Achieving High-Level Functionality through Complexification

Kenneth O. Stanley and Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 USA
{kstanley, risto}@cs.utexas.edu

Abstract

An appropriate but challenging goal for evolutionary computation (EC) is to evolve systems of biological complexity. However, specifying complex structures requires many genes, and searching for a solution in such a high-dimensional space can be intractable. In this paper, we propose a method for finding high-dimensional solutions incrementally, by starting with an initial population of very small genomes and gradually *complexifying* those genomes by adding new genes over generations. That way, search begins in an easily-optimized low-dimensional space and increments into increasingly high-dimensional spaces. We describe an existing method for implementing complexification, and further propose that combining complexification with an indirect genetic encoding, in which genes are reused in the specification of the phenotype, can lead to the discovery of highly complex solutions.

Introduction

A major challenge in artificial intelligence is the automatic generation of highly complex structures such as large-scale neural networks, robot designs, and controllers. Biological evolution has achieved high-level complexity on a massive scale, and evolutionary computation (EC) aims at replicating this success artificially. However, encoding such domains genetically requires on the order of thousands or even millions of structural units for a single phenotype (Deloukas *et al.* 1998; Zigmond 1999, p. 9). Searching for a solution in such a high-dimensional space can take prohibitively long regardless of the encoding. A method is needed to discover high levels of complexity that avoids searching in the intractably large space as much as possible.

In this paper, we argue that the most promising way to reach biological levels of complexity is to start evolution with small, simple genomes, and *complexify* them over generations by adding new genes. In this way, evolution begins in a low-dimensional space that can easily be optimized, and gradually extends towards higher complexity. In fact, natural evolution has itself utilized this strategy, occasionally adding new genes that increased phenotypic complexity (Martin 1999). In biology, this process is called *complexification*.

We begin with an analysis of the shortcomings of traditional fixed-length encodings in EC, followed by a review of biological support for complexification. We then describe the NeuroEvolution of Augmenting Topologies method for evolving increasingly complex neural networks through complexification (NEAT; Stanley and Miikkulainen 2002a,b,c,d), and show how it can be generalized to any genetic encoding. Finally, we argue that a most promising research direction is combining complexification with indirect encodings. Such encodings utilize a developmental phase where the same genes are reused multiple times, allowing compact specification of complex structures (Bentley & Kumar 1999; Bongard 2002; Hornby & Pollack 2001b; Stanley & Miikkulainen 2003). In complexifying evolution of indirect encodings, search can take place in the smallest possible spaces with the most efficient possible representations.

Limitations of Fixed-Length Genomes

Many common structures vary in size and in the number of parameters that define them. In particular, phenotypes that can contain a variable number of parts can be represented by a varying number of genes. For example, the number of parts in neural networks, cellular automata, and electronic circuits vary (Stanley & Miikkulainen 2002d; Mitchell, Crutchfield, & Das 1996; Miller, Job, & Vassilev 2000a). However, despite such variability, two neural networks with different numbers of connections and nodes can approximate the same function (Cybenko 1989). Thus, it is not clear what number of genes is appropriate for solving a particular problem. Researchers evolving fixed-length genotypes must use heuristic rules, such as "small neural networks generalizing better than large ones," to estimate *a priori* what the appropriate number of genes is for a given problem.

A major obstacle to using fixed-length encodings is that such heuristic rules do not exist for very complex problems. For example, how many nodes and connections are necessary for a neural network that controls a ping-pong playing robot? Or, how many bits are needed in the neighborhood function of a cellular automata that performs information compression? These questions cannot be answered based on empirical experience or analytic methods, since little is known about the solutions. One possible approach is to sim-

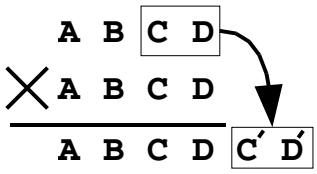

Figure 1: **Gene Duplication.** Two genomes of equal length are crossed over. The letters represent the trait expressed by each gene. The offspring has two additional redundant genes, resulting from the duplication of genes C and D from the first parent.

ply make the genome extremely large, so that it encodes an extremely large space and a solution is likely to lie somewhere within it. However, this strategy is likely to make the search intractable. Even if a ping-pong playing robot lies somewhere in the 10,000 dimensional space of a 10,000 gene genome, searching such a space may take prohibitively long.

