9 Open-ended Neuroevolution

9.1 Openended Discovery of Complex Behavior

Neuroevolution has produced several convincing demonstrations where complex behavior is discovered in behavioral tasks, sometimes rivaling complexity seen in nature. However, there is one striking difference: neuroevolution is set up to solve a particular problem, whereas biological evolution has no goal. In nature, solutions are discovered continuously as challenges and opportunities come up. Such openendedness is still a challenge for artificial evolution, especially when the goal is to evolve general intelligent agents (Miikkulainen and Forrest 2021). This section reviews five elements of openendedness in biology that may, if we can implement them well, lead to openended neuroevolution: neutrality with weak selection, enhanced exploration through extinction events, highly evolvable representations, powerful genotype-to-phenotype mappings, and major transitions in complexity.

9.1.1 Neutral mutations with weak selection

Current evolutionary computation approaches, including those that evolve neural networks for behavior, aim to be strong and efficient. They utilize small populations that can be evaluated quickly; the crossover and mutation operations are often carefully crafted to make it likely that fitness is improved; fitness is measured precisely, and selection is strongly proportional to fitness. As a result, evolution converges the population quickly around the most promising solutions and finds good solutions there fast. This approach is effective e.g. in many engineering problems where the search space and fitness are well defined and the problem consists largely of optimizing the design.

However, this success often comes with the expense of reduced extrapolation and thus reduced creativity. It is also not very effective when the agents need to be general, i.e. cope with uncertain and changing environments and solve multiple tasks simultaneously. Other mechanisms are needed to counterbalance the effective search, such as diversity maintenance methods, novelty search, and quality diversity search (Section 5.3). They are intended to keep the population of solution diverse for a longer time, and spread it out further in the solution space. The idea is to not miss solutions that are complex or unexpected, i.e. hard to find through greedy search.

Interestingly, biological solutions are sometimes highly creative and unexpected, yet do not seem to result in any special mechanisms for diversity maintenance. If anything, biological solutions need to be viable always, which seems to counteract the need for diversity. How does biology do it?

Nature seems to employ an entirely different approach to creativity (Lynch 2007; Wagner 2005; Miikkulainen and Forrest 2021). The populations are very large, and selection is weak. Often there is also a lot of time for these processes to find solutions. Phenotypic traits are coded redundantly through several genes, much of the DNA exists in noncoding regions, and many of the mutations are neutral, i.e. do not affect fitness. As a result, diversity can exist in such populations: there is time to create it, and it stays even if it isn't immediately beneficial. The population as a whole can thus stay robust against changes, develop expertise for multiple tasks, and maintain evolvability through time.

There is a good reason for the strong and impatient approach that EC has taken until now. Evolutionary optimization is computationally intensive, and such techniques were necessary in order to take advantage of what was available. However, now that we have million times more compute than just a couple of decades ago (Routley 2017), it may be time to rethink the approach. This is precisely what happened with deep learning. Much of the technology, such as convolutional networks, LSTMs, autoencoders, existed since the 1990s, but they only started working well when taking advantage of the massive increases in scale (LeCun, Bengio, and Hinton 2015).

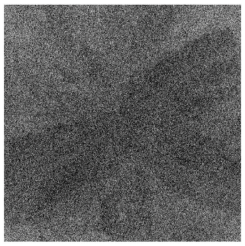
A similar opportunity may exist for evolution in general, and neuroevolution in particular. It may be possible to scale up to large populations, large redundant genomes, noncoding DNA, neutral mutations, and deep time. It may be possible to take advantage of massive amounts of behavioral data, and large-scale simulations, to evaluate the solutions. The evaluations may be multiobjective and high level, instead of carefully engineered to produce solutions of expected kind. Eventually it may even be possible to create foundation models for neuroevolution, i.e. large, diverse populations of neural networks that have many different abilities and are thus highly evolvable to solve new tasks.

One way to accelerate evolution in such populations is through extinction events, as will be discussed next.

