An Agent Too Far:
The Genetic Distance Evaluation of a Simulated World

E J P Earon and G M T D'Eleuterio

Institute for Aerospace Studies, University of Toronto
Toronto, Ontario, Canada, M3H 5T6
eea@sr.utias.utoronto.ca, gabriele.deleuterio@utoronto.ca

Abstract. One of the fundamental features, and one of the lesser understood phenomena, in biology is that of speciation. In order to better understand the development and creation of species, and their role in evolution, a method for tracking speciation in simulation is presented. While there is much dispute in the field of biology as to the precise definition of the term species, there is little debate that the natural world is full of distinct subpopulations. A genetic framework which permits a precise definition of a species is presented as it is implemented in the Cyberis simulator. The framework uses the notion of the Levenshtein distance as applied to the genomes of the agents to dynamically specify speciation during simulation. This allows for the study of artificial evolution on a more accurate scale by investigating the performance and longevity of the species, a more robust indicator of the genetics than a single agent. The genetics also allow much freedom in the specification of the agent and allows similar flexibility in experiments in neutrality and genetic difference calculations.

1 Introduction

Evolutionary computing has gained widespread use throughout a broad spectrum of research fields from developing amplifier circuits [1] to robotics research [2, 3]. Though there have been notable successes in these fields, the underlying processes at work in evolutionary systems are still being actively researched and a variety of simulations have been developed. Some simulations attempt to answer specific questions concerning populations or evolutionary dynamics ([4–6]), while others are more directed towards the simulation of artificial life ([7, 8]).

Biological systems provide an excellent example of the power of evolution as a developmental engine. The array of solutions to the problem of survival in a complex, hazardous world tendered by natural systems is staggering. If it is possible to understand the fundamental mechanisms at work, the potential benefits are likewise impressive. In biology, the concept of speciation is of utmost importance. Unfortunately, there is still little consensus on the precise definition of a species. Hey [9] argues that this is due in large part to an almost innate idea of species classification amongst humans which hinder the acceptance of

more precise definitions. Other research is ongoing which seeks to understand the fact that classical phylogenetic trees do not seem to describe species ancestry adequately in the natural world. A web or network may be a more appropriate metaphor for phylogenies than a tree [10] and ongoing research indicates that lineages may converge in previously unexpected ways [11].

While the argument could be made that the added complexity of tracking the descendancy of a subpopulation is unnecessary, there are several reasons why it is a very importance feature. First, there is little doubt that biological systems are organised into distinct groups. The Cyberis simulator attempts to understand how this process occurs in an artificial system. The benefits of robustness of the genotype over individuals are also particularly useful. It is not enough merely to understand which genome provides the most "successful" agent (particularly as defining "successful" is, in general, a rather difficult task).

2 The Simulated World

The Cyberis simulator is a program which allows a number of autonomous agents to live and react to their environment. On the initialisation of a simulation the size and initial configuration of the environment, the initial configuration of the population and the parameters for calculating genetic difference during the simulation are configured.

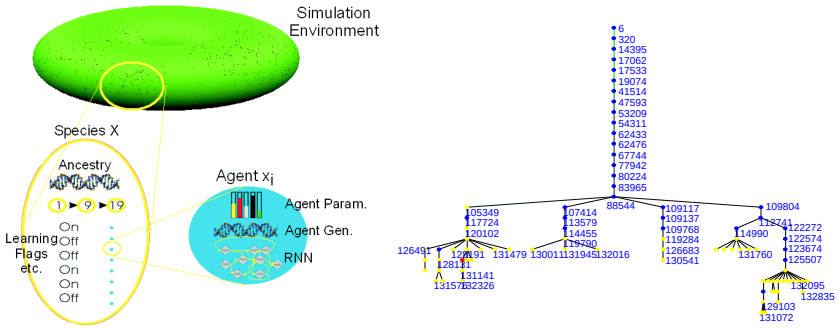

Fig. 1. Left The agents exist on the surface of a toroidal world. The population is comprised of a number of species each containing a subpopulation of agents. **Right** Phylogenetic tree generated by the Cyberis simulator showing all species that survived to $t = 5425000$ and their ancestors (phylogenies which did not survive to $t = 5425000$ are note shown).

