Binomics: Where Metagenomics meets the Binary World

Inman Harvey[1,2], Nicholas Tomko[1,2]

[1]CCNR, University of Sussex, Brighton, U.K.
[2]Evolutionary and Adaptive Systems Group, University of Sussex
inmanh@gmail.com, nick.tomko@gmail.com

Abstract

Artificial Life and Evolutionary Computation studies have until now failed to model the symbiotic evaluation methods and the extensive amounts of horizontal gene transfer that are starting to be recognized in recent Metagenomic approaches to understanding microbial populations. Examples can be seen, in Learning Classifier Systems, and the SANE algorithm, of symbiotic evaluations; the Microbial Genetic Algorithm (GA) introduced horizontal gene transfer. Here for the first time these two are brought together in the Binomic GA, which is shown to perform well in a series of trials. It is proposed that Binomics, defined as computational algorithms inspired by Metagenomic studies, forms a potentially fruitful field of study waiting to be investigated.

Introduction

For many years our conventional understanding of Darwinian evolution has been dominated by the idea of species of individuals, where those individuals favoured by selection become parents and pass on their genes to their offspring. Although selection takes place at the individual level, this vertical transmission of genetic material leads to an identifiable entity at the species level that has the capacity to adapt over time. Our models of artificial evolution, such as Genetic Algorithms (GAs), have typically followed this picture.

But in the last decade or so some biologists have started to realize that a significant part of evolution on this planet – in particular bacterial evolution – has important mismatches with this picture. There can be a significant amount of horizontal gene transmission between different individuals. As a result, much of their functionality can be passed on from their neighbours rather than inherited from parents. This makes the concept of a species in such circumstances rather looser than previously thought. Further, the fitness of a population of diverse bacteria floating in the sea may depend significantly on their local collective symbiotic functionality, rather than simply on the individual fitness of each.

Studies of the collective genetic properties of such a diverse population have come to be known as Metagenomics. Research into these natural processes has been driven by recent major advances in gene sequencing techniques. Analysis of Metagenomic results now needs new tools from complex systems theory, and already some people have started applying ideas from Artificial Life (AL) and Evolutionary Computation (EC). What has been conspicuously missing so far has been a movement of ideas in the other direction. This paper is primarily a position paper calling for new developments in AL and EC, as applied to synthetic problems, to be inspired by these new discoveries in the natural world. Drawing on biologists' use of the '-omics' suffix to refer to the collective properties of a totality, we propose *Binomics* as a new sub-field where ideas from Metagenomics are applied to applications in the binary computational world.

We start with a brief review of Metagenomics, and then a survey of those main techniques within AL that do already distil some relevant ideas. We focus on symbiotic evaluation, where individuals are evaluated collectively; specifically we look at Learning Classifier Systems (LCS) and the SANE algorithm for artificial neuro-evolution. Then we consider horizontal gene transfer, looking at the Microbial GA. We note that to date nobody seems to have combined symbiotic evaluations with horizontal gene transfer.

So we do just this with a proposal for a *Binomic Genetic Algorithm*. Although this is primarily a position paper, we can demonstrate its performance in a series of trials and compare with other evolutionary techniques. These are preliminary studies, but gratifyingly we can report that in these trials the Binomic GA outperformed the competitors by at least an order of magnitude. We suggest that this is a fruitful new area for further study, and discuss the types of applications where the particular properties of a Binomic GA could be beneficial.

Metagenomics

As a very recent field, most of the reporting on Metagenomics comes in specialised technical research papers. Useful overviews for a more general audience include Handelsman (2004), a report by the Committee on Metagenomics (2007), and Eisen (2007).

Previously our understanding of microbes has been based on studying rather few samples. In order to perform reproducible scientific experiments, well-defined species have been used, often with great care taken to culture them in the lab in isolation to ensure their purity. It is typically assumed that the test-tube is full of a single species that is genetically well-defined. It has been belatedly realized that such assumptions may not hold true in the real world.

In microbial communities there may often be large functional differences between close relatives; further, horizontal gene transmission means that many functions

(chemical cycles) typically performed by one species may be also performed by very different species. Microbes such as bacteria do not undergo sexual reproduction, but reproduce by binary fission. But they have a further method for exchanging genetic material, *bacterial conjugation*. Chunks of DNA, plasmids, can be transferred from one bacterium to the next when they are in direct contact with each other. Whereas the genomes of different humans vary by around 0.1%, different members of what may conventionally be termed a microbial species (or phylotype) can differ by up to 30%. It now makes conceptual sense – and technical developments make it possible – to perform shotgun sequencing of a whole bucketful of microbes taken from the Sargasso Sea (Venter et al., 2004) and consider the metagenomic sequence of the whole community, together with the functions that such a community collectively performs. Shotgun analysis involves breaking up the DNA randomly into small segments that are individually sequenced; then using computational methods, by seeking overlaps in these fragments, they are built up again into a complete sequence.

There are 10 times as many microbial cells in a human body than there are human cells; the human metagenome contains perhaps a hundred times more genes than the human genome (Qin et al. 2010). Many such bacteria are essential for our human well-being, and in turn they rely on us to provide them with an appropriate environment.

