenna the current state (location and orientation) is maintained and operators add wires or change the current state. The operators are as follows: `forward(length)`, add a wire with the given length and extending from the current location and then change the current state location to the end of the new wire; `rotate-x(angle)`, change the orientation by rotating it by the specified amount (in radians) about the x-axis; `rotate-y(angle)`, change the orientation by rotating it by the specified amount (in radians) about the y-axis; and `rotate-z(angle)`, change the orientation by rotating it by the specified amount (in radians) about the z-axis. Since we constrained antennas to a single, bent wire with no branching each node in the genotype has at most one child.

For example, in executing the program `rotate-z(0.5236) forward(1.0)`, the `rotate-z()` operator causes the the current orientation to rotate 0.5236 radians (30°) about the Z axis. The `forward()` operator adds a wire of length 1.0 cm in the current forward direction.

4. EXPERIMENTS

To test the hypothesis that the ALPS method of managing individuals in the population is better at producing high quality solutions we compare it to HFC and a regular EA on an antenna design optimization problem. Researchers have been investigating evolutionary antenna design and optimization since the early 1990s [15, 9, 2, 16], and the field has grown in recent years as computer speed has increased and electromagnetics simulators have improved. The goal of this antenna problem is to produce an omni-directional monopole antenna operating at 50Ω with a gain pattern of ≥0 dBic from 0° - 80° from zenith for both transmit (2288 MHz) and receive frequencies (2106 MHz), a voltage standing wave ratio (VSWR) of under 1.5 at both frequencies, and fit inside a cylinder of height 6cm and radius of 5cm.[1]

The configuration for all three EAs uses the same total population size, the same tournament size, the same overall elitism size, and all three use the same representation with the same variation operators. The only differences between the canonical EA and the other two is that both ALPS and AHFC regularly introduce new, randomly generated individuals the population and both manage their populations differently from the canonical EA. Thus performance and behavioral differences between the canonical EA and both ALPS and AHFC must be a consequence of whether or not the algorithm introduces new individuals into the population and/or the differences in how they manage the population. As for ALPS and AHFC, both replace the initial layer with a new group of randomly generated individuals at the same rate – every 20 generations – but they differ in using *age* versus *fitness* to segregate individuals and restrict mating and competition, thus this will be cause of performance differences between these two algorithms. In this way the experiments allow us to determine which algorithmic features are important and advantageous.

4.1 Antenna Optimization Problem

The fitness function used to evaluate antennas is a function of the VSWR and gain values on the transmit and receive frequencies. The VSWR component of the fitness function is constructed to put strong pressure toward evolving antennas with receive and transmit VSWR values below the required amounts of 1.2 and 1.5, reduced pressure at a value below these requirements (1.15 and 1.25) and then no pressure to go below 1.1:

$$\begin{aligned}
v_r &= \text{VSWR at receive frequency} \\
v_r' &= \begin{cases} v_r + 2.0(v_r - 1.25) & \text{if } v_r > 1.25 \\ v_r & \text{if } 1.25 > v_r > 1.1 \\ 1.1 & \text{if } v_r < 1.1 \end{cases} \\
v_t &= \text{VSWR at transmit frequency} \\
v_t' &= \begin{cases} v_t + 2.0(v_t - 1.15) & \text{if } v_t > 1.15 \\ v_t & \text{if } 1.15 > v_t > 1.1 \\ 1.1 & \text{if } v_t < 1.1 \end{cases} \\
vswr &= v_r' v_t'
\end{aligned}$$

The gain-penalty component of the fitness function uses the gain (in dBic) in 5° increments about the angles of interest: from $0° \leq \theta \leq 90°$ and $0° \leq \phi \leq 360°$. For each angle, the calculated gain score from simulation is compared against the target gain for that elevation and the outlier gain, which is the minimum gain value beyond which lower gain values receive a greater penalty. Gain penalty values are further adjusted based on the importance of the elevation:

```
gain_penalty (i, j):
    gain = calculated gain at  $\theta = 5°i$ , $\phi = 5°j$;
    if (gain ≥ target[i]) {
        penalty := 0.0;
    } else if ((target[i] > gain) and (gain ≥ outlier[i])) {
        penalty := (target[i] - gain);
    } else { /* outlier[i] > gain */
        penalty := (target[i]-outlier[i]) +
                        3.0 * (outlier[i] - gain));
    }
    return penalty * weight[i];
```