The agents exist on the surface of a toroid as shown in Figure 1. While the simulator does have the ability to restrict subpopulations to isolated regions and thus investigate allopatric speciation, thus far all simulations for this work have been sympatric.

As shown in Figure 1, the population is broken into a number of subpopulations, or species. However, it should be noted that these distinctions are purely clerical. The agents are free to act and interact throughout the world with no limits. The only exception to this occurs when restrictions are placed on the agents' reproductive radius, which determined based on the genetic disparity between potential mates. The agents are able to sense and interact with the environment through a number of sensors and actuators. The sensory capability for the agents include such features as rudimentary vision and internal states sensors while the action set includes movement, eating, attacking, and reproducing.

The behaviour of the agents is controlled by a fully recurrent neural network. The cells of the network, as well as performing the computation for the networks also process the genome of the agent. The architecture of the network starts out with a single cell on the creation of the agent. As this cell processes the genome of the agent, it divides, modifies its connections and the corresponding connection weights and produces the artificial proteins, can be thought of as acting as "variables" for the cells and the agent as a whole. Each cell in the network, produced through the reproduction of the original cell, has its activation function specified by the genome.

3 Genomic Agent Description

In the Cyberis simulator, the genetics are best described as a variable length program that governs the phenotype of the agent. The programming language contains operations for jumping to different places in the program, conditional clauses and a fixed number of both internal and external variables. External variables are global in scope and are accessible by every program executing entity in the agent. The program executing objects in the simulator are the cells comprising the recurrent neural network of the agents. The scope of internal variables are local to the executing cells. Another way to view the genetics is as a Turing tape passing through a Turing machine, the cell.

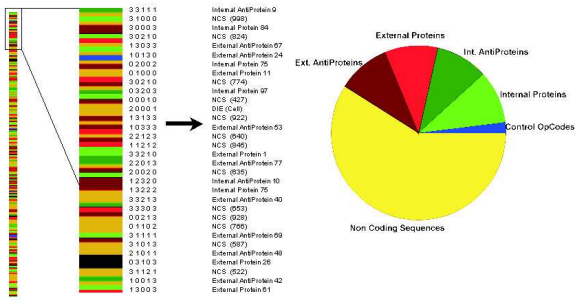

Fig. 2. A sample genome represented graphically and in its raw form.

The genetic program (or genome) γ is a variable length string of n characters. The alphabet for the genome has a length of 4. The characters are arranged in groups of five, called **codons**. This is similar to the work of Gilber and Eppstein who used a biological inspiration for their genetic encoding [12] though the format used in the Cyberis simulator is slightly different.

As in the biological systems, the genotype alphabet is comprised of four letters, but instead of groupings of three to create codons, they are arranged in strings of 5 letters. These codons then specify individual proteins. This process slightly simplifies the chemical processes which encode the proteins, while still permitting a great deal of flexibility in terms of genetic code. There are then $4^5 = 1024$ different codons, each of which specifies a specific instruction which is then read sequentially by the agent's cells.

4 Genetic "String" Theory

The fundamental information storage engine in this research is the genome. In evolutionary computing, genomes are represented all across a huge spectrum of research in a number of different ways. In the early days of evolutionary computing Holland used a simple fixed length binary string as an encoding of the finished product [13]. John Koza in his "genetic programming" research [14, 1] uses a variable length Lisp tree as the genome. Lenski et al and Wilke et al used program segments as a genome in the AVIDA simulation [4, 15]. In AVIDA, the genome is also the phenotype[1]. In the Cyberis simulator, the genome is similar to that used by the AVIDA simulation in that it is a segment of executable code. However the AVIDA genomes are the programs themselves. In this research the genome is an assembly code specifying the phenotype which then interacts with the world[2].

One of the most important features of this research is that this work defines the concept of a "species" and then organises the simulated agents into the relevant population subgroups, or species. In order to define the concept of a species based on genetics, the first requirement is the definition of a distance metric with which to compare the genomes of individuals. Three different methods were investigated for this purpose. The first, and a very popular method for calculating genetic difference in evolutionary computing, is the Hamming Distance [16]. While the Hamming distance was originally only defined for strings of equal length, this issue can be resolved. A more problematic feature is that it does not provide information on the number of mutations required to convert genome γ_A to γ_B or vice versa. In fact, without modification, a single insertion or deletion in one string relative to the other early in the string will greatly over-exaggerate the difference between the two. The Levenshtein (or Edit) Distance[17], provides just such detail. Unfortunately, the Levenshtein distance proved to be too computationally intensive (particularly for genetic strings with lengths of over