Even more problematic are open-ended problems where phenotypes are meant to improve indefinitely and there is no known final solution. For example, in competitive games, it is difficult to estimate how complex the player should be because such an estimate implicitly assumes that no better player can exist. How could we ever know that? Moreover, many artificial life domains are aimed at evolving increasingly complex artificial creatures indefinitely long (Maley 1999). Fixing the size of the genome in such domains also fixes the maximum complexity of evolved creatures, defeating the purpose of the experiment.

Natural evolution avoids these problems by evolving variable-length genomes: Complexity can be increased by adding new genes to the genome. The next section discusses how this mechanism works in biology.

Complexification in Biology

Mutation in nature does more than optimize the genome. New genes are occasionally added, allowing evolution to perform a *complexifying* function over and above optimization. Complexification allows evolution to begin with simple solutions and elaborate on them incrementally, as opposed to evolving elaborate solutions from the start. Furthermore, elaboration is protected in nature in that interspecies mating is prohibited. Such *speciation* creates important dynamics differing from standard genetic algorithms. In this section, we discuss how these important characteristics of natural evolution bear on the artificial evolution of complex phenotypes.

The primary means of complexification in nature is *gene duplication*. Gene duplication is a special kind of mutation in which one or more parental genes are copied into an offspring's genome more than once. The offspring then has redundant genes expressing the same proteins (figure 1). Gene duplication has been shown responsible e.g. for key innovations in overall body morphology over the course of natural evolution (Amores *et al.* 1998; Carroll 1995; Force *et al.* 1999; Martin 1999).

A major gene duplication event occurred around the time that vertebrates separated from invertebrates. The evidence for this duplication centers around *HOX genes*, which determine the fate of cells along the anterior-posterior axis of embryos. HOX genes are crucial in shaping the overall pattern of developmental in embryos. In fact, differences in HOX gene regulation explain a great deal of arthropod and tetrapod diversity (Carroll 1995). Amores et al. (1998) explain that since invertebrates have a single HOX cluster while vertebrates have four, cluster duplication must have significantly contributed to elaborations in vertebrate body-plans. The additional HOX genes took on new roles regulating how the vertebrate anterior-posterior axis develops, considerably increasing body-plan complexity. Although Martin (1999) argued that the additional clusters can be explained by many single gene duplications accumulating over generations, as opposed to massive whole-genome duplications, researchers agree that gene duplication has contributed to important body-plan elaborations.

A detailed account of how duplicate genes can take on novel roles was given by Force et al. (1999): Base pair mutations in the generations following duplication *partition* the initially redundant regulatory roles of genes into separate classes. Thus, the embryo develops in the same way, but the genes that determine the overall body-plan are confined to more specific roles, since there are more of them. The partitioning is complete when redundant clusters of genes are separated enough that they no longer produce identical proteins. After partitioning, mutations within the duplicated cluster of genes alter different steps in development than mutations within the original cluster. In other words, the opportunities for mutation increase through duplication because duplication creates more points at which mutations can occur. In this way, developmental processes elaborate.

Gene duplication allows nature to add new dimensions to the genetic space. Natural evolution can thus begin searching in a simple space even if more advanced phenotypes cannot be found in that space. Because major biological shifts in body-plan complexity have resulted from adding new genes, EC should be able to utilize this kind of mutation as well. However, adding new genes requires variable-length genomes, which can be difficult to implement, as the next section discusses.