Comparisons: Metagenomics and AL, EC

Horizontal gene transfer rarely features in EC, though we give one example below with the Microbial GA. We can analyse the real world of bacteria floating in the sea in terms of two separate fitness criteria: internal (individual) and external (symbiotic). Firstly, each individual organism (given a sustaining environment) has to have the appropriately functioning internal mechanisms to *individually* survive. Secondly and collectively, their interactions -- the inputs and outputs of all such organisms -- must have an appropriate fit with their neighbours, so that they can *collectively* survive. In artificial evolution, we can choose to take the internal fitness criteria for granted and focus our attention solely on the external criteria, of fit to the environment. If we want to follow the Metagenomic metaphor, we shall be evolving individual entities whose value (as assessed by a fitness function) will depend on how they cooperate to tackle some task. Penn and Harvey (2004) demonstrated how ecosystem-level evolution can take place *without* genetic change in the component species, but here we want to focus on ecosystem-level evolution driven *by* genetic change.

We now discuss two areas of EC where relevant work has been done in the next sections on LCS and SANE.

Learning Classifier Systems

Learning Classifier Systems (LCS) were devised by John Holland (Holland 1976, Holland and Reitman, 1978) as a means of using a GA to do just this; for an introduction see Bull (2004). The *classifiers* are condition-action rules, typically expressed as a string of symbols, where the first part represents a template that expresses the conditions under which this classifier could match a possible input string; and the second part represents the output string of the classifier when the condition is met. Inputs to a classifier may come from the external task (e.g. they could come from sensors if this is a robot control task, or from a visual array if the task is pattern classifying), or come from other classifiers; outputs from a classifier could be to the external solution (e.g. strings interpreted as robot motor actions) or to other classifiers. Internal message-boards can be used for communication between the classifiers.

As Bull (2004) comments:

> It is important to note that the role of the GA in LCS is to create a cooperative set of rules which together solve the task. That is, unlike a traditional optimisation scenario, the search is not for a single fittest rule but a number of different types of rule which together give appropriate behaviour. The rule-base of an LCS has been described as an evolving ecology of rules - "each individual rule evolves in the context of the external environment and the other rules in the classifier system." [Forrest & Miller, 1991].

This raises a major issue in deciding how to assign a fitness to each rule, when this can only be evaluated in the context of a collective ecology. Two main approaches have been developed for LCS, named for the places where they were first proposed.

Pittsburgh LCS. In this approach each individual in the evolving population is a complete set of rules or classifiers. The rules play a role more similar to that of genes in an organism than being themselves independent organisms. In this way the problem of assigning value to each rule is avoided. The GA reproduces, with recombination and mutation, from the fitter rule sets.

Michigan LCS. In this approach the individuals in the population are the individual rules or classifiers themselves. During evolution, any of the individual rules can be operational, and this needs some arbitration mechanism to decide between them if some are matching in their input conditions but potentially conflicting in their outputs. Further complications arise from deciding how to allocate fitness to each rule that is actually operational, bearing in mind that only the collective can be evaluated. In some cases there may be a temporal element, in that the consequences of one specific condition-action rule may not be immediately apparent, but only become evident due to later knock-on consequences.

Many different methods have been proposed for tackling these issues, including auctions with specificity-based arbitration mechanisms to allow default hierarchies to form, and bucket-brigade algorithms for the temporal credit-assignment problem. This has resulted in many different flavours of Michigan LCS.

Implicit Niching in LCS

In a typical evolutionary algorithm such as a GA, we can expect selection to drive the population in the direction of genetic convergence, where it consists almost entirely of

copies, or near-copies, of the single fittest individual. But in the context of an LCS, where fitness will likely depend on the co-existence of several different individuals performing sub-functions of the whole task, such loss of diversity is undesirable. There is a need to find and maintain a diverse and cooperative set of classifiers. Some form of *niching* in the population is desirable. One approach to achieving this is through an island model, where distributed populations are separated into different demes.

Another approach is through *fitness sharing* (Goldberg and Richardson 1987), which requires some distance metric or similarity measure (either genotypic or phenotypic) between any two individuals. By using suitable methods to adjust the fitnesses of any individual according to how many other similar individuals there are nearby in this metric space, there is a tendency for the population to spread out over multiple peaks or niches in the fitness landscape; thus diversity is maintained. It can be shown that LCS models where fitness is shared amongst cooperating individuals can produce *implicit niching* (Horn et al. 1994), and this will be discussed further with the Binomic GA.

Comparisons: LCS and Metagenomics

We can relate the condition-action classifiers to the bacteria in the sea. The evaluation of the symbiotic functionality of groups of these does indeed reflect, in the context of artificial evolution, some aspects of what we observe in real world Metagenomics. The Michigan style of LCS does, at the expense of often complex auction and bucket-brigade schemes, manage the evaluation of individual 'organisms' (classifiers) that can only function effectively as part of a set. The evolutionary aspect is limited to the vertical genetic transfer between generations that is traditional with GAs.