[1] In other words, the transformation from genotype to phenotype is identity

[2] The transformation in this case from genome to phenotype is a non-unique many-to-one mapping

one thousand codons as are commonly found in the Cyberis simulations) to be conveniently used for simulation. To that end a third method, an Approximate Levenshtein Distance (ALD) calculation was developed which closely tracks the true Levenshtein distance closely (relative to the Hamming distance). While the ALD requires more computational effort to calculate than the Hamming distance, it is much faster that the Levenshtein distance. Figure 3 shows the performance of the different metrics for a continuously mutating genome.

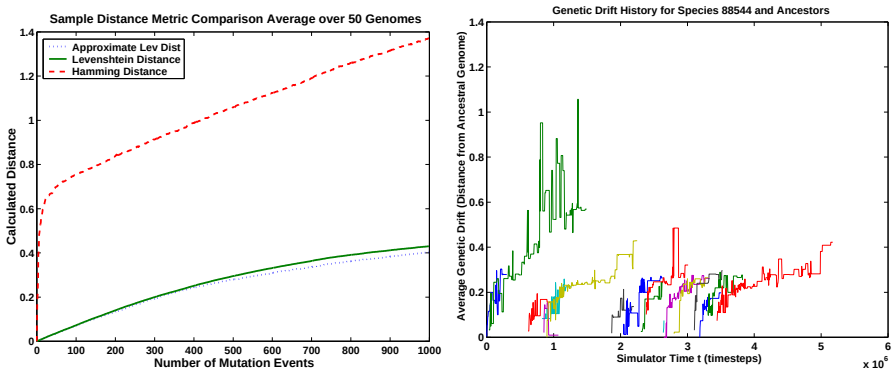

Fig. 3. Left Distance calculations from Levenshtein distance, Hamming distance, and an Approximate Levenshtein distance. Plot was generated by comparing a constantly mutating genome with its original, non-mutated form. While the Hamming distance is the fastest to compute, the Approximate Levenshtein distance tracks the true Levenshtein distance much more accurately. **Right** Genetic Drift history for Species 88544 and its ancestors. Each species' genetic drift starts at zero then, as the population collectively mutates away from the ancestral phenotype, the drift increases until the extinction of the species.

All three of these algorithms calculate the genetic distance between two genomes:

$$d = L(\gamma_1, \gamma_2) \tag{1}$$

and are true distance metrics in that they obey the following properties

$$L(\gamma_1, \gamma_2) = L(\gamma_2, \gamma_1) \geq 0 \tag{2}$$

$$L(\gamma_1, \gamma_1) = 0 \tag{3}$$

$$L(\gamma_1, \gamma_2) + L(\gamma_2, \gamma_3) \geq L(\gamma_1, \gamma_3) \tag{4}$$

for any γ_1, γ_2, $\gamma_3 \in \Gamma$, the set of all possible genomes. Where L is the distance metric, either the Levenshtein distance, the Hamming distance or, more commonly, the ALD.

Genetic Levenshtein Distance The Levenshtein distance between two genomes γ_1 and γ_2 (with lengths of n_1 and n_2, respectively) represents the length of the

minimum path to convert one genome into the other. This is computed by recursively generating a $(n_1 \times n_2)$ matrix, $\boldsymbol{D}$, and filling in the element values, $D^{i,j}$ through

$$D^{i,1} = i \tag{5}$$
$$D^{1,j} = j \tag{6}$$
$$D^{i,j} = min(D^{i-1,j} + 1, D^{i,j-1} + 1, D^{i-1,j-1} + f_d(\gamma_1^i, \gamma_2^j)) \tag{7}$$

for $1 \leq i \leq n_1$ and $1 \leq j \leq n_j$. The value in the bottom right element of the matrix (D^{n_1,n_2}) is the Levenshtein distance between the genomes. The value f_d represents the equality between genetic elemets:

$$f_d(\gamma_1^i, \gamma_2^i) = \begin{cases} 1 \text{ if } \gamma_1^i \neq \gamma_2^i \\ 0 \text{ otherwise} \end{cases} \tag{8}$$