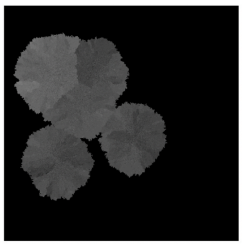
9.1.2 Extinction events

In biological evolution, large-scale extinction events have occurred several times, often seemingly changing the course of evolution (Raup 1986; Meredith et al. 2011). For instance, the Cretaceous-Peleogene extinction displaced dinosaurs with mammals, eventually leading to the evolution of humans. An interesting question is: Are such events simply historical accidents, or do they implement a principle that in some way enhances, or hinders, evolution in the long term? Even though such events obviously destroy a lot of solutions, can they possibly serve to reset evolution so that better evolvability is favored, which in the long term results in accelerated evolution and more complex solutions?

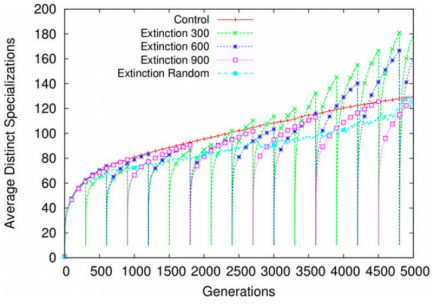
While it is difficult to evaluate this hypothesis in nature, it is possible to do so in computational experiments. It is possible to set up a large population with many different solutions, representing adaptations to different niches. If evolution runs in a stable manner for a long time, those niches are eventually filled with good solutions, and evolution stagnates. At such a point in time, extinction event eliminates most such solutions. Those that remain,



(*a*) Abstract: No extinction (15,000 gens)



(*b*) Abstract: Random extinctions (15,000 gens)



(*c*) Walker: Niches filled over time

Figure 9.1: **Effect of extinction events on evolvability.** While extinctions are catastrophic in the short term, they may empower evolution in the long term. (a) Without extinction events, the population in the abstract domain evolves to fill in the available niches (i.e. cells in the 401×401 grid). A variety of evolvability levels exists in the end, indicated by the grey-scale values (lighter is more evolvable). (b) With extinction events, higher evolvability is favored. Such events occurred at random intervals averaging 2,500 generations. In this snapshot, five individuals survived a recent event, and the population is currently expanding to fill in the available niches. On average these individuals are about 50% more evolvable than those in (a), indicated by the lighter color. (c) In the bipedal walker domain, extinction events rebound quickly, filling in more niches than before the event, and eventually more than evolution without extinction events. Thus, extinction events accelerate evolution and result in the discovery of more novel solutions. (Figures from Lehman and Miikkulainen 2015)

even just very few, are then free to evolve to fill the open niches. Such evolution can be described as radiation from the remaining niches, but note that there is also a meta-level selection at play: The solutions that are more evolvable, i.e. faster to adapt to the open niches, will spread faster and wider, making them more likely to survive the next extinction event. Thus, under repeated extinction events, evolution favors higher evolvability. Extinction events can thus have a positive long-term effect, accelerating evolution, and possibly resulting in more complex solutions as well.

To visualize the basic idea, consider a very simple computational setup (Lehman and Miikkulainen 2015). The niches are cells in a toroidal 401×401 grid world. Individuals consist of grid coordinates and a probability of changing those coordinates. Thus, adaptation means moving to a new cell, and high evolvability is represented by high probability of change. Initially there is only one individual at the center, and evolution creates more individuals by cloning and then mutating grid coordinates, and at the same time, mutating the probability. Over time, the population spreads to fill in all niches simply through drift (Figure 9.1*a*). However, with extinction events, only five individuals at random locations survive. If such events occur often, there is a strong selection towards individuals that mutate with a high probability. Thus, after prolonged evolution, the population evolved with extinction events is more evolvable than a population evolved without them (Figure 9.1*b*).

Do these results hold at the level of behavior as well? Consider again the bipedal walker domain described in Section 5.3. As before, the controllers are neural networks evolved

with NEAT, taking the location of the two feet (whether on the ground or not) as input, and torque to the six motors (one in each knee, two in each side of the hip) as output. A behavioral niche can be defined on the grid as in the abstract domain, i.e. the final location of the bipedal walker after 15 seconds of simulation. This location is also used to measure novelty, and evolution is set to maximize novelty. Evolvability can then be measured as the behavioral diversity of the offspring: The individual is mutated 200 times; the number of distinct final locations of the offspring represents its evolvability.