Symbiotic Evaluations: SANE

There is a different perspective on evaluating different individuals on the basis of their group performance, taken by Moriarty and Miikkulainen (1996, 1999) in their proposal of the SANE algorithm. SANE stands for Symbiotic, Adaptive Neuro-Evolution, and this is one approach to evolving Artificial Neural Networks (ANNs). The motivation is described thus (Moriarty and Miikkulainen, 1999):

> SANE incorporates the idea of diversity into neuro-evolution. SANE evolves a population of neurons, where the fitness of each neuron is determined by how well it cooperates with other neurons in the population. To evolve a network capable of performing a task, the neurons must optimize different aspects of the network and form a mutualistic symbiotic relationship. Neurons will evolve into several *specializations* that search different areas of the solution space.

In an example implementation, they show a simple ANN with 2 layers of connection weights, from Input to Hidden neurons and from Hidden neurons to Outputs. They treat each Hidden neuron, together with its incoming and outgoing connections, as a member of the evolving population. Figure 1 shows how a complete network could be formed from e.g. 3 such Hidden neurons selected at random from the population. The network as a whole is evaluated on some required task, and the network's score is added to the fitness of each Hidden neuron that it contains. Thereafter, the selection, replication, crossover and mutation of members of the population is carried out by conventional GA methods.

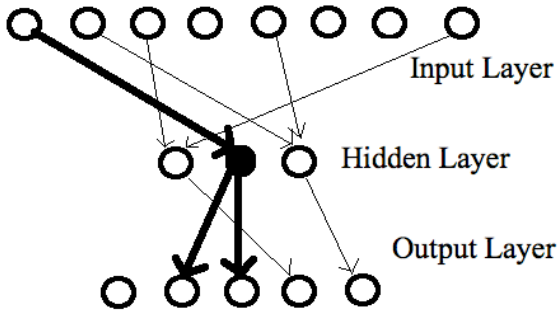

Figure 1. Each Hidden Layer neuron, with its associated incoming and outgoing connections (e.g. the highlighted central one with its links), is a member of the population. Here 3 such neurons combine to make a complete feedforward ANN.

Moriarty and Miikkulainen (1999) report that this implementation of SANE works well on such simple ANNs. They also comment that it is feasible to extend this approach to different neuron encodings, and to diverse network architectures including recurrency.

Comparisons between SANE and Metagenomics

Much as we did with the condition-action classifiers of LCS, we can relate the Hidden neurons (with incoming and outgoing connections) to the bacteria in the sea. Once again, these are only evaluated in the context of a group, which is why it has been called symbiotic (artificial) evolution. Implicit niching is again important. We can characterize this approach in much the same way as LCS, in that there are similarities in this symbiotic evaluation to some aspects of what we observe in real world Metagenomics; the evolutionary aspect is still restricted to the vertical gene transfer of conventional GAs.

Horizontal Gene Transfer: Microbial GA

Significant features of evolution that were under-recognised before Metagenomic studies included the symbiotic nature of functionality of groups of organisms, and the prevalence of horizontal gene transmission. In Genetic Algorithms, vertical genetic transmission has been very much the norm. An exception has been the Microbial GA (Harvey 2001, 2010 In Press) that we review here in a reprise of relevant sections of Harvey (2010). This is the result of stripping away as much as possible from a traditional GA, whilst maintaining the bare essentials of a population with Heredity, Variation and Selection. The Microbial GA uses Tournament Selection within a Steady State GA, hence we introduce these concepts first.

Steady State GAs

Traditionally GAs were first presented in generational form. This roughly corresponds to some natural species that has just one breeding season, say once a year, and after breeding the parents die out without a second chance. There are many natural species that do not have such constraints, with birth and death events happening asynchronously across the population. Hence the Steady State GA, which in its simplest form has as its basic event the replacement of just one individual from P by a single new one. One reason for using Steady State in a minimalist GA is that it allows for a very simple implementation of selection.

Tournament Selection

There are many problems with the traditional GA method of fitness-proportionate selection that are avoided by using some form of rank-based selection. In this, once all the members of the population have been evaluated, each fitness is rescaled on the basis of their relative ranking. A common choice made is to allocate (at least in principle) 2.0 reproductive units to the fittest, 1.0 units to the median, and 0.0 units to the least fit member, similarly scaling pro rata for intermediate rankings; this is linear rank selection. The probability of being a parent is now proportional to these rank-derived numbers, rather than to the original fitness scores.

It is possible to achieve equivalent results to this through tournament selection. If two members of the population are chosen at random, their fitnesses compared (the 'tournament'), and the Winner selected, then the probability of the Winner being any specific member of the population exactly matches the reproductive units allocated under linear rank selection.

Who to Breed, Who to Die?

Selection can be implemented in two very different ways; either is fine, as long as the end result is to bias the choice of those who contribute to future generations in favour of the fitter ones. The usual method in GAs is to focus the selection on who is to become a parent, whilst making an unbiased, unselective choice of who is to die. In the standard Generational GA, every member of the preceding generation is eliminated without any favouritism, so as to make way for the fresh generation reproduced from selected parents. In a Steady State GA, once a single new individual has been bred from selected parents, some other individual has to be removed so as to maintain a constant population size; this individual is often chosen at random, again unbiased.