Implementing Variable Length Genomes

When duplication is allowed, the number of genes is variable, which means information can be lost during crossover. Figure 2 shows that as new genes are added in different lineages through different duplications, the same gene may exist at different positions in different genomes. Conversely, different genes may exist at the same position. Thus, crossover may lose essential genes through misalignment. Moreover, it may be difficult for a variable-length genome GA to find innovative solutions: Optimizing many genes takes longer than optimizing only a few, meaning that more complex genotypes may be eliminated from the population before they have a sufficient opportunity to be optimized. Although many existing systems utilize variable length genomes (Bentley & Kumar 1999;

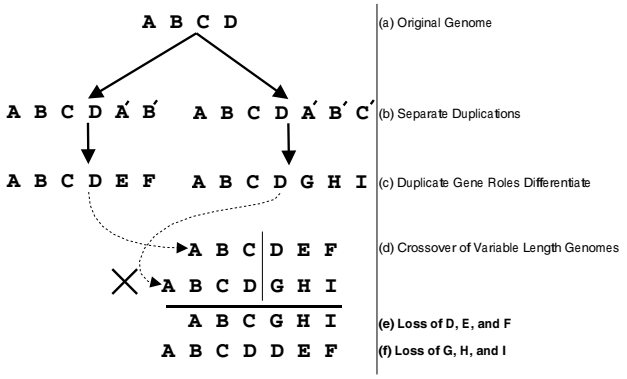

Figure 2: **The Problem of Aligning Variable Length Genomes.** In order to complexify solutions, it is necessary to have variable length genomes. The diagram shows how critical genes can be lost in the crossover of such genomes. A sequence of events is depicted from top to bottom (a through f). (a) The original genome contains four genes, A, B, C, and D. (b) In separate instances of reproduction, the original genome undergoes two different duplications. In one case a cluster of two of its genes, A and B, is duplicated. In the other case, three genes, A, B, and C, are duplicated. The resulting genomes now have differing lengths. (c) Over generations, the roles of the duplicated genes differentiate from their originally redundant roles. This functional divergence is represented using different letters, E and F, and G, H, and I. These new genes may serve important new roles in their respective lineages. (d) The two genomes of differing length are crossed over. (e) In one possible offspring, maintaining the length of the smaller genome, genes D, E, and F are lost. Particularly troublesome is the loss of D, which was in the original genome. (f) Another potential crossover, preserving the length of the larger genome, loses G, H, and I, along with duplicating D. When information is lost during crossover, the phenotype may no longer develop properly. Moreover, there is no way to ensure that all the necessary genes are included in the offspring without a mechanism for checking which genes from one genome correspond to those from another. See figure 3 for a solution to this problem.

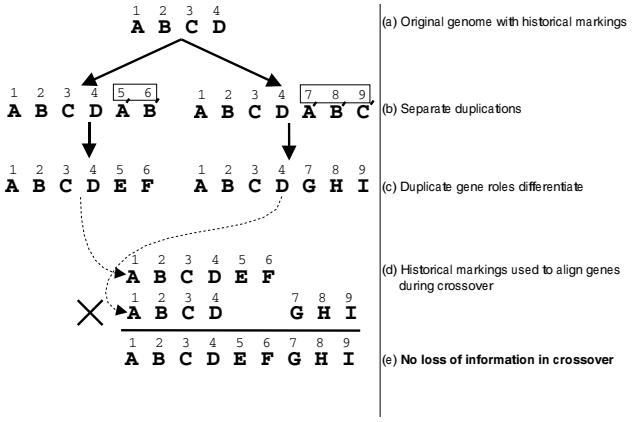

Figure 3: **Solving the Alignment Problem with Historical Markings.** Historical markings are numbers assigned to each gene that represent the order in which new genes appeared over evolution. (a) The original genome contains four genes, A, B, C, and D, assigned historical markings 1 through 4. (b) When new genes appear through duplication, they are assigned numbers in the order in which they appear. Assuming the duplication on the left happened before the one on the right, the new genes, A' and B', and A', B', and C', are assigned the numbers 5 through 9. (c) As the products of the duplicate genes differentiate, their historical markings continue to serve as a record of their origins. (d) During crossover, those genes that have matching historical markings are aligned, while those that are disjoint are purposefully not aligned. (e) The result is that any kind of crossover can preserve the information and relationships between all the genes in variable length genomes by utilizing the historical markings. Historical markings are an abstraction of synapsis, the process used in nature to match up alleles of the same trait during crossover (Sigal & Alberts 1972; Radding 1982).