The computational effort required to compute the Levenshtein distance is $O(n_1 \times n_2)$. This algorithmic complexity can cause problems, particularly when simulating large populations with long genomes. The distance is calculated:

1. When an agent is created
   - At most, once per species to identify the appropriate species for the agent
   - Once per agent in the appropriate species to re-calculate the Mean Agent
2. Every time an agent attempts to mate sexually
   - To confirm the eligibility of the mating
3. When an agent dies
   - Once per agent in the species to re-calculate the Mean Agent

As such, the added complexity of the Levenshtein distance quickly slows the simulator down to the point that it quickly becomes infeasible to use the Levenshtein distance

Approximate Levenshtein Distance The ALD uses a unique feature of the Cyberis genetics in order to calculate the distance value. This feature is the Index of the codon being measured. The index initially represents the position of the codon in the gene but as the gene is mutated this ordered indexing is lost. The index still provides a useful landmark when tracking the difference in genomes.

In the calculation of the distance between γ_1 and γ_2, the ALD moves down both, checking the codons as it traverse the genomes. If there is no discrepancy between indices, it checks the elements of the codons and calculates the number of mutations required to transform one to the other. If there is an index discrepancy at position i, it traverses one genome then the other until it can resolve this discrepancy at position j. This corresonds to the number of insertions or deletions between γ_1 and γ_2 from positions i to j. If there are several mutations in a given codon, it is considered a replacement as opposed to a codon with many mutations.

The intrinsic value of each of these operations, Insertion, Deletion, Replacement and Mutation can be adjusted, assigning a different distance "weight" to each. However, in this work, the weight for each operation were set equal to 1.

5 Protein Generation and Description

The question of how to use an agent's genotype to specify its phenotype now becomes the most pertinent. In this work, where the genome specifies only artificial proteins, the mapping is performed through the protein chemistry of the agents. As mentioned earlier, the possible sequences encoded by the codon permutations ($1024 = 4^5$) are broken down into a number of functional groups. Table 1 shows the breakdown of the proteins into their respective functional groups and the codon sequences that code for those proteins.

Protein ($\wp$) Number	**Function**
0 - 20	Genetic Language Operators
20 - 120	Cellular Internal Proteins
120 - 220	Cellular Internal Anti-Proteins
220 - 320	Cellular External Proteins
320 - 420	Cellular External Anti-Proteins
420 - 1024	Non-Coding Junk, or Neutral, Codons

Table 1. Functional group of proteins synthesized by their respective codons.

In the case of non-operational codons, or proteins, there exist both proteins and anti-proteins (as seen in Table 1. For each protein there exists a corresponding anti-protein, which act as its inverse. One unit of anti-protein eliminates one unit of protein. The levels most be greater than or equal to zero at all times.

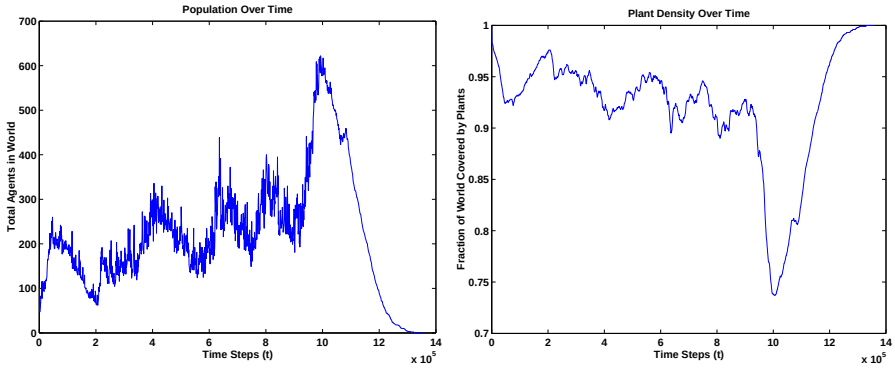

Fig. 4. Left Population graph for a simulation terminated by a population-wide extinction event and **Right** plant density plot for the same simulation, showing the cyclic nature of the environmental dynamics.

5.1 The Genetic Program

In order to allow for general genetic programs, and in particular for arbitrary neural network configurations (as specified by the genetics) it is critical for the genetics to be as general as possible in terms of the possible output sequences. To that end, the genetic programming language was designed.