As can be seen in Figure 9.1*c*), evolution without extinction events expands to fill in the various niches monotonically. With extinctions, there is an immediate drop to five niches and a fast rebound to a higher level than before the event. Moreover, the rebounds become more effective over time, eventually filling more niches than evolution without extinctions. Thus, extinction events result in accelerated evolution and solutions with increased novelty.

These computational experiments suggest how extinction events can accelerate evolution in biology. Although major such events have taken place only a few times, they can be frequent at a smaller scale, resulting e.g. from fires, volcanic eruptions, climate events, predator migrations, and even human impact. The results also suggest that the same effect could be harnessed in engineering applications of computational evolution, leading to better results in the long term. Combining it with large populations and weak selection, as discussed in Section 9.1.1, is therefore a compelling direction for future work.

9.1.3 Evolvable representations

This chapter so far has outlined an approach to open-ended evolution that is still largely building on genotypic and phenotypic *diversity*, with a constant mapping between them. An alternative approach is to take advantage of *evolvability*, which can be defined as adapting the genotype-phenotype mapping over time such that the search operators are more likely to generate high-fitness solutions. High evolvability is often based on indirect encodings, which can provide a substrate for this adaptation.

The main challenge is that whereas high evolvability provides a future benefit for evolution, it needs to be developed implicitly based on only current and past information. In biology, evolvability may selected for in three ways (Kirschner and Gerhart 1998): more genetic variation can be stored in the population (because fewer mutations are harmful); it makes organisms more tolerant against stochastic development; and it makes it more likely for the populations to survive in changing environments.

Each of these can be evaluated in computational experiments. Opportunities for the first one were already discussed above in Section 9.1.1. Opportunities for the second one are illustrated in sections on development (Sections 4.2 and 14.4. In short, an individual is not complete at birth, but goes through a period of physical and mental development that results in a more complex and capable individual (Müller 2014). Often this period involves interactions with the environment, i.e. at least some of the complexity is not innate, but is extracted from the environment. These interactions can be synergetic and encoded into critical periods of development. For example, human infants need to receive language input when they are one-to-five years old, otherwise they do not develop full language abilities (Section 14.7.1). In this manner, instead of coding everything directly into genes, evolution also encodes a learning mechanism that results in a more evolvable encoding (Elman et al. 1996a; Valsalam, Bednar, and Miikkulainen 2005).

The third advantage opens up an opportunity that is particularly well aligned with open-ended evolution. Given a domain with known structure, such as evolution of symmetric bitstrings, evolution can be given an open-ended series of challenges in the form of different target bitstrings (Reisinger and Miikkulainen 2006). The population has to discover each target by continuing evolution of the current population (initially random). The target changes at given intervals, which has to be long enough for success to be possible. The evolvable representation consists of linkage parameters between bit locations, biasing the mutations that occur. Over time, evolution discovers linkages that favor symmetric strings, which makes discovery of targets gradually faster and more likely. In other words, the representations become more evolvable in this domain.

How can such representations be designed for more complex solutions such as neural networks and behavior? It turns out that the idea of linkages that adapt to the domain can be scaled up to neural networks, with an approach that is motivated by Genetic Regulatory Networks (GRNs; Wang 2013). As was discussed in Section 4.2.1, GRN is one way in which biology establishes an indirect encoding. Building on the operon implementation of GRNS in Section 4.2.1, GRNs can be modeled more generally with a set of rules (Reisinger and Miikkulainen 2007). As usual in rule-based system, each rule has an antecedent that is matched with the current state of the system, and a consequent that determines what output, or product, is generated. When used to construct neural networks, the products are either hidden or output nodes. When the antecedent is matched with currently existing products within a similarity tolerance, connections are created between nodes. The tolerance, amount of products, and the resulting connection weights are determined by regulatory factors in the antecedents. A simple example of this process is depicted in Figure 9.2.