Bongard & Paul 2000; Dellaert & Beer 1994; Gruau 1993; Hornby & Pollack 2001b; Komosinski & Rotaru-Varga 2001; Luke & Spector 1996; Sims 1994), none include mechanisms that would directly avoid these fundamental problems.

How has nature solved these problems? First, nature has a mechanism for aligning genes with their proper counterparts during crossover, so that data is not lost nor obscured. This alignment process has been most clearly observed in *E. coli* (Radding 1982; Sigal & Alberts 1972). A special protein called *RecA* takes a single strand of DNA and aligns it with another strand by attaching the *homologous genes*, i.e. genes that express the same traits. This process is called *synapsis*. In experiments in vitro, researchers have found that RecA protein does not complete the process of synapsis on fragments of DNA that are not homologous (Radding 1982).

Second, innovations in nature are protected through speciation. Organisms with significantly different genomes never mate because they are in different species. Thus, organisms with larger genomes compete for mates among their own species, instead of with the population at large. That way, organisms that may initially have lower fitness than the general population still have a chance to reproduce, giving novel concepts a chance to realize their potential without being prematurely eliminated.

It turns out that speciation and synapsis can both be utilized in EC based on features of evolution that are available only through computational means. Stanley and Miikkulainen (2002a,b,c,d) showed that the ancestral *history* of genes in an evolving population can be used to tell which genes should line up with which during crossover. If a counter assigns increasing integers to new genes every time they appear through a mutation, and if those integers are preserved every time genes are subsequently inherited, then the origin of every gene is known throughout evolution. The numbers assigned to each gene are called *historical markings*. Since two genes with the same origin must express the same trait, it is possible to know exactly which genes line up using the historical markings (figure 3).

Stanley and Miikkulainen (2002c,d) also showed that historical markings can be used to speciate the population, separating incompatible organisms into different niches. The extent to which two genomes have different genetic histories is a measure of their incompatibility. Therefore, matching historical markings allow a simple way to test whether

two genomes belong in the same species. This measure can be used to cluster genomes into compatibility groups, or species. *Explicit fitness sharing* (Goldberg & Richardson 1987) further ensures that highly fit species cannot crowd smaller species out of the population before they have a chance to reach their potential. That way, gene duplications do not need to immediately improve fitness in order to survive. On the other hand, since organisms without duplications are also protected in their own species, smaller genomes are preserved as long as they are competitive, avoiding bloating the genome.

The system utilizing these principals, called NeuroEvolution of Augmenting Topologies (NEAT), evolves neural networks of increasing complexity. By beginning search in the space of simple neural networks with only direct connections between inputs and outputs, evolution is able to search for the simplest solution possible. This process results in very fast search; For example, NEAT finds solutions to the non-Markovian double-pole balancing problem five times faster than other methods to date (Stanley & Miikkulainen 2002c,d). In addition, because NEAT does not begin searching directly in the space of the final solution, it is able to find significantly more complex controllers than fixed-topology evolution, as demonstrated in a robotic strategy-learning domain (Stanley & Miikkulainen 2002a,b). In fact, when evolution is forced to begin search directly in a the same space as the final solution found by NEAT, it cannot find a comparable solution. These empirical results confirm that complexification is a powerful approach that is possible to implement in practice.

So far, the NEAT approach has been applied only to a direct encoding of neural networks. In principle, historical markings can be applied to any genetic encoding, because history is a property of genes regardless of what type of phenotype they encode or how they encode it. Thus, the problems with variable length genomes can be overcome in many domains, and complexifying evolution is is a general approach to generating complex structures.

Discussion and Future Work

So far we have discussed how complexification helps find complex solutions, how nature utilizes complexification, and how it can be implemented in EC. In this section, we discuss the benefits of complexification in more detail and propose to enhance it further by combining it with indirect genetic encoding.

Benefits of Complexification

In fixed evolution, the complexity must be guessed just right. Too little structure will make it impossible to solve the problem and too much will make the search space too large to search efficiently. Moreover, *even if* complexity is guessed right, searching in the the high-dimensional space of the final solution may be intractable because search begins in a random part of the space.