Each cell in the agent, as it processes the genome, treats the first 20 codons in the codon sequence as control codes. The codes and their functions are detailed in Table 2. When a operator code is encountered by the cell in the processing of the genome, the cell also reads ahead by m codons where m is the number of operands for that operation. These values are then passed to the operator.

Op Code Number	**Operator**	**# of Operands**	**Operand Type**	**Function**
0	NOP	0	None	No Operation
1	IF	5	Scalars or Variables	Conditional IF
2	ELSE	0	None	Compliment to IF
3	JUMPUP	1	Scalar	Move Back n steps
4	JUMPDOWN	1	Scalar	Move Forward n steps
5	STOP	0	None	Stop Executing Genome
6	SPLIT	6	Scalars	Split cell
7	DIE	0	None	Kill cell
8	STOPCELL	0	None	Stop protein production
9	CONNECT	1	Scalar	Modify network architecture
10	END	0	None	End clause (IF, ELSE, CONNECT)
11	STARTGENE	0	None	Start Gene Flag
12	ENDGENE	0	None	Stop Gene Flag

Table 2. Table outlining functions of genetic operational codes in the genome.

The operator IF is unique in that it can use either scalars or variables for arguments as well as use different equivalence functions ($<$, $\leq$, $=$, $\geq$, or $>$). The particulars are specified by the five arguments. If the IF operation is to take scalar arguments, the value of the operands are used. If variables are used, the arguments specify particular proteins (either internal or external) and the levels of these proteins are used as arguments. SPLIT also uses a number of function arguments. For other operators, such as JUMPUP or CONNECT the argument simply specifies the number of steps or connections to modify.

5.2 Reserved Proteins

In order to specify the agent's physical phenotype, a number of external proteins also code for particular attributes of the agents. The proteins in Table 3 are external. This means that they are accessible by each cell in the agent and each cell contributes to the protein levels in the agent.

Ext. Protein	Attribute	Specifies:
0	MASS	The physical mass of the agent
1	SPEED	The distance the agent can travel in one time step
2	REDCOL	The intensity of the red pigment in the agent
3	GREENCOL	The intensity of the green pigment in the agent
4	BLUECOL	The intensity of the blue pigment in the agent
5	CARNIV	Agent's ability to metabolize plants or agents
6	STRENGTH	The amount of damage inflicted when attacking
7	RANGE_A	Agent acuity in sensing distances
8	MASS_A	Agent acuity in sensing mass
9	COLOUR_A	Agent acuity in sensing colour
10	MOVE_A	Agent acuity in sensing movement
11	REP_TYPE	Agent's reproductive type

Table 3. External proteins reserved for agent physical description.

5.3 The Protein Map

Every genome also contains a protein map. In fact, the protein map is typically the same for every genome in the simulation. However, it is only necessary that parents and offspring have the same mapping.

The map essentially provides a metaphor in the digital system for the protein chemistries in biological systems, though as mentioned in section 3, this is a simplification of the natural process. The sequence of codons is rearranged through a mapping, maintained by each genome such that any ordering between adjacent codon types can be removed. This mapping will distribute the potential, non-ordered, effects from a single point mutation on the genome.

Consider the effect of mutation on an artificial genome *without* a protein map. If the mutation occurs in a less significant letter, it is very likely that the function of the codon will not change. In other words, if the codon represented an internal protein before the mutation, it will most likely represent an internal protein after the mutation. Conversely, should the mutation affect a higher significance element, the codon will be very likely to change type. While this may seem insignificant to the operation of the simulation, it is important to remove, or at least reduce, this artificial ordering. In natural systems, a small change in the chemical composition of a protein can drastically alter its shape, and therefore its function. For this reason, a uniformly random mapping from codon value to protein type is used. In the previous tables (1,2,3) the value quoted for the proteins or opcodes are the values *after* the mapping.

The importance of heredity of the protein map is now evident. Should a parent pass a genome to its offspring, but with a different mapping, then the inherited genome will be, in general, completely different from its progenitor. It is for this reason, and to allow full reproductive freedom to the agents, that a single protein map is typically used for the entire simulation.

