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

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

ABSTRACT

To reduce the problem of premature convergence we define a new method for measuring an individual's age and propose the Age-Layered Population Structure (ALPS). This new measure of age measures how long the genetic material has been evolving in the population: offspring start with an age of 1 plus the age of their oldest parent instead of starting with an age of 0 as with traditional measures of age. ALPS differs from a typical evolutionary algorithm (EA) by segregating individuals into different age-layers by their age and by regularly introducing new, randomly generated individuals in the youngest layer. The introduction of randomly generated individuals at regular intervals results in an EA that is never completely converged and is always exploring new parts of the fitness landscape. By using age to restrict competition and breeding, younger individuals are able to develop without being dominated by older ones. Analysis of the search behavior of ALPS finds that the offspring of individuals that are randomly generated mid-way through a run are able to move the population out of mediocre local-optima to better parts of the fitness landscape. In comparison against a traditional EA, a multi-start EA and two other EAs with diversity maintenance schemes we find that ALPS produces significantly better designs with a higher reliability than the other EAs.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Design, Performance, Reliability

Keywords

Age, Computer-Automated Design, Premature Convergence, Evolutionary Algorithms, Open-ended Design

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
ACM 1-59593-186-4/06/0007.

1. INTRODUCTION

A common problem experienced in running an evolutionary algorithm (EA) is that after some number of evaluations the population converges to a local optima and no further improvements are made no matter how much longer the EA is run. What has happened is called *premature convergence*, which is that the existing genetic material in the population has converged such that the variation operators cannot produce new individuals which will move the population into a better part of the fitness landscape [1, 6, 8, 16]. To overcome the problem of premature convergence much work has has been done to investigate its causes and to reduce its effects. Many attempts at creating a more robust EA have been tried but so far none has successfully resulted in an improved EA with which naive users can expect to reliably find near-optimal solutions.

Perhaps the easiest methods for reducing the likelihood of premature convergence are to slow convergence through increasing the mutation rate, mutation size or 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. 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 simply increases the number of generations before it converges with the hope of increasing the likelihood of finding the fitness peak with global optima. With this approach deciding on the best population size is a challenge: using 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 be too large to be feasible.

Aside from increasing the mutation rate/size or increasing the population size, there have been various modifications to the canonical EA to better maintain genotypic diversity over the course of evolutionary search. Diversity of the population can be maintained by modifying the replacement strategy, such as with *preselection* [3], *crowding* [6], and *deterministic crowding* [17]. Another approach is the use of *sharing functions*, which modify the fitness of individuals based on their genotypic similarity [7]. Alternatively, 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 [18]. In addition, there are many newer methods which are variations on these basic techniques. While these methods

work to varying degrees, drawbacks are that methods which work through genotypic comparisons are better suited to bit strings than to more sophisticated representations, such as 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 that are in new areas of the fitness landscape. This method of getting around the premature convergence problem is a fairly common practice that is implicitly 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, and 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.

To better integrate new, randomly generated individuals we propose the Age-Layered Population Structure (ALPS), which uses a novel measure of age to restrict competition and breeding between individuals in the population. With ALPS an individual's age is a measure of how long its genetic material has been evolving in the population. Randomly created individuals start with an age of 0 and each generation in which an individual is used to produce an offspring its age is increased by one. Individuals created through mutation or recombination start with an age of 1 plus the age of their oldest parent, since their genotypic material has been evolving in the population for at least that many generations. The population is separated into multiple age-layers, with each age-layer having a maximum allowable age for individuals, with selection, breeding, and replacement restricted to adjacent layers. At regular intervals the individuals in the initial layer are replaced with new, randomly-generated individuals.

By structuring the population so that individuals only compete against other individuals of similar ages, young individuals are able to explore newly-discovered basins of attraction while being protected from older individuals that have had more time to evolve up their local optima. In addition, by competing against individuals of a similar age, individuals will only cluster about a local optima as long as it has the best fitness for similarly-aged individuals and the population will move off of it as younger individuals discover better local optima.