The rules and the regulatory factors in them are modified through evolution in order to construct a neural network to solve the task. Note that this is a continuous, soft process, where a given product can gradually increase (through neutral mutations), until a tolerance is reached. It therefore has significant potential for evolvability: A general GRN structure is discovered where mutations often lead to viable offspring.

This process was demonstrated in Nothello, a board game similar to Othello but with a diamond-shaped board of 36 cells and an objective of fewest pieces on the board. It offers faster evolution with still much of the same complexity than full Othello. The networks were evolved the serve as heuristic board evaluators for minimax search; a single-ply lookahead was used to allow for longer evolutionary runs. In a coevolutionary setup, each candidate was evaluated with a random sampling of other individuals in the population. Note that coevolution provides an environment where the fitness function is constantly changing. As discussed above, such an environment should encourage evolvable representations to emerge. Evolvability is also directly useful because it results in discovering better game play over time.

Indeed the GRN-based implicit encoding approach results in discovering better networks over time compared to e.g. standard NEAT neurovolution, as seen in Figure 9.3*a*. This improvement is likely due to increased evolvability. Evolvability was measured as the average fitness of the local mutation landscape: Each representation was mutated to increasing extent and the performance of the offspring measured. The GRN-based implicit encoding results in much more robust mutations, i.e. improved evolvability (Figure 9.3*b*). It is also interesting to see that the network structures that results are different. Whereas

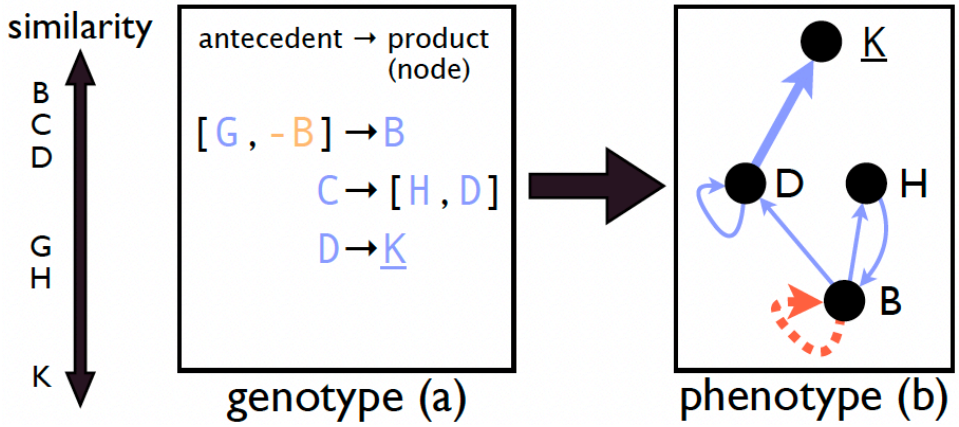
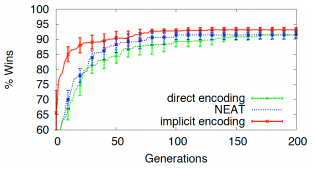
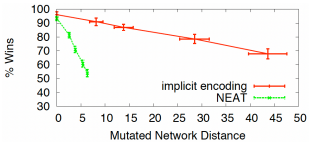


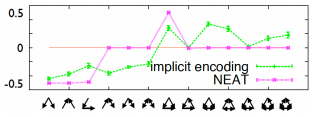
Figure 9.2: **Constructing neural networks with a GRN**. GRNs, a mechanism for decoding genetic representations in biology, it can also be used as an indirect encoding for neural networks. The GRN is encoded as a set of rules. The current state is represented by products (indicated by letters). The antecedents are matched with the current products, leading to generation of more products. The match is based on similarity between products, implemented through regulatory factors. In mapping the GRN to a network, products create nodes and antecedent matches connections between them. In this case, starting with products G and B as a starting point, matching the first rule creates a negative connection from B to itself. Because C is a similar product to B, H and D are created as hidden nodes and connected to B. Matching D in turn leads to a recurrent self-connection as well as creating and connecting to an output node K. In this manner, a recurrent structure is created; it can be further evolved by modifying the rule set and the regulatory factors. (Figure from Reisinger and Miikkulainen 2007)



