(J. Bongard 2011). For instance in evolving locomotion, the robots may start with an eel-like body plan and gradually lose it in favor of a legged design. The gaits on robots that go through such a process can be more robust than those evolved on the legged design directly. To make morphological innovations feasible, it may be useful to protect them by temporarily reducing evolutionary selection pressure (Cheney et al. 2018). Such protection is a useful general principle in discovering complexity, similar to speciation in NEAT (Section 3.4).

The most extreme demonstration of this approach is GOLEM (genetically organized life-like electromechanics;Figure 6.5*b* Lipson and Pollack 2000). Not only were the hardware designs and the neural network controllers coevolved, but the robots themselves were 3-D printed according to the evolved designs. The designs were evaluated for their locomotive ability in simulation. The best ones were then printed and evaluated in the physical world, and found to perform as expected. The evolved virtual creatures (Dan Lessin, Don Fussell, and Miikkulainen 2014; Daniel Lessin, Donald Fussell, and Miikkulainen 2013) discussed in Section 14.5 extend this approach to more complex morphologies and behaviors, all the way to fight-or-flight, albeit in simulation and with a hand-constructed syllabus. However, it is possible to imagine a future where robot bodies and brains are coevolved automatically, the results created on multimaterial 3D printers—and once the printing is finished, the robots wake up and walk off the printer on their own.

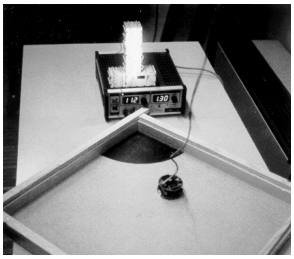
Evolutionary robotics has already been scaled up to swarms, i.e. robot teams that exhibit collective behavior (Dorigo, Theraulaz, and Trianni 2021; Trianni et al. 2014). The challenge in this area is to evolve the swarm to perform tasks that single robots could not. For instance, such robots can hook up and form a linear train that can get over obstacles and gaps that a single robot could not (Figure 6.5*c*). Many interesting issues come up in evolving neural controllers for such robots. For instance, should they all be clones of each other, or each evolved to fill a specific role in the team? Collective behavior in general is an important area of neuroevolution, discussed in depth in Chapter 7.

6.1.4 Discovering flexible strategies

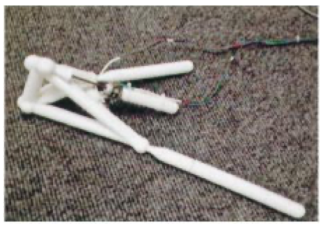
The neuroevolved solutions so far have focused on control. At this level, adaptation most often means modulating or adjusting a single existing behavior: Throttle one of the engines a little more, move one leg a little faster, flap a little harder. When behavior extends from such low-level control to a high-level strategy, goal-driven coordination of multiple behaviors is required. For instance, offensive vs. defensive play in robotic soccer may require getting open vs. covering an opponent; actions required of a household robot are very different when it is vacuuming vs. emptying the dishwasher vs. folding laundry; game agents may need to gather resources, attack, and escape.

Evolving high-level strategies is challenging not only because the agent must have command of a much larger repertoire of behaviors, but it also needs to know when and how to switch between them. Proper switching is difficult for two reasons: first, in some cases it may have to be abrupt, i.e. small changes in the environment may require drastically different actions; second, sometimes the different strategies need to be interleaved or blended instead of making a clean switch.

The first challenge can be illustrated e.g. in the half-field soccer domain, where five offenders try to score on five defenders, using eight behaviors: getting open and intercepting the ball, and holding the ball, shooting at the goal, and passing it to one of the four