ALPS has similarities to other types of EAs, but the differences result in a significantly better algorithm. Prior to ALPS there have been many variations of evolutionary algorithms that have used some sort of age parameter [4, 5, 12, 13, 14]. These systems differ from ALPS in that the age value of offspring is a measure of how long that particular individual has been in the population thus individuals created through mutation or recombination start with an initial age value of 0 or 1. In contrast, with ALPS age is a measure of how long the genotypic material has been evolving in the population with offspring taking the age of 1 plus that of their oldest parent. In addition, with these other age-based systems randomly generated individuals are only created at the beginning of an evolutionary run so these systems are limited to finding solutions in the basin of attraction of the initial population.

Another EA that separates the population into different layers and regularly introduces new, randomly generated individuals in the initial layer is the Hierarchical Fair Competition (HFC) model [9]. The main difference between HFC and ALPS is that HFC uses fitness, instead of age, as the characteristic to segregate individuals. Using fitness to segregate individuals 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 have been made – Adaptive Hierarchical Fair Competition (AHFC) [11] and Continuous Hierarchical Fair Competition (CHFC) [10] – but these have not been shown to be significantly better than regular HFC.

To demonstrate the effectiveness of ALPS it is compared against a traditional EA, a multi-start EA, HFC, and an EA using deterministic crowding. The design problem on which these algorithms are compared is a challengin antenna design problem that is a variant of the antenna we evolved for NASA's ST5 Mission [15]. Evolution with a multi-start EA and with deterministic crowding both do better than the traditional EA and HFC, but all four are significantly outperformed by ALPS. Analysis of the search behavior of ALPS shows that by using age to restrict competition and breeding the genetic material of newly generated random individuals is able to discover and 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. Section 2 consists of a definition of our novel `age` measure and a description of the ALPS. Section 3 is a description of our experimental setup, the results are presented in section 4, and a discussion of our findings is given in section 5. In section 6 we describe how to combine ALPS with other diversity maintenance techniques and then conclude

2. THE ALPS PARADIGM

ALPS differs from traditional EAs by segregating individuals by a novel measure of age and using this to control breeding. The age-measure that we define for the ALPS-EA is a count of the number of generations in which an 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 of 1 plus the age value of their oldest parent since their genetic material comes from their parents and it has now been in the population for one more generation than it was with their parents. If an individual is copied to the next generation, such as through elitism, its age is increased by 1 if it was used as a parent in that generation and its age remains unchanged if it was not used as a parent. This increase to an individual's age even though it was not changed is because its genetic material was used in that generation and it forces the individual to compete with its offspring. Even if an individual is selected to reproduce multiple times in one generation its age is still only

Table 1: Different systems for setting the age-limits for each age-layer and the corresponding maximum age in each layer for an `age-gap` of 1.

	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

increased by 1 when it is copied to the next generation since age is a measure of how long the genetic material has been in the population and not how widely it is used.

To restrict competition and breeding among individuals the population consists of a number of age-layers, somewhat similar to the island-model EA [18]. 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 age limits, such as by using linearly, polynomially 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 ages for the layers are: 20, 40, 80, 180, 320, ... This allows some generations of evolutionary search to occur with the randomly-generated individuals in the first layer so that they may find and move into good basins of attraction before being pushed into the next layer. Alternatively, an ALPS-EA can be run with an infinite number of age-layers, with memory for new layers created as they are needed.

Evolution with the ALPS model is similar to a canonical EA, with a few exceptions. First, individuals can only breed with individuals from either 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 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` of 15 and an exponential aging system, only layer 0 is active for the first 15 generations. At generation 15 layer 1 is then used, layer 2 is used starting at generation 30, layer 3 is used starting at generation 60, layer 4 is used starting at generation 120, and so on. When a layer becomes open for use its individuals are created through offspring from parents selected from the previous layer. In addition, when an individual becomes too old for its current layer, it is compared against individuals in the next oldest layer and, if it is more fit than at least one of them, it replaces the least fit individual in that 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. An individual is only guaranteed to stay in the population forever it if is at the global optima, otherwise it will eventually be replaced as better individuals are evolved. This differs from HFC, in which individuals can stay in the population forever if their fitness is at the maximum of their fitness layer.