6 Some Sample Results

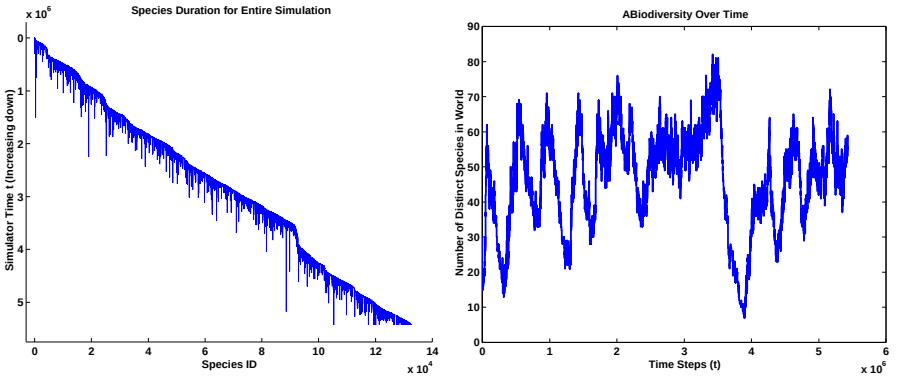

Fig. 5. Left New species creation plot showing duration of species existence. Exaggerated vertical drops in the plot correspond to extinction events. **Right** Simulation ABiodiversity (Artificial Biodiversity). The abiodiversity refers to the number of distinct species operating in theworld at that time. See also Figure 6 for further evidence of extinction events.

Figure 1**:right** shows a sample phylogeny from a simulation run. In reconstructing this information from the simulator data, there are a number of important points to bear in mind. Data files are periodically output from the simulator engine and these files form the basis for the simulation analysis. The output data essentially forms a "fossil record" of the species operating through time. While the simulator can be set to output these files at arbitrary intervals, for practical reasons the interval is usually set longer than several generations in the simulations. The consequence of this is that it is possible for a species to not appear in the fossil record if it is created and is extinguished within a data save interval. These species are said to be "missing links" as they provide a direct path between two known species but their existence can only be confirmed through the ancestry information stored by their descendant species.

One of the most interesting features of the simulations thus far has been extinctions. While a cyclic pattern of population and energy (food) availability is expected, and can readily be seen in Figure 4, there are more extreme patterns observable. Figures 5 and 4 show drastic drops in population. Usually the environment can recover from such population fluctuations, but Figure 4 shows an example where it does not and the population drop leads to a complete environment extinction. Initial investigations show that for small environments it is possible for genetic traits which drastically destabilise populations to propagate through the entire population. This is particularly true if the initial (though as of yet undeveloped trait) appears very early in the simulation in a species whose descendants proliferate thoroughly. These traits typically affect architecture of

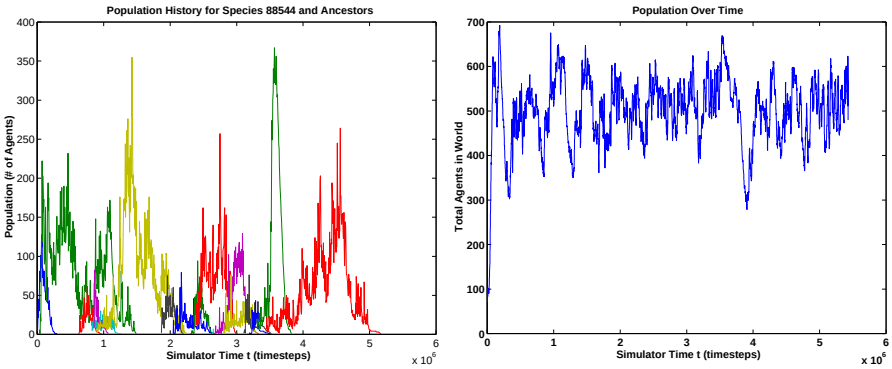

Fig. 6. Population over time details for simulation run. *Left* Population information for Species 88544 and its ancestors and *Right* Population information for all species.

the recursive neural network which controls the agents. This effect is made possible through the genome set used by the Cyberis simulator. The genetic language used permits genetic defects to be maintained though not expressed. The evolutionary process will eliminate this defect, though in simulations with small numbers of individuals, this elimination can have disastrous effects on the ecosystem as a whole.