Adding new genes to the genome solves both of these problems. Before the addition, the values of the existing genes have already been optimized over preceding generations. Thus, after a new gene is added, the genome is *already* in a promising part of the new, higher-dimensional space. Thus, the search in the higher-dimensional space is not starting blindly as it would if evolution began the search in that space.

As a result, complexification can find high-dimensional solutions that would otherwise be difficult to discover. Unlike standard evolution, complexifying evolution can also be applied to problems where the complexity of the final solution is unknown or unbounded.

Any arbitrarily-sized structure can potentially evolve through complexification. In addition to neural networks, cellular automata (Mitchell, Crutchfield, & Das 1996), electrical circuits (Miller, Job, & Vassilev 2000a; 2000b), genetic programs (Koza 1992), robot body morphologies (Lipson & Pollack 2000), Bayesian networks (Mengshoel 1999), finite automata (Brave 1996), and building and vehicle architectures (O'Reilly 2000) are all structures of varying complexity that can benefit from complexification.

So far complexifying evolution has only been used in systems where the phenotypic structure is directly encoded in the genome. A potentially powerful and unexplored possibility is to combine complexification with indirect genetic encoding. Because indirect encodings reuse genes during the development of the phenotype, their search space is smaller, and potentially highly efficient, as will be discussed next.

Combining Complexification and Indirect Encoding

Many kinds of genes can be tracked through historical markings, including indirect encodings (Stanley & Miikkulainen 2003). For example, in grammatical rewrite systems, a gene is a rule that specifies how a symbol in the developing phenotype should be expanded (Belew & Kammeyer 1993; Boers & Kuiper 1992; Hornby & Pollack 2001b; 2001a; Kitano 1990; Lindenmayer 1968). Another approach is to encode development as a tree of instruction genes that are executed in parallel by different parts of a developing phenotype (Gruau, Whitley, & Pyeatt 1996; Komosinski & Rotaru-Varga 2001; Luke & Spector 1996). Other indirect encodings attempt to simulate genetic regulatory networks (GRNs) in biology (Bongard & Pfeifer 2001; Astor & Adami 2000; Dellaert & Beer 1994; Eggenberger 1997; Jakobi 1995). In a GRN, genes produce signals that either activate or inhibit other genes in the genome. Some genes produce signals that cause e.g. cells to grow or axons to form. The interaction of all the genes forms a network that produces a phenotype. All these encodings support variable length genomes and historical markings, making complexification possible.

Direct encodings such as NEAT generally expand the genome by adding random genes (Angeline, Saunders, & Pollack 1993; Pujol & Poli 1998; Stanley & Miikkulainen 2002d). In contrast, nature uses duplication as the primary means of expanding the genome. The reason is that when genes are duplicated, the phenotype is not dramatically altered (Force *et al.* 1999). Such stability is important because generally major mutations in the genome could permanently

and immediately disabled the lineage. As Force et al. (1999) explained, subsequent mutations repartition the roles of both the original genes and the duplicated genes without significantly altering the overall developmental plan. Once duplicate genes have undergone sufficient mutation to be activated at different points during development than their original counterparts, subsequent mutations can begin to alter development at these new points. Thus, because of the duplicate genes, evolution has the flexibility to alter the developmental process at additional points.

Such a gradual process is difficult to achieve with direct encodings. When each gene maps to a single unit of phenotypic structure, duplicating genes is equivalent to duplicating part of the phenotype, which can significantly alter its functionality and structure. While in some cases such duplication is not destructive (e.g. with single neurons), duplicating an entire substructure of multiple components likely is.

In contrast, with indirect encoding, the duplicate genes can have overlapping, redundant roles. Thus they are guaranteed to play a role in development as soon as they are incorporated. Then, over the following generations, they can be partitioned gradually into different but related roles. Indirect encoding therefore allows making use of duplication as a nondestructive method of complexification.

The duplication process must carefully integrate the new genes into the already-existing developmental plan of the organism, and the subsequent mutations must not be too severe. If the genes become disconnected from the existing developmental plan, subsequent mutations will likely have little effect. Thus, in order to allow duplicate genes to gradually take on new roles, the conditions under which they activate should lie on a continuum. A slight mutation in one duplicate should cause it to be activated in some but not all cases where its counterpart was formerly always activated.