3. EXPERIMENTAL SETUP

To determine the effectiveness of the ALPS paradigm, experiments are performed comparing it to a canonical EA, HFC and to a multi-start EA. 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 introduces new, randomly generated individuals in its first layer thus by comparing ALPS against HFC the relative advantage of using age versus fitness to segregate and manage individuals in different layers can be inferred. A multi-start EA is included for comparison since it is both the most simple and most commonly used method for regularly generating a new batch of randomly generated individuals and evolving them. All systems use the same representation and variation operators with the only difference being in how the population of individuals is managed. The details of the different EAs will now be described followed by a description of the antenna design problem on which they are compared.

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 consisting of 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. 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. An elitism of 30 is used since this results in the same number of individuals being copied into the next generation as with ALPS and its use of an elitism of 3 in each of its 10 layers. New individuals are created with an equal probability of mutation or recombination.

A single trial with the multi-start EA consists of 100 runs of canonical EA for 100000 evaluations, and the best individual of a multi-start individual is the best individual of these 100 runs.

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. With this con-

figuration it will create as many random individuals as with ALPS.

3.2 Representation

The problem domain on which these algorithms compared is that of evolving antennas. 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 [15]. 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.

3.3 Antenna Optimization Problem

The antenna problem on which we compare the different EAs is similar to the X-band antenna we evolved for NASA's Space Technology 5 mission [15]. The goal for this particular 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 fitness function we use to evolve antennas for this problem is essentially the same as the one we used in evolving the ST5 antenna [15] and is a product of three components scoring the VSWR, gain values, and gain smoothness 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 1.5, reduced pressure at a values below 1.25 and then no pressure to go below 1.1:

$$f_v(v) = \begin{cases} v + 2.0(v - 1.25) & \text{if } v > 1.25 \\ v & \text{if } 1.25 > v > 1.1 \\ 1.1 & \text{if } v < 1.1 \end{cases}$$

$$vswr = f_v(VWSR_{2106MHz}) f_v(VSWR_{2288MHz})$$

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  θ = 5°i , φ = 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:

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

The objective of the EA is to produce an antenna design that minimizes F.

The Numerical Electromagnetics Code, Version 4 (NEC4) was used to evaluate all antenna designs [2]. 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. EXPERIMENTAL RESULTS

A total of 20 trials was performed with the canonical EA, AHFC and ALPS. After 100000 evaluations – approximately 100 generations, a typical length of an evolutionary run – these three EAs all have similar performance (figure 1.a). Already a leveling off in performance appears to be occurring and runs with traditional EAs may be stopped after this many evaluations since they generally have converged and there is little improvement to be had from running