Target gain values at a given elevation are stored in the array `target[]` and are 2.0 dBic for i equal from 0 to 16 and are -3.0 dBic for i equal to 17 and 18. Outlier gain values for each elevation are stored in the array `outlier[]` and are 0.0 dBic for i equal from 0 to 16 and are -5.0 dBic for i equal to 17 and 18. Each gain penalty is scaled by values scored in the array `weight[]`. For the low band the values of `weight[]` are 0.1 for i equal to 0 through 7; values 1.0 for i equal to 8 through 16; and 0.05 for i equal to 17 and 18. For the high band the values of `weight[]` are 0.4 for i equal to 0 through 7; values 3.0 for i equal to 8 through 12; 3.5 for i equal to 13; 4.0 for i equal to 14; 3.5 for i equal to 15; 3.0 for i equal to 16; and 0.2 for i equal to 17 and 18. The final gain component of the fitness score of an antenna is the sum of gain penalties for all angles.

To put evolutionary pressure on producing antennas with smooth gain patterns around each elevation, the third component in scoring an antenna is based on the standard deviation of gain values. This score is a weighted sum of the standard deviation of the gain values for each elevation θ. The weight value used for a given elevation is the same as is used in calculating the gain penalty.

These three components are multiplied together to produce the overall fitness score of an antenna design:

$$F = vswr \times gain \times standard\ deviation$$

Since the objective of the EA is to produce antenna designs that minimize F, this function is actually a *cost* function rather than a *fitness* function.

The Numerical Electromagnetics Code, Version 4 (NEC4) [4] was used to evaluate all antenna designs. Antenna designs were analyzed on top of a 4" ground-plane that was approximated with a wire-mesh, for which each antenna simulation took a several seconds of wall-clock time to run

4.2 Results

A total of 15 trials were performed with each of the three EAs described in the previous section, with each trial run for 2 million evaluations. The averaged results of these trials are (mean±s.e.): standard EA, 150.3±20.3; AHFC, 152.8± 30.6; and ALPS, 61.1±39.4. Using a two-tailed Mann-Whitney test the difference between both ALPS and the standard EA as well as ALPS and AHFC is highly significant, with P < 0.001. In contrast, using a two-tailed

[1]VSWR is a way to quantify reflected-wave interference, and thus the amount of impedance mismatch at the junction.

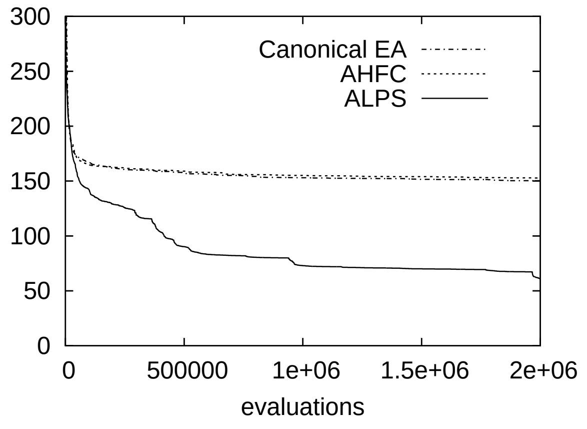


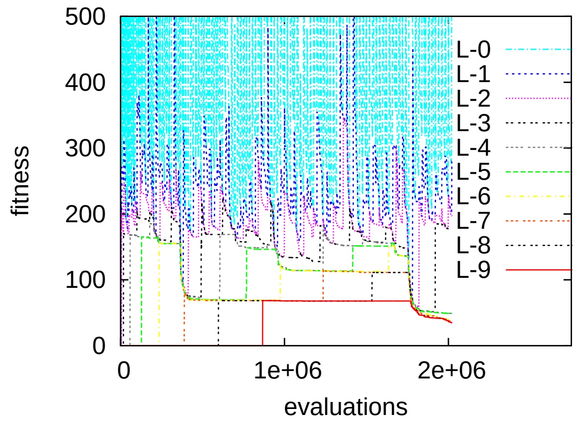
Figure 1: A plot of the best individual in the population for the three different EAs averaged over 15 trials each.