In addition to reducing the search space, in complexifying evolution with indirect encoding important substructures only need to be discovered once, even when they appear in the phenotype multiple times. Reuse of new structure is *inherent* in the underlying developmental program that has already evolved. For example, appendages can evolve digits all at once since the existing genetic specification ensures that each appendage follows the same developmental process. Thus, new genes that specify new structure at the end of such a process will be encountered each time the process occurs. On the other hand, complexification with direct encoding would require digits to be discovered separately on several occasions, since each set of digits must be specified by a separate set of genes. Thus, the combination of complexification and indirect encoding is potentially powerful because it allows global elaboration of all repeating structures simultaneously.

Indirect encodings that implement both synapsis and gradual divergence of duplicate genes will allow researchers also to understand the process of gene duplication better.[1] For example, how should clusters of genes be chosen for duplication? Everything from copying single genes to duplicating whole genomes is possible. While biologists continue to debate this issue (Amores *et al.* 1998; Martin 1999), EC can also begin to address it through experimentation. Calabretta et al. (2000) have already shown that distinct neural modules emerge when clusters of genes are duplicated in the evolution of neural networks. Thus duplicating entire groups of genes can be beneficial.

A complexifying system that starts with small, simple genomes will first evolve basic structures, such as bilateral symmetry, and then elaborate on them in future generations by adding new genes. The original simple developmental plan provides a framework on which to add specific elaborations and enhancements. The accumulation of such enhancements can ultimately create complex structures that would have been difficult to discover all at once. One of the most intriguing phenomena that might emerge from a successful implementation is repetition with variation. That is, instead of duplicating the same structure multiple times, a general *theme*, such as a limb, can be reused multiple times with differing manifestations. Such patterns do not follow traditional modular design in engineering, in which discrete identical parts are assembled into larger constructions. Instead, the beginnings and ends of individual parts are amorphous, and their internal structure is only vaguely constrained. The capacity to reuse parts with variation is potentially a very powerful way to create complexity, and a most intriguing direction of future research.

Conclusion

We argue that searching for highly complex structures directly in the space of the final solution is intractable. Complexification, i.e. adding new genes to the genome over the course of evolution, is a practical and powerful alternative. Natural evolution itself utilizes complexification. By starting search in a small space of few genes and incrementally adding more, evolution can build up complexity gradually. In order to implement complexification in EC, care must be taken to ensure that (1) crossover does not lose information, and (2) innovation is not prematurely lost. We proposed a method, NEAT, that accomplishes both these objectives and thereby allows complexification. Although the NEAT method can be generalized to any genetic encoding, a particularly promising research direction is to combine complexification with indirect encoding. It is now possible to implement and experiment with such a system, providing insight both into how complex systems can be artificially generated, and into nature's own mechanisms for continuing innovation.

Acknowledgments

This research was supported in part by the National Science Foundation under grant IIS-0083776 and by the Texas Higher Education Coordinating Board under grant ARP-003658-476-2001.

[1] *Gene deletion* is also possible, although it is potentially more deleterious than duplication. Duplication creates redundancy, which does not cause any loss of functionality. In contrast, deletion may cause important steps in development to be removed, short-circuiting the process.

References

Amores, A.; Force, A.; Yan, Y.-L.; Joly, L.; Amemiya, C.; Fritz, A.; Ho, R. K.; Langeland, J.; Prince, V.; Wang, Y.-L.; Westerfield, M.; Ekker, M.; and Postlethwait, J. H. 1998. Zebrafish HOX clusters and vertebrate genome evolution. *Science* 282:1711–1784.

Angeline, P. J.; Saunders, G. M.; and Pollack, J. B. 1993. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks* 5:54–65.

Astor, J., and Adami, C. 2000. A developmental model for the evolution of artificial neural networks. *Artificial Life* 6(3):189–218.

Belew, R. K., and Kammeyer, T. E. 1993. Evolving aesthetic sorting networks using developmental grammars. In Forrest, S., ed., *Proceedings of the Fifth International Conference on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann.

Bentley, P. J., and Kumar, S. 1999. The ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, 35–43. San Francisco, CA: Morgan Kaufmann.