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

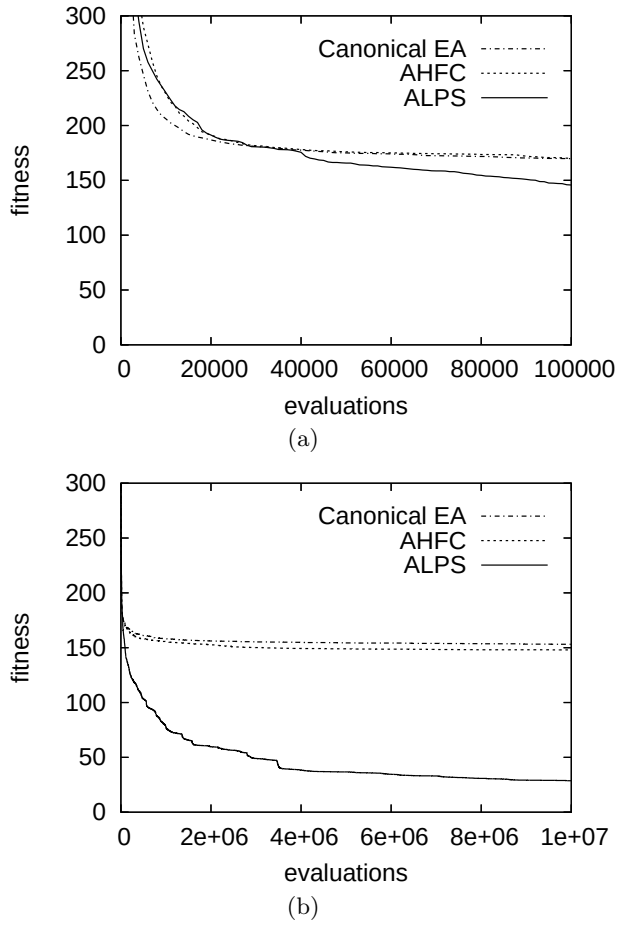


Figure 1: A plot of the best individual in the population for the three different EAs averaged over 20 trials each: (a) after the first 100000 evaluations; (b) after 10 million evaluations.

the EA longer. Continuing evolution a further 9.9 million evaluations shows that both the canonical EA and AHFC have made few improvements (figure 1.b) yet with ALPS there were large gains in fitness. To summarize, after 10 million evaluations the averaged results of these trials are (mean±s.e.): standard EA, 153.1±16.4; AHFC, 148.0±35.4; and ALPS, 28.7±10.8. 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 Mann-Whitney test, the performance difference between the standard EA and AHFC is not significantly different, with $P \geq 0.05$.

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 produce 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 with which to control breeding than is fitness. Finally, all solutions found with ALPS are better than all solutions found with either AHFC or a regular EA, which suggests that a single run of ALPS is better than performing multiple shorter runs of a traditional EA.

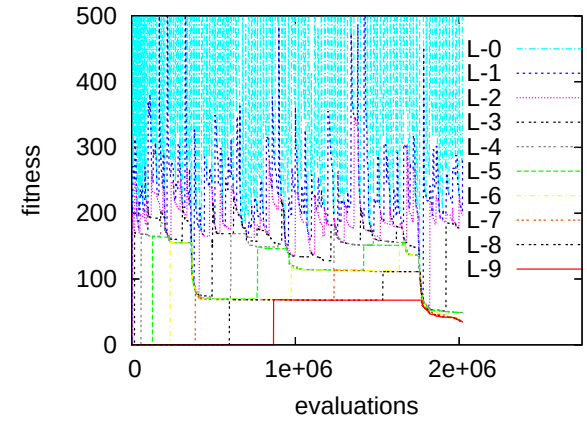
To determine how well ALPS compares against performing multiple short runs with a traditional we ran an additional set of experiments using a multi-start paradigm in which a single trial consists of 10 million evaluations divided into 100 separate runs of 100000 evaluations with the canonical EA. The best individual from a single multi-start EA trial is the best individual found from the 100 separate runs. The results of the multi-start EA trials are 47.6±15.3. Using a two-tailed Mann-Whitney test finds that the superior performance of ALPS over the multi-start EA is highly significant, with $P < 0.001$.

5. DISCUSSION

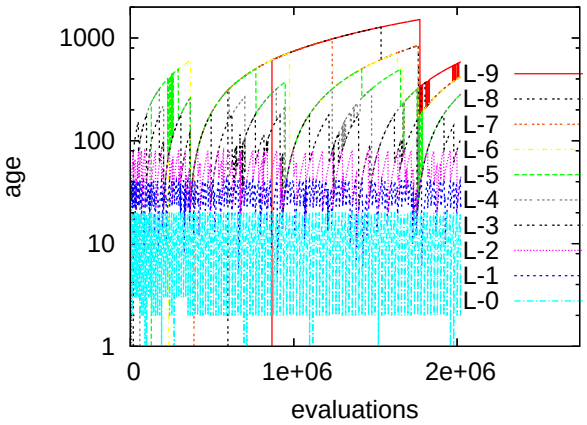
An intuitive understanding of why evolution with ALPS works so well can be gained by examining the fitness and age values of the best individual in each layer over the course of evolution. The two graphs in figure 2 show the first 2 million evaluations of one of the trials with the ALPS paradigm: the graph in figure 2.a consists of a plot of the fitness of the best individual in each layer of the population, and the graph in figure 2.b consists of a plot of 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, L-0 starts with a very poor fitness, improves to a fitness value centered around 250 over the course of 20 generations and then starts afresh at something higher then 500. In figure 2.b the plot of the age of this best individual in layer L-0 shows that 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 the layer it is in 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 that layer the rest of the individuals in the layer that are genotypically similar are also in the process of being aged-out of that layer. At this point the layer is reset with new genetic material from the best individuals of the previous layer, hence the smooth arcs formed from the plots of adjacent layers show a continuous line of development of both fitness and age of genotypic material progressing through multiple age-layers.