Some people, however, will implement a method of biasing the choice of who is removed towards the less fit. It should be appreciated that this is a *second* form of selective pressure, that will compound with the selective pressure for fit parents and potentially make the combined selective pressure stronger than is wise. In fact, one can generate the same degree of selective pressure by biasing the culling choice towards the less fit (whilst selecting parents at random) as one gets by the conventional method of biasing the parental choice towards the more fit (whilst culling at random).

This leads to an unconventional, but effective, method of implementing Tournament Selection. For each birth/death cycle, generate one new offspring with random parentage; with a standard sexual GA, this means picking both parents at random, but it can similarly work with an asexual GA through picking a single parent at random. A single individual must be culled to be replaced by the new individual; by picking two at random, and culling the *Loser*, or least fit of the two, we have the requisite selection pressure.

Going further, we can consider a yet more unconventional method, that combines the random undirected parent-picking with the directed selection of who is to be culled. Pick two individuals at random to be parents, and generate a new offspring from them; then use the same two individuals for the tournament to select who is culled -- in other words the weaker parent is replaced by the offspring.

It turns out that this is easy to implement, and is effective. This is the underlying intuition behind the Microbial GA.

Microbial Sex: Horizontal Gene Transmission

We can reinterpret the Tournament described above, so as to somewhat resemble bacterial conjugation. If the two individuals picked at random to be parents are called A and B, whilst the offspring is called C, then we have described what happens as C replacing the weaker one of the parents, say B; B disappears and is replaced by C. If C is the product of sexual recombination between A and B, however, then ~50% of C's genetic material (give or take the odd mutation) is from A, ~50% from B. So what has happened is indistinguishable from B remaining in the population, but with ~50% of its original genetic material replaced by material copied and passed over from A. We can consider this as a rather excessive case of horizontal gene transfer from A (the fitter) to B (the weaker).

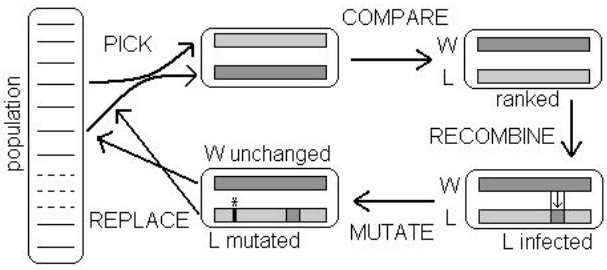

Figure 2. Sketch of the Microbial GA. The genotypes are represented as a pool of strings. One cycle of the GA is represented by the operations PICK (at random), COMPARE (their fitnesses to determine Winner = W, Loser = L, RECOMBINE (some proportion of Winner's genetic material 'infects' the Loser) and MUTATE (the revised version of Loser.

The Microbial GA in schematic form

We now have the basis for a radical, minimalist revision of the normal form of a GA, although functionally, in terms of Heredity, Variation and Selection, it is performing just the same job as the standard version. This is illustrated in Figure 2. Here the recombination is described in terms of 'infecting' the Loser with genetic material from the Winner, and we can note that this rate of infection can take different values. In bacterial conjugation it will typically be rather a low percentage that is replaced or supplemented; if instead we want to reproduce the typical effects of sexual reproduction, as indicated in the previous section, this rate should be ~50%. But in principle we may want, for different effects, to choose any value between 0% and 100%.

From a programming perspective, this cycle is very easy to implement efficiently. For each such tournament cycle, the Winner genotype can remain unchanged within the genotype-array, and the Loser genotype can be modified (by 'infection' and mutation) *in situ*. We can note that this cycle gives a version of 'elitism' for free: since the current fittest member of the population will win any tournament that it participates in, it will thus remain unchanged in the population -- until eventually overtaken by some new individual even fitter. Further, it allows us to implement an effective version of geographical clustering for a trivial amount of extra code.

Microbial GA: with a Trivial Geography

For some purposes we may not want a panmictic population, and instead constrain the operations of choosing tournament participants, and hence exchange of genetic material, to be within some local geographical distribution, perhaps within demes. This allows for more genetic diversity to be maintained across sub-populations. Spector and Klein (2005) note that a one-dimensional geography, as in Figure 3 where the population is considered to be on a (virtual) ring, can be as effective as higher dimensional versions. If we consider our array that contains the genotypes to be wrap-around, then we can implement this version by, for each tournament cycle: choose the first member A of the tournament at random from the whole population; then select the next member B at random from a deme, or sub-population that starts immediately after A in the array-order. The deme size D, <= P, is a parameter deciding just how local each tournament is.

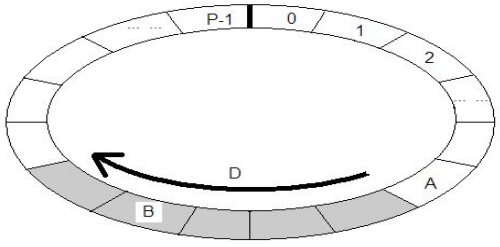

Figure 3. The population is geographically distributed on a ring, numbered from 0 to P-1. For a tournament, A is picked at random from the whole population; then B is picked at random from the deme (here of size D=5) immediately following A.