Mann-Whitney test, the performance difference between the standard EA and AHFC is not significantly different with P ≥ 0.05.

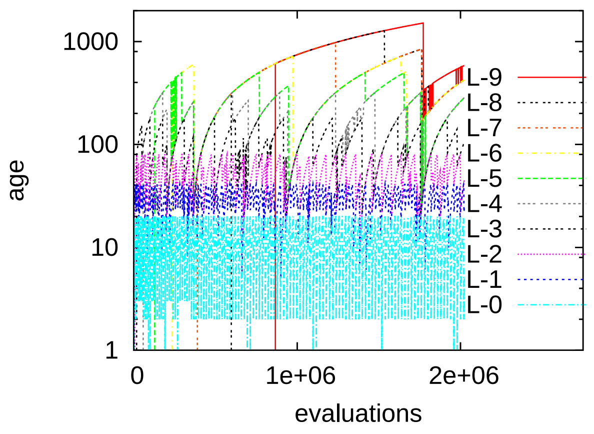
That the ALPS model significantly outperformed the canonical EA, and by a large margin, tells us that continuously introducing new, randomly-generated individuals into the population and restricting breeding and competition by an individual's age can be a better evolutionary algorithm. Since the performance difference between AHFC and a canonical EA is neither large nor statistically significant, these results also show that just introducing new individuals into the population is not sufficient to produce a better EA and that age is a better attribute to control individuals than is fitness.

5. DISCUSSION

An intuitive understanding of why evolution with the ALPS paradigm works so well can be gained by examining the fitness and age values of the best individual in each layer over the course of evolution. The two graphs in figure 2 show one of the trials with the ALPS paradigm, with the graph in figure 2.a consisting of a plot of the fitness of the best individual in each layer of the population and the graph in figure 2.b consisting of a plot the age of these individuals. Since the bottom layer, L-0, of the population is replaced by a new group of randomly generated individuals every 20 generations (which is 20000 evaluations once all layers are populated) it can be seen to oscillate rapidly in both graphs: in figure 2.a it starts with a very high fitness, descends to a fitness value centered around 250 over 20 generations and then starts afresh at something higher then 500; and in figure 2.b the plot of the age of the best individual in layer L-0 shows the age of this individual starts at 0 and then increases to 20 over the course of 20 twenty generations, hand-in-hand with improvements in fitness, and then is reset to 0 as this layer is replaced with new, randomly generated individuals. Similarly, the other layers in these two graphs can be seen to oscillate with a polynomially increasing gap between "resets". Since the best individual in a layer generally takes over a layer with its offspring, all individuals in a layer tend to have a similar age-level. Consequently, when the best individual in a layer is aged out of a layer the rest of the individuals in the layer that are genotypically similar are



(a)



(b)

Figure 2: A plot of the fitnesses of the best individuals in layers 0, 1, 2 and 9 of an EA run using ALPS.

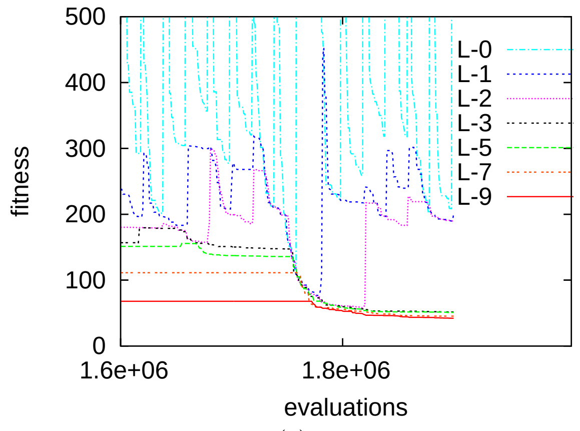
also in the process of being aged-out of the layer. At this point the layer is reset with new genetic material from the best individuals of the previous layer, hence the connecting arcs showing a continuous development of fitness and age of a genetic line progressing through multiple layers.