Boers, E. J., and Kuiper, H. 1992. Biological metaphors and the design of modular artificial neural networks. Master's thesis, Departments of Computer Science and Experimental and Theoretical Psychology at Leiden University, The Netherlands.

Bongard, J. C., and Paul, C. 2000. Investigating morphological symmetry and locomotive efficiency using virtual embodied evolution. In *Proceedings of the Sixth Internation Conference on Simulation of Adaptive Behavior*, 420–429. MIT Press.

Bongard, J. C., and Pfeifer, R. 2001. Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In Spector, L.; Goodman, E. D.; Wu, A.; Langdon, W. B.; Voigt, H.-M.; Gen, M.; Sen, S.; Dorigo, M.; Pezeshk, S.; Garzon, M. H.; and Burke, E., eds., *Proceedings of the Genetic and Evolutionary Computation Conference*, 829–836. San Francisco, CA: Morgan Kaufmann.

Bongard, J. C. 2002. Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation*.

Brave, S. 1996. Evolving deterministic finite automata using cellular encoding. In Koza, J. R.; Goldberg, D. E.; Fogel, D. B.; and Riolo, R. L., eds., *Genetic Programming 1996: Proceedings of the First Annual Conference*, 39–44. Stanford University, CA, USA: MIT Press.

Calabretta, R.; Nolfi, S.; Parisi, D.; and Wagner, G. P. 2000. Duplication of modules facilitates the evolution of functional specialization. *Artificial Life* 6(1):69–84.

Carroll, S. B. 1995. Homeotic genes and the evolution of arthropods and chordates. *Nature* 376:479–485.

Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems* 2(4):303–314.

Dellaert, F., and Beer, R. 1994. Toward an evolvable model of development for autonomous agent synthesis. In Brooks, R. A., and Maes, P., eds., *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)*. Cambridge, MA: MIT Press.

Deloukas, P.; Schuler, G.; Gyapay, G.; Beasley, E.; Soderlund, C.; Rodriguez-Tome, P.; Hui, L.; Matise, T.; McKusick, K.; Beckmann, J.; Bentolila, S.; Bihoreau, M.; Birren, B.; Browne, J.; Butler, A.; Castle, A.; Chiannilkulchai, N.; Clee, C.; Day, P.; Dehejia, A.; Dibling, T.; Drouot, N.; Duprat, S.; Fizames, C.; and Bentley, D. 1998. A physical map of 30,000 human genes. *Science* 282(5389):744–746.

Eggenberger, P. 1997. Evolving morphologies of simulated 3D organisms based on differential gene expression. In Husbands, P., and Harvey, I., eds., *Proceedings of the Fourth European Conference on Artificial Life*, 205–213. Cambridge, MA: MIT Press.

Force, A.; Lynch, M.; Pickett, F. B.; Amores, A.; lin Yan, Y.; and Postlethwait, J. 1999. Preservation of duplicate genes by complementary, degenerative mutations. *Genetics* 151:1531–1545.

Goldberg, D. E., and Richardson, J. 1987. Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J. J., ed., *Proceedings of the Second International Conference on Genetic Algorithms*, 148–154. San Francisco, CA: Morgan Kaufmann.

Gruau, F.; Whitley, D.; and Pyeatt, L. 1996. A comparison between cellular encoding and direct encoding for genetic neural networks. In Koza, J. R.; Goldberg, D. E.; Fogel, D. B.; and Riolo, R. L., eds., *Genetic Programming 1996: Proceedings of the First Annual Conference*, 81–89. Cambridge, MA: MIT Press.

Gruau, F. 1993. Genetic synthesis of modular neural networks. In Forrest, S., ed., *Proceedings of the Fifth International Conference on Genetic Algorithms*, 318–325. San Francisco, CA: Morgan Kaufmann.

Hornby, G. S., and Pollack, J. B. 2001a. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2002 Congress on Evolutionary Computation*.

Hornby, G. S., and Pollack, J. B. 2001b. Body-brain coevolution using L-systems as a generative encoding. In Spector, L.; Goodman, E. D.; Wu, A.; Langdon, W. B.; Voigt, H.-M.; Gen, M.; Sen, S.; Dorigo, M.; Pezeshk, S.; Garzon, M. H.; and Burke, E., eds., *Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco, CA: Morgan Kaufmann.