Comparisons: Microbial GA and Metagenomics

The Microbial GA is a deliberately minimalist version of a classical GA, but re-described in terms of horizontal gene transmission. The parameter that determines what proportion of genetic material is copied from Winner to Loser after each tournament can be varied according to need. Setting this at 50% gives the closest analogy to a classical GA, but other values may be of interest. Low 'rates of infection' may reflect typical values of gene transfer seen in real world Metagenomic studies; setting the rate to 100% would correspond to replication by fission of the Winner, since the Loser then becomes an identical copy. The addition of geographical demes could be tailored to correspond to any model of local interactions between, for example, bacteria swimming in the sea.

So this is a rare example of a GA with horizontal gene transmission. If we want to replicate in an evolutionary algorithm more of the essential properties that we see in Metagenomic studies of bacteria in a sea, then what is still missing is the aspect of assessing the fitness of each member of the population in some symbiotic or communal fashion.

Binomic GA

We now introduce a Binomic GA, that combines the symbiotic evaluation methodology of SANE with the horizontal gene transfer of the Microbial GA. We start with an outline of the general requirements, and then illustrate in the context of evolving Artificial Neural Networks.

General Requirements

We shall be evolving the equivalent of a Sargasso Sea (*Sea*) of individual organisms (*Orgs*). *Orgs* are not evaluated in isolation, but only as part of a randomly chosen subset of the *Sea*, a *Bucket*; such a *Bucket* may be drawn from a local area (or *Deme*) or from the whole of the *Sea* (Figure 4). The fitness function is used to evaluate a *Bucket* as a whole, and this fitness is passed on equally to all members of that *Bucket*. It is used to update the current fitness of each such *Org*, on the basis of New_Org_fit = R*Bucket_fit + (1.0-R)*Old_Org_fit. With an appropriate choice of R (0.0<R<1.0), the effective

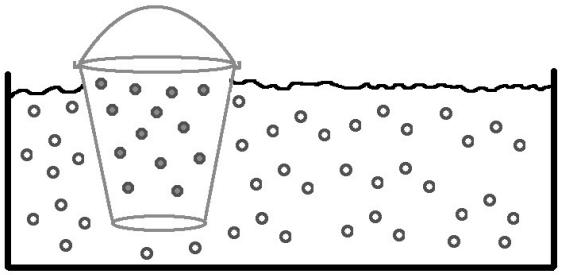

Figure 4. A Bucketful of Orgs is evaluated as a whole, and the resulting fitness assigned equally to all in that Bucket. Such a Bucket can be drawn locally from an area of the Sea, or (with a 'well-mixed' Sea) drawn at random.

fitness of each *Org* has any variance smoothed over several recent evaluations of different *Buckets* that it happens to have featured in, whilst still tracking any general changes in its environment.

Figure 5 sketches the Binomic GA. As with a Microbial GA, genetic changes in the *Sea* of *Orgs* are driven via a Tournament involving selecting two *Orgs* at random, comparing their currently stored fitnesses (as calculated via *Buckets*), and designating one as *W* = Winner, the other as *L* = Loser. *W* will remain unchanged, and *L* is the single *Org* that gets changed as a result of this tournament.

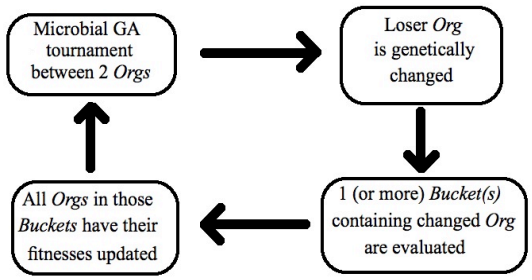

Figure 5. The Binomic GA.

With some small probability V (for vertical gene transmission) *L* is made an identical clone of *W* and also inherits its fitness, that is then updated as below. Otherwise (horizontal gene transmission) some proportion REC of the genes of *W* are copied over so as to replace those genes in *L*.

The tournament is completed by mutating *L*, and then evaluating a number (1 or more) of random *Buckets* that each contain *L*. In this way its inherited fitness is updated, along with the fitnesses of those other *Orgs* that happened to share those *Buckets*. The mutation will be a limited change in the *Org*, either retaining its functionality whilst making a small change in some parameter value (such as, in the case of ANNs, a neural network weight) or making a small change in functionality (such as, with ANNs, adding or deleting a connection between nodes). This should become clearer with a worked example that follows a brief explanation of niching.

Implicit Niching

To understand how implicit niching can occur in an algorithm like this, let us illustrate with a cartoon example. Suppose a population has 4 types of entity, *bread*, *butter*, *jam* and diverse *garbage*. The only collection that has any value is a *bread*+*butter*+*jam* sandwich. When fitnesses are allocated through assessing the value of a *Bucket* of such individuals, we can see that *garbage* would tend to decrease. But further, consider what happens if one of the useful components, e.g. *jam*, is in much shorter supply than the others. Then as a consequence of some *Buckets* containing *bread* and *butter* but no *jam* (and hence valueless), the relative fitness allocated to those individuals will decrease; whereas the relative fitness of *jam* (that will under these circumstances almost always complete a sandwich) increases. In this way, all these different component parts will tend towards similar proportional representation in the population as a whole.