7 Conclusion

The Cyberis simulator is designed to address the questions about evolution, and particularly speciation, that are currently being asked by a number of researchers in the fields of evolutionary computing, artificial life, and natural life systems. Fundamentally these types of questions address the rise of speciation and its role in the evolutionary process. How does speciation develop? Does reproductive isolation lead to post-zygotic speciation in sympatric species or the is the converse true? How does speciation aid in the propagation of highly fit traits in an environment which harbours much risk for individual agents? Does the propagation of a genome outweigh the importance of the individual in a species?

These types of questions are important and extremely relevant to our understanding of evolution, both as a natural process and in the artificial realm where it is being used with increasing success to solve a great many cutting edge research problems. The Cyberis simulator is a first step in answering these questions and the data being generated, even at this early stage, are providing a number of very interesting insights into our world at large.

References

1. Forrest Bennet III, Koza John R., Jessen Yu, and William Mydlowec. Automatic synthesis, placement and routing of an amplifier circuit by means of genetic pro-

gramming. In Miller, Thompson, Thomson, and Fogarty, editors, *Evolvable Systems: From Biology to Hardware, 3rd Int. Conf. on, Edinburgh, Scotland, UK, April 2000, Proceedings.*, volume 1801, pages 1–10. ICES 2000, Springer-Verlag, 2000.

2. E.J.P. Earon, T.D. Barfoot, and G.M.T. D'Eleuterio. From the sea to the sidewalk: The evolution of hexapod walking gaits by a genetic algorithm. In Miller, Thompson, Thomson, and Fogarty, editors, *Evolvable Systems: From Biology to Hardware, 3rd Int. Conf. on, Edinburgh, Scotland, UK, Proceedings.*, volume Vol 1801, pages 52–59. ICES 2000, Springer-Verlag, 2000.
3. Takashi Gomi and Ann Griffith. Evolutionary robotics - an overview. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 40–49, 1996.
4. Claus O. Wilke, Jia Lan Wang, Charles Ofria, Richard E. Lenski, and Christoph Adami. Evolution of digital organisms at high mutation rates leads to survival of the flattest. *Nature*, 412:331 – 333, 2001. Letters to Nature.
5. Elizaveta Pachepsky, Tim Taylor, and Stephen Jones. Mutualism promotes diversity and stability in a simple artificial ecosystem. *Artificial Life*, 8:5–24, 2002.
6. Richard Walker. The evolutionary origin of complex features. *Artificial Life*, 5:271–289, 1999.
7. Larry Yeager. Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or polyworld: life in a new context. *Artificial Life III*, 17:263–298, 1994.
8. Maciej Komosisiki and Szymon Ulatowski. Framsticks software overview. *Kybernetes: The International Journal of Systems and Cybernetics*, 29, 2000.
9. Jody Hey. The mind of the species problem. *TRENDS in Ecology and Evolution*, 16(7):326–329, 2001.
10. J.O. Andersson and A.J. Roger. Evolutionary analysis of the small subunit of glutamate synthase: Gene order conservation, gene fusions and prokaryote-to-eukaryote lateral gene transfers. *Eukaryotic Cell*, 1:304–310, 2002.
11. Jeanine M. Donley, Chugey A. Sepulveda, Sven Gemballa Konstantinidis, and Robert E. Shadwick. Convergent evolution in mechanical design of lamnid sharks and tunas. *Nature*, 429:61 – 65, May 2004. Letters to Nature.
12. Joshua Gilber and Maggie Eppstein. A case for codons in evolutionary algorithms. In Erick Cantú-Paz Et al., editor, *Genetic and Evolutionary Computation*, pages 967 – 990. Genetic and Evolutionary Computation Conference Chicago USA, Springer-Verlag, Berlin, 2003.
13. John H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, Cambridge, Massachusetts, 1992.
14. Koza John R., Jessen Yu, M.A. Keane, and William Mydlowec. Evolution of a controller with a free variable using genetic programming. In Poli, Banzhaf, Langdon, Miller, Nordin, and Fogarty, editors, *Genetic Programming: European Conference, EuroGP 2000, Edinburgh, Scotland, UK, April 2000, Proceedings.*, volume 1802, pages 91 – 105. EuroGP 2000, Springer-Verlag, 2000.
15. Richard E. Lenski, Charles Ofria, Robert T. Pennock, and Christoph Adami. The evolutionary origin of complex features. *Nature*, 423:139 – 144, 2003.
16. Richard Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 9:147 – 160, April 1950.
17. Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics - Doklady*, 6:707–710, 1966.