To help understand why evolution with ALPS is better able to continue finding new and better solutions, even when run for a very large number of generations, in figure reffig:alps4-ex2 we examine a close-up of the evolutionary run from 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 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



(a)



(b)

Figure 2: A plot of the fitnesses and ages of the best individuals of over the first 2 million evaluations of an EA run using ALPS.

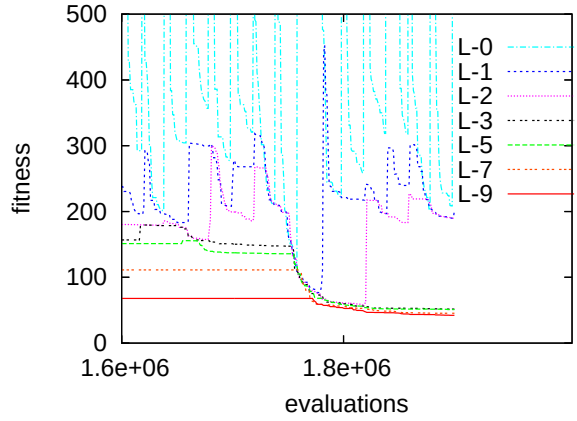
seen that the offspring of the individual that was randomly generated sometime around 1.7 million evaluations quickly progressed up the layers from L-0, to L-1, all the way to L-5, after several thousand evaluations (which is several generations since there are 1000 individuals in the population). About the time 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) do 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.

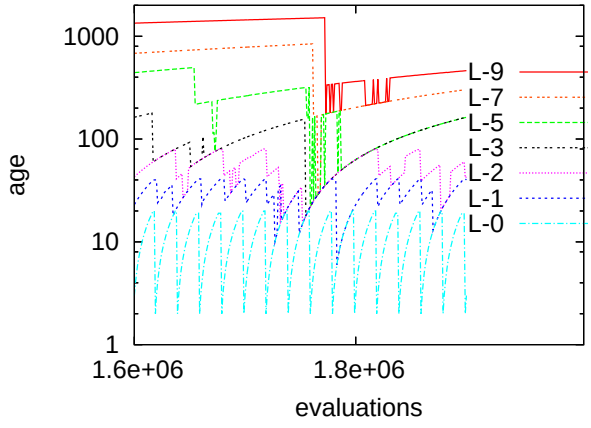
Instead of assigning the age of the oldest parent to offspring created through recombination, with this second age measure, age_2, individuals are given the age of their youngest parent. The graph in figure 3.c is a plot of the age_2 of the best individual in each layer of the same run as the graphs 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_2 of the best individual in the top age-layer has an age_2 of 31, using this second age measure. This means that new best individual is a descendant of an individual that was randomly generated some 31 generations previous. In fact, using this second age-measure, the maximum age_2 of the best individuals in any of the age-layers seldom goes higher than 100, which means that the genetic material of the new, randomly generated individuals is being transferred to the rest of the population in the other age-layers.

The combination of genetic material from recently generated random individuals and individuals whose genetic material has been in the population for many generations is not achievable with a multi-start EA and may be why ALPS outperforms such an algorithm. With a multi-start EA in which all the genetic material from previous runs is discarded individuals from the new run cannot use the genetic material from them. Alternatively, in a multi-start EA in which some of the best individuals are kept from a previous run, their offspring quickly take over the population since they have much better fitness scores than the randomly generated individuals against which they are competing. With the ALPS paradigm new and old individuals are able simultaneously evolve without the old individuals killing off the new ones thereby enabling the ALPS-EA to successfully balance both exploitation and exploration.