(*a*) Champion performance in 1-ply search



(*b*) Performance vs. offspring distance



(*c*) Significance of network motifs

Figure 9.3: **Performance, evolvability, and structure resulting from GRN-based neuroevolution.** The GRN-based encoding has several useful properties, as illustrated in the Nothello game domain. (*a*) The GRN-based indirect encoding evolves better solutions faster. (*b*) This result is likely due to evolvability that the system discovers over evolution, measured by how good the offspring solutions are on average. (*c*) The evolvability is likely due to more varied networks motifs, taking advantage of recurrent structures. The significance is measured by comparing to randomly connected networks with the same size. This example illustrates a fundamental principle of evolvability: It emerges from the continuously changing fitness function (due to coevolution), and makes coevolution more effective, and can thus potentially be harnessed for open-ended discovery. (Figure from Reisinger and Miikkulainen 2007)

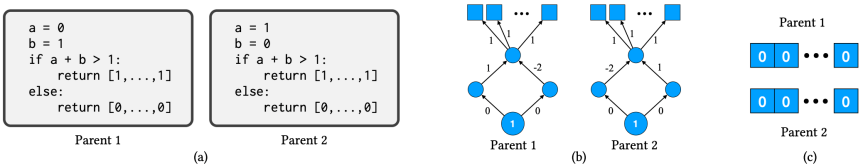


Figure 9.4: **Expressive encodings through GP and Neural Networks**. Expressive encodings make evolution more powerful by allowing for large changes. (*a*) For instance, the phenotypes of these two GP parents are all zeros, but their crossover results in an offspring of all ones with a probability of 0.25. They share most of the structure except for special segments defining the variables a and b. (*b*) A similar encoding through a neural network. The input is a constant 1, and output is all zeros; They differ in the weights of the first layer such that a crossover results in all ones with a probability of 0.25. (*c*) Direct encoding of parents cannot lead to an all-ones offspring. These simple examples illustrates how expressive encodings make such miracle jumps possible when they are not possible through direct encoding. (Figures from Meyerson, Qiu, and Miikkulainen 2022)

the NEAT networks are entirely feedforward, the GRN-based approach takes advantage of many different network motifs, many of which are recurrent (Figure 9.3*c*). In this manner, it likely discovers structures that support evolvability, and thereby coevolution, and thereby open-ended discovery.

9.1.4 Expressive Encodings

The mechanims outlined above can be captured generalized and described mathematically through the concept of expressive encodings (Meyerson, Qiu, and Miikkulainen 2022). The idea is that such encodings allow miracle jumps, i.e. large jumps in the search space: For instance, flipping all bits in a binary encoding from 0 to 1 might be such a jump. A standard evolutionary algorithm with a direct encoding would be unlikely to make such changes, and therefore could not explore the search space as effectively.

Expressive encodings do already exist. For instance, genetic programming utilizes such an encoding (Figure 9.4a). Programs may share structure, but also have segments that make large changes in the phenotype, such as conditionals. Small changes in such segments can create miracle jumps. Neural networks is another expressive encoding (Figure 9.4b): Even when they are not used as mappings from input to output, but simply to encode vectors of outputs (with a constant input), small changes in a few weights can create a miracle jump. Interestingly, such jumps may not be possible through a direct encoding (Figure 9.4c).

The usual approach to making evolutionary algorithms more powerful is to design more complex and intelligent genetic operators that capture the properties of the domain. For instance, estimation of distribution algorithms and covariance-matric adaptation evolutionary strategies aim at capturing the statistics between gene combinations and fitness (Larranaga and Lozano 2002; Nikolaus Hansen and Andreas Ostermeier 1996). In contrast, expressive encodings can work with basic, simple genetic operators such as crossover and mutation. In this sense, they capture the essence of biological expressiveness that is obtained

through interactions and development. Theoretically, both genetic programming and feed-forward neural networks with sigmoid activation functions are expressive encodings for both uniform crossover and single-point mutation.