Jakobi, N. 1995. Harnessing morphogenesis. In *Proceedings of Information Processing in Cells and Tissues*, 29–41.

Kitano, H. 1990. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems* 4:461–476.

Komosinski, M., and Rotaru-Varga, A. 2001. Comparison

of different genotype encodings for simulated 3D agents. *Artificial Life* 7(4):395–418.

Koza, J. R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press.

Lindenmayer, A. 1968. Mathematical models for cellular interaction in development parts I and II. *Journal of Theoretical Biology* 18:280–299 and 300–315.

Lipson, H., and Pollack, J. B. 2000. Automatic design and manufacture of robotic lifeforms. *Nature* 406:974–978.

Luke, S., and Spector, L. 1996. Evolving graphs and networks with edge encoding: Preliminary report. In Koza, J. R., ed., *Late-breaking Papers of Genetic Programming 1996*. Stanford Bookstore.

Maley, C. C. 1999. Four steps toward open-ended evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, 1336–1343. San Francisco, CA: Morgan Kaufmann.

Martin, A. P. 1999. Increasing genomic complexity by gene duplication and the origin of vertebrates. *The American Naturalist* 154(2):111–128.

Mengshoel, O. J. 1999. *Efficient Bayesian Network Inference: Genetic Algorithms, Stochastic Local Search, and Abstraction*. Ph.D. Dissertation, University of Illinois at Urbana-Champaign Computer Science Department, Urbana-Champaign, IL.

Miller, J. F.; Job, D.; and Vassilev, V. K. 2000a. Principles in the evolutionary design of digital circuits – Part I. *Journal of Genetic Programming and Evolvable Machines* 1(1):8–35.

Miller, J. F.; Job, D.; and Vassilev, V. K. 2000b. Principles in the evolutionary design of digital circuits – Part II. *Journal of Genetic Programming and Evolvable Machines* 3(2):259–288.

Mitchell, M.; Crutchfield, J. P.; and Das, R. 1996. Evolving cellular automata with genetic algorithms: A review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96)*. Russian Academy of Sciences.

O'Reilly, U.-M. 2000. Emergent design: Artificial life for architecture design. In *7th International Conference on Artificial Life (ALIFE-00)*. Cambridge, MA: MIT Press.

Pujol, J. C. F., and Poli, R. 1998. Evolving the topology and the weights of neural networks using a dual representation. *Applied Intelligence Journal* 8(1):73–84. Special Issue on Evolutionary Learning.

Radding, C. M. 1982. Homologous pairing and strand exchange in genetic recombination. *Annual Review of Genetics* 16:405–437.

Sigal, N., and Alberts, B. 1972. Genetic recombination: The nature of a crossed strand-exchange between two homologous DNA molecules. *Journal of Molecular Biology* 71(3):789–793.

Sims, K. 1994. Evolving 3D morphology and behavior by competition. In Brooks, R. A., and Maes, P., eds., *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)*. Cambridge, MA: MIT Press. 28–39.

Stanley, K. O., and Miikkulainen, R. 2002a. Competitive coevolution through evolutionary complexification. Technical Report AI2002-298, Department of Computer Sciences, The University of Texas at Austin.

Stanley, K. O., and Miikkulainen, R. 2002b. Continual coevolution through complexification. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. San Francisco, CA: Morgan Kaufmann.

Stanley, K. O., and Miikkulainen, R. 2002c. Efficient reinforcement learning through evolving neural network topologies. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. San Francisco, CA: Morgan Kaufmann.

Stanley, K. O., and Miikkulainen, R. 2002d. Evolving neural networks through augmenting topologies. *Evolutionary Computation* 10(2):99–127.

Stanley, K. O., and Miikkulainen, R. 2003. A taxonomy for artificial embryogeny. *Artificial Life*.

Zigmond, M. J.; Bloom, F. E.; Landis, S. C.; Roberts, J. L.; and Squire, L. R., eds. 1999. *Fundamental Neuroscience*. London: Academic Press.