To help understand why evolution with the ALPS paradigm is better able to continue finding new and better solutions, even when run for a very large number of generations, we examine a close-up of the evolutionary run shown in figure 2. The plot of the best fitness of each layer shown in the graph of figure 3.a starts after 1.6 million evaluations have taken place, and just below it in the graph in figure 3.b is a plot of the age of these individuals. These two graphs show the conjoined oscillation in fitness and age of the best individual in the bottom layer, L-0, every 20 generations. These two graphs also show, by the smooth connection between the lines from adjacent layers, the transfer of genotypic material from one layer to the next. It can be seen that the offspring of individual that was randomly generated sometime around 1.7 evaluations quickly progress up the layers from L-0, to L-1, all the way to L-5, after several thousand evaluations (which is several generations since there are 1000 individuals in the populations). About when this genetic material reaches layer L-5, it is aged out of layer L-0, which then starts evolving a new bunch of randomly generated individuals. Several generations later it also ages out of L-1, and this layer then starts evolving individuals that have moved up to it from L-0. Eventually the genetic material reaches the top layer, L-9, and a new global optima is found. These two graphs show that by using age to restrict competition and breeding between individuals, the ALPS paradigm is able to regularly generate new individuals whose offspring are able to evolve and move the population out of a mediocre local-optima.

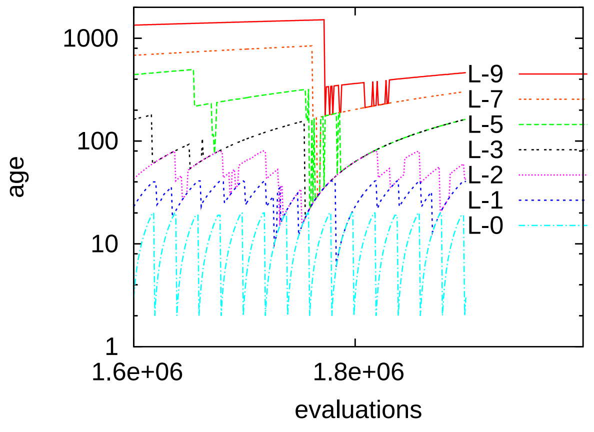
Interestingly, from looking at the plot of ages in figure 3, it shows that the ages of the best individuals from layers L-5, L-7 and L-9 (ages for layers L-4, L-6, and L-8 are not plotted) does not drop down to that from L-2 which suggests that the genetic material that evolved down from L-0 recombined with an older individual from an intermediate layer and the resulting offspring used the age of the older parent. That the new best individual contains some genotypic material acquired through recombination with descendants from the individuals recently created can be shown by plotting the ages of individuals in each generation with a second age-measure. Instead of assigning the age of the oldest parent to offspring created through recombination, with this second age measure individuals are given the age of their youngest parent. The graph in figure 3.c is a plot of the age of the best individual in each layer of the same run as the graphs in figure 3.a and b. This graph shows that at the time the fitness of the best individual in the population starts dropping to a new local-optima the age of the best individual in the top age-layer has an age of 31, using this second age measure. This means that new best individual is a descendant of an the individual that randomly generated some 31 generations previous. In fact, using this second age-measure, the maximum age of the best individuals in any of the age-layers seldom goes higher than 100 which means that the genetic material of th new, randomly generated individuals is being transferred to the rest of the population in the other age-layers.

The behavioral difference between ALPS and HFC can be seen from examining the same fitness and age graphs from a run with the HFC algorithm, figure 4, and comparing them with similar graphs for ALPS. The first graph, figure 4.a, shows the sequential layering of fitness values for each fitness-layer in the population, just as would be expected with the HFC model. Also, since Adaptive HFC (AHFC) was the variant used for these experiments a regular resetting of fitness values can be seen. Both the sequential layering of fitness and the resetting of fitness values are similar to behaviors observed with ALPS. The second graph, figure 4.b, plots the ages of the best individual in each fitness layer. Similar to ALPS, the age of individuals in the bottom layer oscillates from 0, although with AHFC it climbs much higher since there are no age-limits with this system. Unlike ALPS, the age of the best individual in every other layer is the same and comes from the initial population. This second graph shows that AHFC does not enable the movement of the new, randomly generated individuals up the layers.