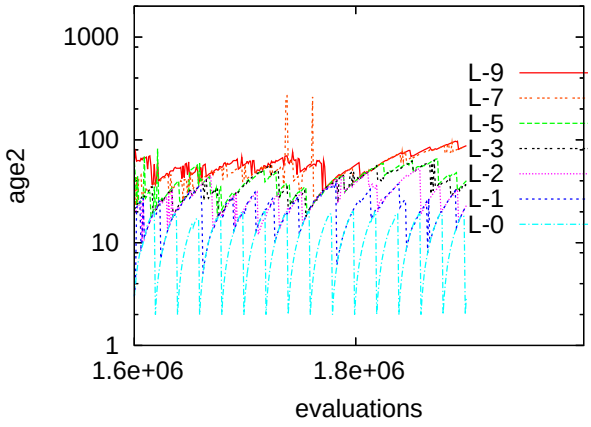
The behavioral difference between ALPS and HFC can be seen by examining fitness and age graphs from a run with the HFC algorithm, figure 4, and comparing them with similar graphs for ALPS. The first graph, figure 4.a, shows the sequential layering of fitness values for each fitness-layer in the population, 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 in the graphs in figures 2 and 3. The second graph, figure 4.b, plots the ages of the best individual in each fitness layer of an AHFC run. Similar to ALPS, the age of individuals in the lowest fitness-layer oscillates starting from an age of 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 the other layers 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 its fitness layers. That AHFC is not successful in enabling new individuals their offspring to stay and evolve in the population demonstrates that age is the better attribute to restrict competition and breeding than is fitness.



(a)

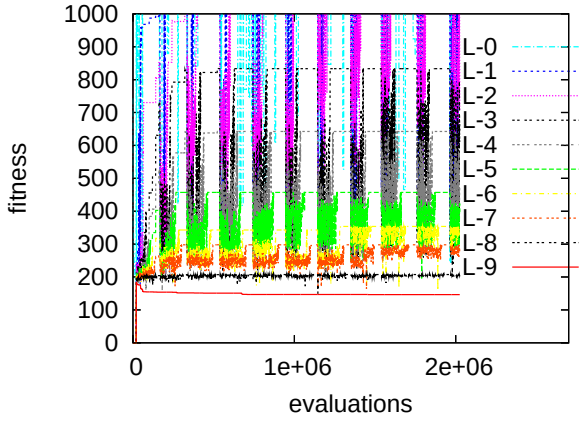


(b)

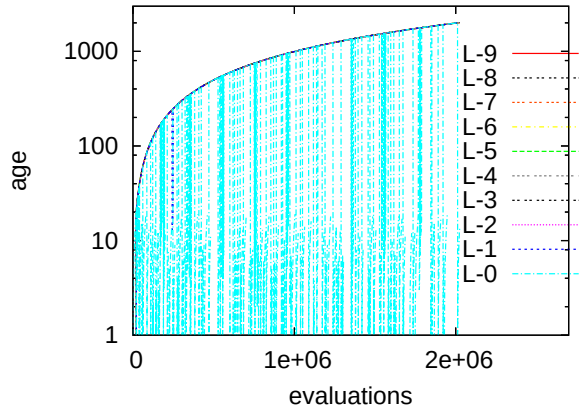


(c)

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 age_2, a different measure of age.