(*a*) Evolving control in hardware



(*b*) Coevolving morphology and control

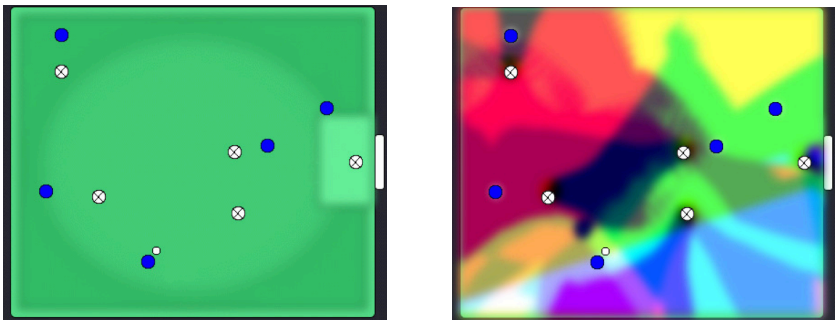
(*c*) Swarm robots working together

Figure 6.5: **Neuroevolution in Evolutionary robotics.** While robotics generally focuses on hardware designs, it is difficult to construct controllers by hand, especially with novel and variable designs. Neuroevolution is often a useful approach in many such cases. (*a*) Neural network controllers can be evolved directly in hardware, for instance to develop homing behavior in Kheperas. The light source identifies the corner with the charging area (painted in black). (*b*) It is possible to evolve the robot morphology and control together, and 3D print the designs, in essence evolving artificial life forms. (*c*) Swarms of robots can perform tasks that single robots may not, such as traversing over holes in the ground. In this manner, neuroevolution makes it possible to develop behaviors for a wide variety of robotic designs. [Figure (*a*) from Floreano and Mondada 1996a, Figure (*b*) from Lipson and Pollack 2000, and Figure (*c*) from Trianni et al. 2014.]

teammates (Figure 6.6; Kohl and Miikkulainen 2011). Depending on the position of the ball, teammates, and opponents, boundaries between these behaviors are very tricky. If an opponent moves even slightly to block a teammate, passing becomes infeasible; if an opponent crosses a threshold distance, holding becomes infeasible. Furthermore, actions that interpolate between these behaviors are not possible: They have to be performed fully or not at all. Thus, the domain can be described as fractured: as the state of the world changes, the correct actions change frequently and abruptly.

It is very difficult for neuroevolution to discover such fractured strategies. In most domains, continuous control works just fine, i.e. when the situation changes a little, the control output changes a little, and continuously so. Neural networks represent such continuity well naturally; in contrast, hard switches are difficult to establish. However, the network architecture can be designed to make them easier to discover, in two ways: (1) instead of sigmoid activation functions, radial basis functions can be used. They each activate a neuron in a specific local region, making it easier to cover fractured decision boundaries. (2) the network topology can be constructed in a cascaded manner, i.e. complexifying by adding neurons as extra layers on top of the existing network, instead of anywhere in the network as usual in NEAT. Such cascade allows each new neuron to implement a refinement of existing behavior, gradually forming more fractured decision boundaries. Indeed in domains like half-field soccer, such approaches perform much better than handcoded solutions as well as standard reinforcement learning and neuroevolution techniques.

A second challenge in constructing an effective strategy is that switching between behaviors needs to be flexible. In some cases, such as switching between batting and fielding in baseball, or vacuuming and emptying the dishwasher, the behavior changes entirely for a



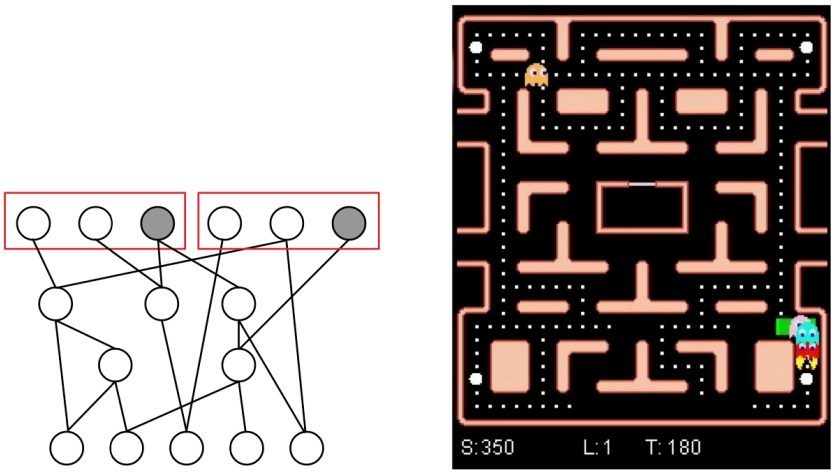
(*a*) Game situation (*b*) Values of actions