Expressive encodings have been shown more powerful than standard evolutionary approaches in various benchmark challenges, including tasks where objectives change over time deterministically or randomly, and in large block assembly, both theoretically and experimentally (Meyerson, Qiu, and Miikkulainen 2022). The approach offers maximum evolvability, to the extent that there is no catastrophic forgetting when the objectives change. It is also similar to biology in that much of the solutions are shared—more than 99% of the genes are the same across humans, for example, and much of the DNA is shared across species (Hardison 2003; Collins, Guyer, and Chakravarti 1997). Only a few crucial differences cause the differences between individuals and species. It is this expressivity that the expressive encodings capture.

One particularly interesting opportunity for neuroevolution is to improve the transmission function over time, i.e. the probabilistic mechanisms through which the child phenotype is generated from the parent phenotypes. Evolution can be used to complexify transmission functions, thus potentially powering openended evolution. With expressive encodings and an evolving transmission function it may be possible to create a system that starts simple, solves problems as they appear, and becomes more effective at it over time. One remaining challenge is to enable transitions to more complex organizations, as will be discussed next.

9.1.5 Major Transitions

In biological evolution it is possible to identify several major transitions in complexity (Maynard Smith and Szathmary 1997; Szathmáry 2015). First there were self-replicating molecules that organized into chromosomes; then these chromosomes were enclosed into cells; next, cells complexified to include several plastids; such cells joined together and specialized to form multicellular organisms; the organisms grouped to form eusocial societies first, and then actual societies, eventually with language and culture. In each of these transitions, the individuals joined together into groups, specialized into distinct, cooperative roles, and lost the ability to reproduce independently. Throughout these transitions, information for biological organisms is still encoded at the molecular level. However, how that information is organized, transmitted between individuals, translated into physical structures, and selected for reproduction changes at each transition. As a result, what it means to be an individual becomes more complex at each transition.

While the transitions are described in detail in biology, the mechanisms that produce them are not well understood. In particular, are there multiple levels of selection operating in parallel, or only one at the highest level? How do the individuals specialize and how do they lose their individual ability to reproduce? Do multiple phases exist at the same time and cooperate and compete to eventually lead to a transition? Are the dynamics the same at each transition, or is each one a separate unique process?

Computational studies could help shed light on such questions, yet it has been very hard to emulate such transitions (Miikkulainen and Forrest 2021). The closest successes focus on defining hierarchical mathematical functions, and organizational structures in abstract

mathematical games (J. R. Koza 1992; Turney 2020; Watson and Pollack 2003). However, they are still far from major transitions in behavior. For instance, the agents might discover ways to communicate, or to construct permanent artifacts such as roads. Further evolution might then discover behaviors that take advantage of these constructs: The agents might communicate to establish flexible roles and coordinate their behavior; they may move longer distances and harness more resources. More generally, neuroevolution might construct network segments that perform useful subfunctions, then group them together to construct more complex behaviors, and multiple behaviors at different times (i.e. general intelligence). Such specialization and grouping could potentially continue for several levels.

Ingredients for such transitions have already been demonstrated in several ways. For instance, it is possible to predesign the representations at different levels by hand—-for instance, a syllabus for evolved virtual creatures allows discovering body and brains for simple locomotion first and build up to fight-or-flight in multiple steps (Daniel Lessin, Donald Fussell, and Miikkulainen 2013). Similarly, mechanisms can be created for discovering cooperative structures that work together at a higher level—for instance in CoDeepNEAT method, neural network modules are evolved to work together well in a large composite network (Miikkulainen, Liang, et al. 2023; J. Liang et al. 2019). Also, competitive process can be established that allow new challenges to emerge—such as the arms race of better runners and more challenging tracks in POET, or more complex prey behaviors and better predators in zebra/hyena simulations (R. Wang et al. 2019b; Rawal, Rajagopalan, and Miikkulainen 2010). Multiple agents can communicate through stigmergy, through observing each other, and through signaling, and thus coordinate their behavior—for example in capturing a prey or a desirable resource in a video game (Yong and Miikkulainen 2010; Werner and Dyer 1992; Bryant and Miikkulainen 2018; Rawal, Rajagopalan, and Miikkulainen 2010). Architectures and approaches have been developed for representing and executing multiple tasks in a uniform manner—for example through a common variable embedding space (Meyerson and Miikkulainen 2021).