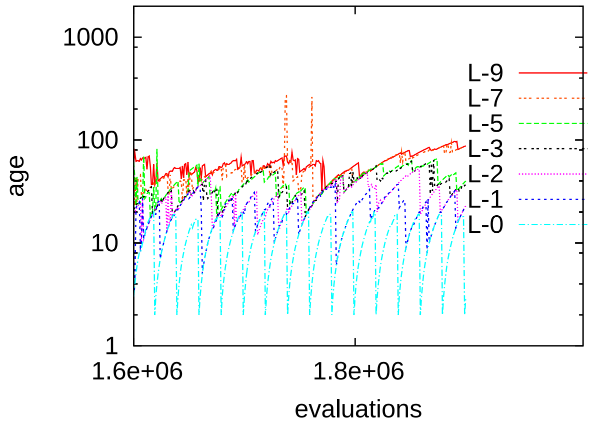
For completeness we include plots of an example run with the canonical EA in figure 5. The graph in figure 5.a plots the fitness of the best individual in the population, and it shows that the fitness of the best individual in population rapidly increases for several thousand evaluations and



(a)



(b)



(c)

Figure 3: Graphs of part of the ALPS run of figure 2 with the lines of some age-layers left out to improve clarity: (a) a plot of the fitness of the best individual in selected layers; (b) a plot of the age of the best individual in selected layers; and (c) a plot of age of the best individual in selected layers using a different measure of age.

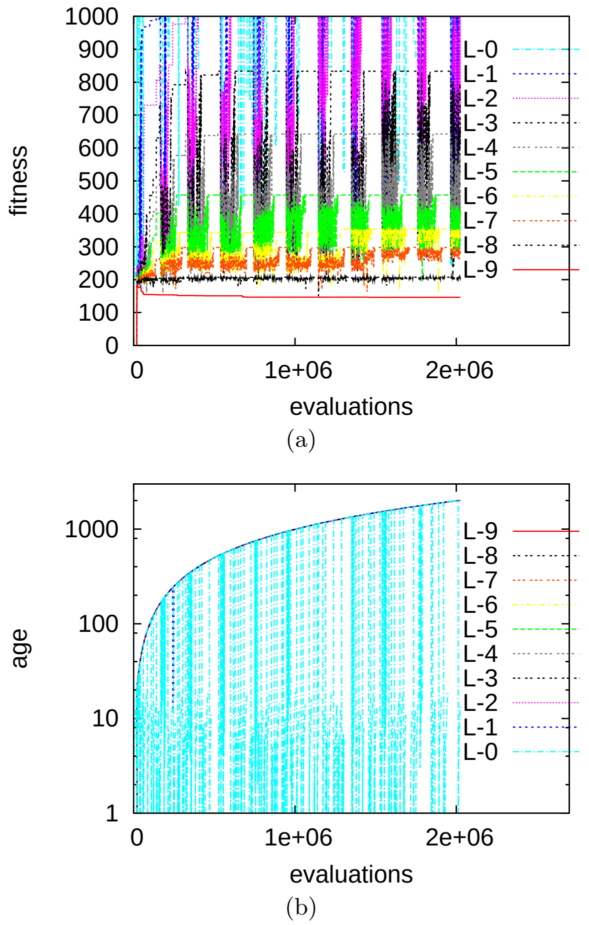


Figure 4: A plot of the fitnesses of the best individuals in layers 0, 1, 2 and 9 of an EA run using ALPS.