(a)



(b)

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

6. HYBRIDIZING ALPS

While ALPS is good at using age to shepherd the development of new, randomly-generated individuals, it may be that this is not sufficient to find near optimal results. One way to try improving ALPS is to combine it with another diversity maintenance technique. From past work on evolving antennas an EA using deterministic crowding (DC) had been found to work best [15]. With deterministic crowding, new individuals are inserted into the population by comparing an offspring against its parent (or most similar parent in the case of recombination) and replacing that parent if it is more fit. To combine DC with ALPS, DC was used in each of the different age-layers of ALPS and, each time a parent was replaced by its child, the parent was then tested against the least fit individual of the next oldest age-layer and replaced it if it had better fitness. Out of 20 trials with both of these algorithms the results are: DC 97.1±54.2; ALPS-DC 22.1±7.8. Regular ALPS outperforms DC and the difference is statistically significant with $P < 0.01$ using a two-tailed Mann-Whitney test. The superior performance of the ALPS-DC combination over both DC alone as well as the multi-start EA is found to be highly significant, with $P < 0.001$ using a two-tailed Mann-Whitney test. The difference in performance between ALPS-DC and ALPS with

tournament selection is not significant, with a two-tailed Mann-Whitney test finding $P > 0.05$.

7. CONCLUSION

In this paper we have defined a new measure of the age of an individual 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 based on 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 handful of other EAs and in all cases it was found to be significantly better in performance. The large difference in performance between ALPS and both a canonical EA and an EA using deterministic crowding demonstrates that regularly introducing random individuals into the population can produce a better EA. Similarly, the large performance advantage of ALPS over the Hierarchical Fair Competition (HFC) model – another EA which regularly introduces random individuals into the population but uses fitness as the attribute to restrict competition and breeding – demonstrates that the novel measure of age defined here is a better attribute to control breeding than is fitness. Similarly, the significant performance advantage of ALPS over a multi-start EA demonstrates that it can be more advantageous to integrate the regular introduction of random individuals into the population with the ALPS approach than to have many disjoint EA runs.

It is hoped that by using age as a means to control breeding between individuals in the population, future variants of the ALPS algorithm – such as by hybridizing ALPS with another diversity maintenance technique – will provide even better ways to prevent premature convergence.

8. REFERENCES

[1] J. E. Baker. Adaptive selection methods for genetic algorithms. In J. J. Grefenstette, editor, *Proc. of the First Intl. Conf. on Genetic Algorithms*, pages 101–111, 1985.

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

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

[4] G. Chakraborty and B. Chakraborty. Ideal marriage for fine tuning in GA. In *Systems, Man, and Cybernetics Conference Proceedings*, pages 631–636. IEEE Press, 1999.

[5] H.-J. Cho, S.-Y. Oh, and D.-H. Choi. Fast evolutionary programming through search momentum and multiple offspring strategy. In *Proceedings of the International Conference on Evolutionary Computation*, pages 805–809. IEEE Press, 1998.

[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, *Proc. of the Second Intl. Conf. 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, *Proc. of the Second Intl. Conf. on Genetic Algorithms*, pages 1–8. Lawrence Erlbaum Associates, 1987.

[9] 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.

[10] J. Hu, E. D. Goodman, and K. Seo. 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.

[11] J. Hu, E. D. Goodman, K. Seo, and M. Pei. Adaptive hierarchical fair competition AHFC model for parallel evolutionary algorithms. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 772–779. Morgan Kaufmann, 2002.

[12] A. Huber and D. A. Mlynski. An age-controlled evolutionary algorithm for optimization problems in physical layout. In *International Symposium on Circuits and Systems*, pages 262–265. IEEE Press, 1998.

[13] J.-H. Kim, J.-Y. Jeon, H.-K. Chae, and K. Koh. A novel evolutionary algorithm with fast convergence. In *IEEE International Conference on Evolutionary Computation*, pages 228–29. IEEE Press, 1995.

[14] N. Kubota, T. Fukuda, F. Arai, and K. Shimojima. Genetic algorithm with age structure and its application to self-organizing manufacturing system. In *IEEE Symposium on Emerging Technologies and Factory Automation*, pages 472–477. IEEE Press, 1994.

[15] J. D. Lohn, G. S. Hornby, and D. S. Linden. Rapid re-evolution of an X-band antenna for NASA's space technology 5 mission. In T. Yu, R. L. Riolo, and B. Worzel, editors, *Genetic Programming Theory and Practice III*, volume 9 of *Genetic Programming*, chapter 5, pages 65–78. Springer, Ann Arbor, 2005.

[16] 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.

[17] 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.

[18] 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.