Altruism and cheating. Any procedure that uses some form of group selection raises concerns about the possibility of cheating. If fitness is allocated collectively, why should an individual altruistically contribute to the common good, why not benefit from others' efforts whilst making no contribution itself? This potential pitfall is avoided by the use of *Buckets* allocating fitness within a temporary local subset of the whole *Sea*, even if that subset is taken at random from the whole well-mixed *Sea*. Restricting *Buckets* to (overlapping) local regions within a geographically distributed *Sea* provides yet more pressure to eliminate cheats and garbage.

Evolving ANNs with a Binomic GA

The SANE algorithm, discussed above, implemented the equivalent of *Orgs* as subsets of a 3-layer ANN, each one based on a single node in the middle (Hidden) layer with connections and weights to genetically specified Input or Output nodes. We can generalize this to ANNs of arbitrary topology (including recurrent networks such as CTRNNs) by first making each *Org* in principle equivalent to the whole fully-connected ANN; but then setting the majority of connections between nodes to zero, with a small subset of genetically specified non-zero weights. We can maintain, throughout evolution, the typical proportion of weights that are non-zero by monitoring the add-link and delete-link components of the mutation operator. Thus if, for instance, at any mutation each non-zero weight was mutated to zero with probability 9%, and each zero weight mutated to a non-zero value with probability 1%, we can expect the proportion of non-zero weights to stay around 10%. In this manner, each *Org* is a only a small part of the whole possible ANN, and may very well be functionless on its own through having no connected path from inputs to outputs.

When a *Bucket* is assembled, then this is treated as the full ANN with any specific weight on a connection calculated as the sum (an alternative method would be to use the mean) of all values for that connection as specified on all the constituent *Orgs*; a variant method with subtle differences would be to exclude any zero values in the calculation of such a mean.

Designing an Autoencoder ANN with a Binomic GA

As a working demonstration we chose to use the Binomic GA to evolve ANNs in the form of an autoencoder, as described below. This allows us to compare performance with other versions of evolutionary algorithms that we had developed for similar autoencoders in a separate study.

Such autoencoders (Hinton and Salakhutdinov, 2006) are ANNs with a feedforward succession of layers, potentially fully connected between each successive layer. When appropriate weights are found, it should reduce high-dimensional input data through a lower-dimensional Bottleneck layer and then recover the input pattern and replicate it at the final output layer. Between Input and Bottleneck there is a Hidden Layer, which should encode the input pattern into the Bottleneck; thereafter a further Hidden Layer should decode to the Output.

We used autoencoders of the form N-h-M-h-N (see Figure 6), where N is number of Inputs/Outputs, M is the size of the Bottleneck layer, and h is the size of each Hidden layer. In our

implementation, all inputs were either 1 or -1. The Hidden Layer transfer functions were hyperbolic tangents, whereas the Bottleneck transfer function was linear. The output layer transfer function was a discrete step function that mapped positive/negative values into +1/-1 respectively. For simplicity, no biases were used in any of the networks.

We report here initial results on evolving with the Binomic GA appropriate weights for such autoencoders of sizes 3-12-2-12-3 and 4-24-4-24-3. Evaluations of such networks tested every possible binary input pattern and assessed how many output patterns matched. We compared performance of the Binomic GA (BGA) with two versions of a straightforward Microbial GA (recombination or 'infection' rate 0.5) where each individual in the population was a complete autoencoder with the appropriate architecture and genotypes specifying all the weights. The Microbial GA versions differed in mutation method: either a single weight was mutated, or all weights were mutated together.

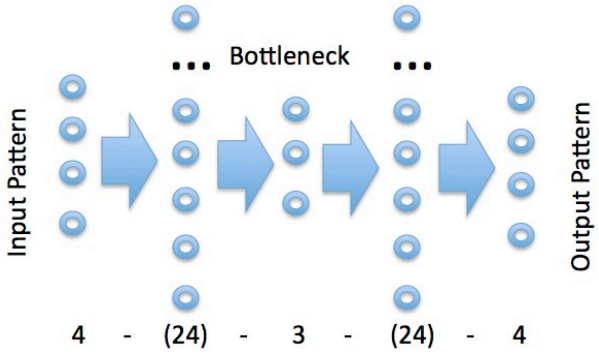

Figure 6. The task for this 4-24-3-24-3 autoencoder ANN is for the binary Input Pattern (here 4 bits) to be replicated at the final Output Layer, despite having passed through a narrower Bottleneck (here 3 nodes) in the middle.