In sum, mechanisms of cooperative and competitive coevolution, multitasking, multiobjectivity, evolvability, and expressive encodings are potentially useful ingredients in producing major transitions. However, they do not yet drive actual transitions. How such transitions can be established is an important challenge for neuroevolution—one that would also have a large impact on our understanding of biology.

9.1.6 Openended Evolution of Intelligence

Many of the possible ingredients for openended neuroevolution do already exist. The recently available computational power could be harnessed to set up evolutionary processes that harness large populations, weak selection, neutral mutations, and deep time. While many of the current indirect genotype-to-phenotype mappings still focus on a single task, the emerging theoretical understanding of expressive encodings could lead to mappings that allow searching indefinitely for more complex solutions as the environments and tasks change. Such mechanisms could be harnessed to establish evolutionary innovation that operates continuously.

However, openended innovation also requires that the environment presents the evolutionary system continually with new challenges. The environments themselves can change and evolve, or it may be possible to create multiple competing species in the environment,

thus establishing an evolutionary arms race. While current multiagent and multipopulation systems still largely focus on solving a single task, evolution in such domains has already been shown to lead to specialization and discovery of cooperation, which could lead to major transitions. Multitask and multiobjective evolution is already known to result in more robust solutions, and in such environments could lead to progressive development of general intelligence. Perhaps most promising avenue is to have the agents themselves modify the environment, building artifacts and complexity into it that persists (Lehman et al. 2022). In this manner, the environment and the agents in it can complexify indefinitely.

What goals might such experiments be set to achieve? An important one is a better understanding of biological evolution, i.e. the origins of major transitions and intelligence. Another one is to construct better artificial systems, i.e. systems that can be deployed in natural environments and social environments where they adapt to existing challenges and changes to them indefinitely–much like people do. Such ability is one essential ingredient in artificial general intelligence.

To make these ideas concrete, the next two subsections review two concrete experiments on changing environments. In the first one, the body of the agent is co-evolved with the brain in a cooperative manner. In the second, the environment is coevolved to provide more difficult challenges in a competitive manner.

9.2 Cooperative Coevolution of Body and Brain

In Section 3.2, we explored the simple idea of evolving the weights of a neural network to control a bipedal walker, and showed how to use evolution algorithms in the context of neural networks to learn a set of weights that can allow a neural network with such weights to perform a given task. If such a network is an agent operating in an environment, we have so been evolving a policy to manipulate an agent, whose design is fixed, to maximize some notion of cumulative reward. The design of the agent's physical structure is rarely optimized for the task at hand. Unlike gradient-based methods, evolution algorithms are more flexible and thus can optimize parameters beyond the weights of a neural network agent. In principle, we can even optimize parameters governing the agent as well. Why constraint ourselves to weights, when we can also optimize other important design choices governing our agents?

In this section, we explore the possibility of learning a version of the agent's design that is better suited for its task, jointly with the policy. We look at a minor modification to the environment, where we parameterize parts of an environment, and allow an agent to jointly learn to modify these environment parameters along with its policy. We show that an agent can learn a better structure of its body that is not only better suited for the task, but also facilitates policy learning. Joint learning of policy and structure may even uncover design principles that are useful for assisted-design applications.

In addition to the weight parameters of our agent's policy network, the agent's environment is also parameterized, which includes the specification of the agent's body structure. This extra parameter vector, which may govern the properties of items such as width, length, radius, mass, and orientation of an agent's body parts and their joints, will also be treated as a learnable parameter. Hence the weights w we need to learn will be the parameters of