Figure 6.6: **Fractured high-level strategy in half-field soccer.** High-level strategies are difficult to discover and implement because they often require changing behaviors abruptly based on small changes in the input. (*a*) For instance in half-field soccer, five offenders (blue dots) try to score on five defenders (white dots) by holding the ball, passing to one of the teammates, and shooting. (*b*) Visualization of successful actions for an offender with a ball at various locations in the field given the positions of all other players. Each color represents a subset of actions that would be successful. Small changes to just this one variable have a large effect on success, making good strategies highly fractured and difficult to evolve. Neuroevolution with local neurons and cascaded refinement is an effective approach in such cases. For animations of these behaviors, see https://neuroevolutionbook.com/neuroevolution-demos. (Figures from Kohl and Miikkulainen 2011)

long period of time. Such tasks are isolated and can be implemented even with different neural networks and a switch network that decides between them. However, in other cases the behaviors are interleaved, occurring several times in rapid succession. For instance, the possession of the ball in soccer can change rapidly, requiring the players to switch between offensive and defensive play often, and even anticipate such switches. In yet others such as dodgeball, the offensive and defensive behaviors are blended because there are multiple balls at play, and a player may attempt to throw a ball at the same time as avoiding getting hit by one. Thus, intelligent agents must be capable of different behaviors at different times, as well as interleaving and blending them.

A good platform to study such behaviors is the Ms. Pac-Man video game (Figure 6.7 Schrum and Miikkulainen 2016). In a maze, the player eats pills while trying to avoid getting eaten by ghosts. Upon eating a power pill, the ghosts become edible too. Thus, the behaviors of running away from threatening ghosts and approaching edible ghosts are interleaved. However, as soon as a ghost is eaten it returns as a threat, and at that point, the tasks are blended: The player has to run away as well as approach some of the ghosts at the same time. With slight modifications to the game, isolated tasks can be studied as well, i.e. by fixing the ghosts to be either threatening or edible.

A network controlling Ms. Pac-Man sees the state of the game e.g. as distances to pills, power pills, and ghosts in different directions, and whether the ghosts are edible. As its output, it decides which way to move. A simple such network can be evolved e.g. with NEAT but it does not perform very well: It has a difficult time separating the different



(*a*) Preference neuron architecture (*b*) Invoking the luring module

Figure 6.7: **Discovering effective and surprising multimodal task divisions**. Behavioral strategies are often multimodal, i.e. require performing different behaviors at different times. Modular network structures are a natural way to encourage multimodal behavior to emerge. (*a*) A powerful approach is to evolve a network with multiple output modules together with preference neurons (grey) to indicate when each module should be used to control the agent. (*b*) Such a system may discover surprising task divisions. For instance in Ms. Pac-Man, instead of separating threatening and edible ghost situations to different modules, it separates general easy movement into one module, and behavior when ghosts are close into an escape module (active during the green trace). That module is used to lure the ghosts nearby and then escaping to eat a power pill; afterward the movement module is used to eat up the ghosts (which is easy because they are nearby), resulting in a high score. Such division and behavior would be difficult to discover and prescribe by hand, yet evolution discovers it as an effective solution to a multimodal game. For animations of these behaviors, see https://neuroevolutionbook.com/neuroevolution-demos. (Figure (*a*) from Schrum and Miikkulainen 2016)

behaviors, and tends to blend them and not perform any one of them very well. This result indeed illustrates the main challenge in learning high-level strategies with neuroevolution.

The opposite approach would be to have a human expert identify what behaviors are needed, and evolve each one separately, as well as a selection neural network that decides which behavior needs to be used when. This approach works well when the tasks are clearly separated (e.g. fight-or-flight in Section 14.5), but it can also work when two behaviors need to be combined, such as evading a predator while simultaneously catching a prey (Jain, Subramoney, and Miikkulainen 2012).

However, it may also be possible to learn multiple behaviors in a single network, taking advantage of commonalities between them. For instance, it is possible to evolve a single multitask network with different outputs to control Ms. Pac-Man when the ghosts are threatening and when they are edible. The division is not learned but implemented algorithmically. This approach works well with isolated and interleaved versions of the task. Since

the same part of the network is used consistently in similar situations, evolution discovers effective offensive and defensive behaviors. In blended situations it is not effective though. A third set of outputs can be evolved for such situations, but it does not learn very well.