Parameters used

We report on initial BGA experiments using a population or *Sea* of 50 *Orgs*, where each *Org* was a subset of the full autoencoder with (initially) 50% of the weights set to zero, the rest set to small random numbers with mean zero and standard deviation 0.1. Each *Bucket* took 25 *Orgs* at random from the *Sea*, and superimposed these on each other to form an autoencoder with weights on each connection equal to the sum of the respective weights on each *Org*. The fitness score of this *Bucket* was allocated equally to all of its component *Orgs*, their fitnesses updated with a smoothing factor R=0.1. No geographical demes were used.

Each tournament took two *Orgs* at random from the *Sea*, and determined Winner and Loser depending on their current fitnesses. The Loser was modified with a probability 0.5 of Vertical Gene Transmission (becoming a copy of the Winner), otherwise Horizontal Gene Transmission occurred (with 50% of the Winner's genes, or genetically specified weights, overwriting the corresponding Loser's genes). In order to maintain the proportions of zero/non-zero weights at around the initial 50/50 ratio, each non-zero weight in the Loser was deleted (set to zero) with probability (Number of non-zero weights)/(Number of weights) and conversely each zero weight was made non-zero, set to an initial small random value, with probability (Number of zero weights)/(Number of weights). Then a single non-zero weight of the Loser was mutated by adding a mutation, mean value 0.0, standard deviation 0.5 (the same mutation method as used with the single-weight-mutation Microbial GA).

Each time a tournament was completed, and the Loser thus modified, one *Bucket* containing the Loser was evaluated and all the *Orgs* within that *Bucket* had their fitnesses adjusted. This completes the Binomic GA cycle.

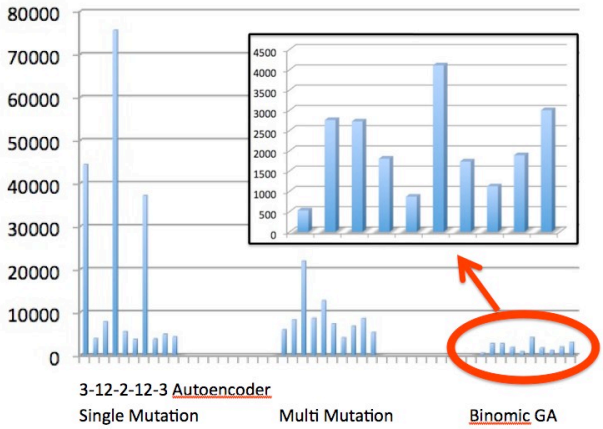

Figure 7. Number of evaluations needed to achieve a perfect score using 3 different GA methods (10 runs each) on the 3-12-2-12-3 autoencoder. The Microbial GA, with single weight mutation, took mean 19,092, std. dev. 24,932, maximum 75,623 evaluations; with multiple mutations 8,921, 5,118, 21,868 respectively. The Binomic GA took mean 2,052, std. dev. 1,098, maximum 4,105 evaluations, and is shown rescaled in the insert.

Experimental Results

For making comparisons, we take the significant factor to be the number of autoencoders that need evaluating before a perfect score is achieved. Runs were terminated if no success was achieved by a cutoff point. Each experiment was repeated 10 times; as is common with GAs, there was variance between runs; but there was a clear and striking pattern. The Binomic GA clearly outperformed its competitors.

We show in Figures 7 and 8 results for the 3-12-2-12-3 autoencoder, and the more difficult 4-24-3-24-4 autoencoder. In both cases the Binomic GA reliably generated perfect results, overall significantly faster than the competing methods, and with less variance. These are initial tests to demonstrate in principle that this method works, and it is gratifying to see the striking performance.

Discussion

GAs have been based on a traditional view of Darwinian evolution with individuals being evaluated for their fitness, and vertical gene transmission down the generations. Metagenomic studies have recently started to transform our

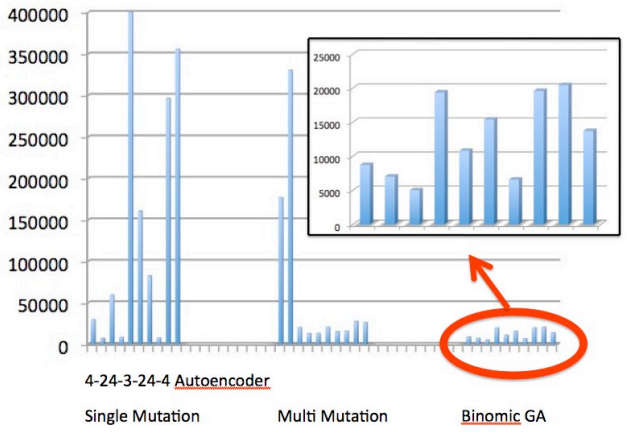

Figure 8. Results similarly shown for the 4-24-3-24-4 autoencoder. The Microbial GA, single weight mutation, took mean 140,608, std. dev. 154,047, maximum (cutoff without success) 400,000 evaluations; with multiple mutations 65,681, 105,408, 329,895 respectively. The Binomic GA took mean 12,681 evaluations, std. dev. 5,856, maximum 20,454.

view of evolution in the world of bacteria, which were amongst the earliest living entities and continue to play an enormous, often under-appreciated, role. We have highlighted the symbiotic nature of evaluations in communities of such real organisms, as emulated in part in the artificial world with LCS and SANE. We have shown how the horizontal gene transmission of bacteria is emulated in the Microbial GA. But as yet nobody appears to have combined these two aspects into applications in AL or EC.