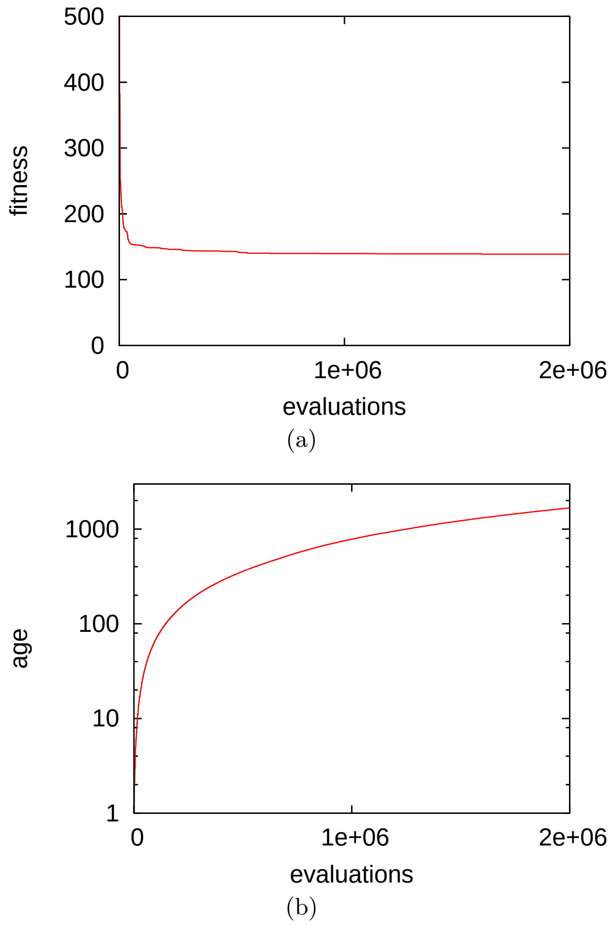


Figure 5: A plot of the fitnesses of the best individuals in layers 0, 1, 2 and 9 of an EA run using ALPS.

then very quickly levels off with only minor improvements made over the remaining 1.95 million evaluations. The corresponding ages of these individuals is plotted in the graph in figure 5.b, and it shows that the best individual in population increases in age by roughly 1 for every generation, as expected for a traditional, generational EA. Interestingly, the plots in these two graphs virtually match the corresponding plots of AHFC – except for the plot of the bottom layer, L-0, of AHFC which is replaced with new, randomly generated individuals every 20 generations. Again, this strongly suggests that AHFC is not successful in developing individuals that are generated at random.

To summarize, the graphs presented in this section show that with the ALPS paradigm new genetic material is continually introduced into the population at regular intervals and is segregated from older individuals by *age*, thereby allowing it time to evolve to its potential. This segregation and development of individuals up the age-layers allows evolution to continue finding new and better solutions even when it is run for a very large number of generations. That this is not happening the AHFC demonstrates that age is the better attribute to restrict competition and breeding than is fitness.

6. COMBINING ALPS WITH OTHER TECHNIQUES

While ALPS is good at using age to shepherd the development of new, randomly-generated individuals, it may be that some problems better control of the individuals inside each population. To address cases in which neither ALPS nor a diversity maintenance scheme are sufficient these two techniques can be combined to achieve better results.

From past work on evolving antennas for other problems an EA using deterministic crowding (DC) had been found to work best [reference omitted]. While evolution using ALPS as described in section 2 resulted in antenna designs that were better then had been produced previously, the resulting designs were still inadequate. Combining ALPS with DC resulted in a system that, based on a few initial results, produces far superior results than evolution with either system alone. The graphs in figure 6 show the performance of one run of an ALPS-DC EA for 7.5 million evaluations. With this system the best antenna that was evolved has a fitness of 12.4, which is better than that of any antenna evolved with the EAs described in section 3.

7. CONCLUSION

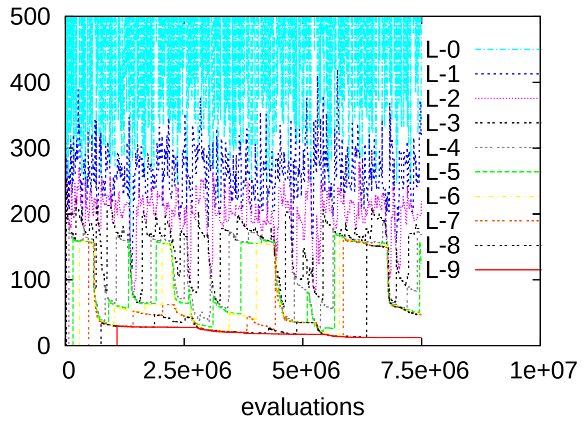


Figure 6: An example of a run of ALPS using deterministic crowding inside each layer.