This is primarily a position paper drawing attention to this lack of AL/EC work inspired by Metagenomics, despite significant traffic in the other direction. We propose a new sub-field of *Binomics* bringing these two ideas together as potentially fruitful in synthetic applications. The *Binomic GA* has been demonstrated to work well in preliminary tests, and this new approach opens up a whole range of new questions.

We need to investigate what parameter settings work well for what kind of problem. Does the autoencoder problem have some special property that is relevant? We note a potential relationship with neutral networks in the fitness landscape. The effects of varying *Bucket* size and the impact of drawing the *Buckets* locally within the *Sea* need to be studied. Taking account of this Metagenomic inspiration, we may expect that an appropriate application could be Evolutionary Computation that needs to be carried out online, with the evolving population actually carrying out its function in real time whilst adapting to environmental changes. One such example could be anti-virus (the computer variety of virus) software agents where a diverse population protects a system in real time, whilst reacting and adapting to new environmental threats.

Our preliminary work with the BGA leads us to believe that there is enormous scope for further developments. We hope this paper will stimulate interest in what has been until now a surprising gap in Artificial Life studies.

References

Bull, L., editor (2004). *Applications of Learning Classifier Systems*. Springer.

Committee on Metagenomics (2007). *The New Science of Metagenomics: Revealing the Secrets of Our Microbial Planet*. National Research Council, National Academies Press, Washington, DC. Downloadable from *www.nap.edu/catalog/11902.html*

Eisen, J. A. (2007). Environmental Shotgun Sequencing: Its Potential and Challenges for Studying the Hidden World of Microbes. *PLoS Biol.* 5(3): e82. Doi: 10.1371/journal.pbio.0050082.

Farmer, J. D. (1990). A Rosetta stone for connectionism. In Forrest, S., editor, *Emergent Computation: Proc. 9th Int. Conf. of the Center for Nonlinear Studies on Self-organizing, Collective, and Cooperative phenomena in Natural and Artificial Computing Networks,* pages 153-187. Amsterdam: North-Holland.

Forrest, S. and Miller, J. (1991). Emergent Behavior in Classifier Systems. *Physica D* 42: 213-217.

Goldberg, D. E., and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41-49. Hillsdale, NJ: Lawrence Erlbaum Associates.

Handelsman, J. (2004). Metagenomics: Application of Genomics to Uncultured Microorganisms. *Microbiology and Molecular Biology Reviews*, 68(4): 669-685.

Harvey, I. (2001). Artificial Evolution: a Continuing SAGA. In Gomi, T., editor, *Evolutionary Robotics: From Intelligent Robots to Artificial Life.* Springer-Verlag LNCS 2217.

Harvey, I. (2010 in Press). The Microbial Genetic Algorithm. In G. Kampis et al., editors, *Proc. of Tenth Eur. Conf. on Artificial Life.* Springer.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786): 504-577.

Holland, J. H. (1976). Adaptation. In Rosen, R. and Snell, S. N., editors, *Progress in Theoretical Biology*, 4. Plenum.

Holland, J. H. and Reitman, J. H. (1978). Cognitive Systems Based in Adaptive Algorithms. In Waterman, D. and Hayes-Roth, F., editors, *Pattern-directed Inference Systems*. Academic Press.

Horn, J., Goldberg, D. E. and Deb, K. (1994). Implicit Niching in a learning classifier system: Nature's way. *Evolutionary Computation*, 2(1): 37-66.

Moriarty, D. E. and Miikkulainen, R. (1996). Efficient Reinforcement Learning through Symbiotic Evolution. *Machine Learning*, 22: 11-33.

Moriarty, D. E. and Miikkulainen, R. (1999). Learning Sequential Decision Tasks. In Honavar, V., Patel, M. and Balakrishnan, K., editors, *Advances in the Evolutionary Synthesis of Neural Systems.* MIT Press, Cambridge, MA.

Penn, A. and Harvey, I., (2004). The Role of Non-Genetic Change in the Heritability, Variation and Response to Selection of Artificially Selected Ecosystems. In Pollack, J., Bedau, M., Husbands, P., Ikegami, T., and Watson, R.A., editors, *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems, ALIFE'9*, pages 352-357. MIT Press, Cambridge MA.

Qin, J., Li, R. et al. (2010). A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464: 59-67.

Smith, R. E. and Brown Cribbs, H. (1994). Is a Learning Classifier System a Type of Neural Network? *Evolutionary Computation*, 2(1): 19-36.

Spector, L. and Klein, J., (2005). Trivial Geography in Genetic Programming. In Yu, T., Riolo, R.L., Worzel, B., editors, *Genetic Programming Theory and Practice III*, pp. 109-124. Boston, MA: Kluwer Academic Publishers.

Venter, J. C., Remington, K. (and 21 further co-authors) (2004). Environmental genome shotgun sequencing of the Sargasso Sea. *Science* 304(5667): 66-74.