In this paper we have defined a measure of age of individuals and with this measure have proposed the Age-Layered Population Structure (ALPS) as a system to reduce the problem of premature convergence. Unlike canonical EAs, ALPS continues to explore new parts of the fitness landscape by continuously creating a new sub-population of randomly generated individuals in its bottom layer. By segregating individuals into different layers by their age, and using this to restrict competition and breeding, promising new individuals are able to develop without being dominated by older ones.

To determine the effectiveness of ALPS, it was compared against a canonical EA and the Hierarchical Fair Competition (HFC) model, an EA which also continuously introduces random individuals into the population but uses fitness as the attribute to restrict competition and breeding. ALPS significantly outperformed the other two EAs by a large margin, thereby demonstrating the advantages of using age to restrict breeding and competition. Further improvements in performance where then demonstrated by combining ALPS with deterministic crowding. It is hoped that using age as a means to control the population will provide even better ways to prevent premature convergence in future variants of the ALPS algorithm.

8. REFERENCES

[1] Continuous hierarchical fair competition model for sustainable innovation in genetic programming. In R. L. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practice*, pages 81–98, Ann Arbor, 2003. Kluwer.

[2] E. E. Altshuler and D. S. Linden. Design of a loaded monopole having hemispherical coverage using a genetic algorithm. *IEEE Trans. Antennas & Propagation*, 45(1):1–4, January 1997.

[3] J. E. Baker. Adaptive selection methods for genetic algorithms. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 101–111, 1985.

[4] G. J. Burke and A. J. Poggio. Numerical electromagnetics code (nec)-method of moments. Technical Report UCID18834, Lawrence Livermore Lab, Jan 1981.

[5] D. J. Cavicchio. *Adaptive Search using simulated evolution*. PhD thesis, University of Michigan, Ann Arbor, 1970.

[6] K. A. DeJong. *Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Dept. Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975.

[7] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates, 1987.

[8] D. E. Goldberg and P. Segrest. Finite markov chain analysis of genetic algorithms. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 1–8. Lawrence Erlbaum Associates, 1987.

[9] R. L. Haupt. An introduction to genetic algorithms for electromagnetics. *IEEE Antennas & Propagation Mag.*, 37:7–15, April 1995.

[10] J. Hu and E. D. Goodman. The hierarchical fair competition HFC model for parallel evolutionary algorithms. In *Proc. of the 2002 Congress on Evolutionary Computation*, pages 49–54. IEEE Press, 2002.

[11] J. Hu, E. D. Goodman, and M. P. K. Seo. Adaptive hierarchical fair competition AHFC model for parallel evolutionary algorithms. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proc. of the Genetic and Evolutionary Computation Conference*, pages 772–779. Morgan Kaufmann, 2002.

[12] M. Hutter. Fitness uniform selection to preserve genetic diversity. In *Proc. of the 2002 Congress on Evolutionary Computation*, pages 783–788. IEEE Press, 2002.

[13] S. J. Louis and G. J. E. Rawlins. Syntactic analysis of convergence in genetic algorithms. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 141–151. Morgan Kaufmann, 1993.

[14] S. W. Mahfoud. Crowding and preselection revisited. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 27–36. North-Holland, 1992.

[15] E. Michielssen, J.-M. Sajer, S. Ranjithan, and R. Mittra. Design of lightweight, broad-band microwave absorbers using genetic algorithms. *IEEE Trans. Microwave Theory & Techniques*, 41(6):1024–1031, June/July 1993.

[16] Y. Rahmat-Samii and E. Michielssen, editors. *Electromagnetic Optimization by Genetic Algorithms*. Wiley, 1999.

[17] R. Tanese. Distributed genetic algorithms. In J. D. Schaffer, editor, *Proc. of the Third Intl. Conf. on Genetic Algorithms*, pages 434–439. Morgan Kaufmann, 1989.