A fourth approach is to let evolution discover when to use what strategy. That is, each of the output modules is coupled with a preference neuron that indicates how strongly the network believes the corresponding output should be used. In this setting, evolution might be expected to discover offensive and defensive strategies, and how to switch between them. However, it discovers a much more sophisticated, and surprising approach. The strategies that evolve are not offensive and defensive, but instead behaviors that apply to easy and difficult situations. That is, one output module controls Ms. Pac-Man when she is running around eating pills when no ghosts are nearby, whether they are threatening or edible. A second module specializes in escaping when threatening ghosts are nearby. With these modules it implements a highly effective luring strategy: It lets the ghosts get close, then escapes them to the nearby power pill—and is then able to eat the ghosts effectively because they are close!

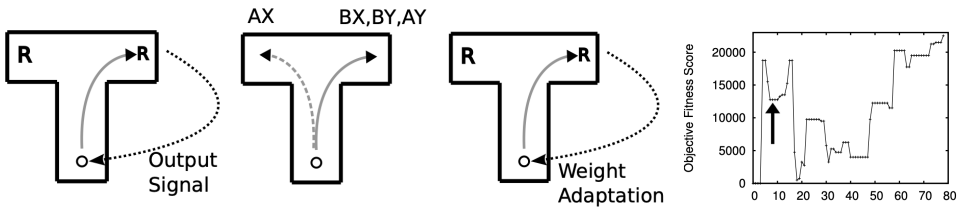
Even though the escape module is rarely active, it is crucial in obtaining a high score in the game. Therefore, half the network is dedicated for this behavior. Such a strategy would have been difficult for human designers to prescribe, yet evolution discovered it as the most effective way to play the game. It demonstrates how effective high-level strategies are not only composed of multiple behaviors, but of intelligent ways of combining them. It also shows that if evolution is allowed enough freedom to explore, it can discover surprising and effective such combinations.

6.1.5 Evolving cognitive behaviors

One potentially important role for novelty search and related methods is in discovering cognitive behaviors such as communication, memory, and learning. Such behaviors are complex and challenging to evolve, and several approaches have been developed to discover them (see e.g. Section 14.7.2; Saunders and Pollack 1996; Ollion, Pinville, and Doncieux 2012; Risi, Hughes, and Stanley 2010; Yamauchi and Beer 1993). They illustrate different challenges and ways to overcome them, often through carefully crafted domains and fitness functions based on domain knowledge. A possible reason, evident even in the most rudimentary versions of these behaviors, is that they require overcoming deception.

For instance, in order to evolve communication, it is necessary to discover what and when to communicate, the mechanisms to send a signal, to receive it, and to interpret it. Each one of these mechanisms requires extra hardware that does not provide an evolutionary advantage unless all of the mechanisms are functional at once. They are thus deceptive, and it is unlikely that evolution would stumble into them all at once. Also, if a partial solution is found, it is difficult for evolution to discard it in favor of a better one (Floreano et al. 2007). They could, however, be discovered as stepping stones by novelty search, making communication more likely to be discovered.

As an illustration of this idea, consider an agent in a T-maze Figure 6.8; Lehman and Miikkulainen 2014. Each agent is controlled by a neural network whose activation is reset before each trial. In each trial, the agent starts at the bottom end. It needs to move to the intersection and decide whether to go left or right in order to get to the reward. An evaluation consists of multiple trials during which the reward stays in one place, but the reward can



(*a*) Communication (*b*) Memory (*c*) Learning (*d*) Solution lineage

Figure 6.8: **Overcoming deception in the evolution of cognitive behaviors.** During an evaluation that consists of multiple trials, the agent needs to use (*a*) communication, (*b*) memory, or (*c*) learning to navigate to the reward in the T-maze reliably. Even when the necessary elements for these abilities are available, fitness-based evolution cannot discover how to put them together. Instead, it only discovers reactive behaviors, i.e. always going to the left or the right. In contrast, they serve as stepping stones for novelty search, which eventually discovers effective cognitive behavior. Thus, the lineage of an eventual successful agent in novelty search includes many drops in fitness (*d*). For instance, the novel behavior of going to the opposite corridor with some inputs (arrow) turns out to be a useful stepping stone in discovering communication. Figures from Lehman and Miikkulainen 2014

move to the opposite end between evaluationss. Thus, if the reward does not move very often, or is most often found in one location, evolution can develop a simple strategy that is better than chance: Go to the location where it is found more often and/or more recently. However, if the reward moves frequently enough, communication, memory, or learning is needed to capture it more reliably.

In a communication task, the agent can generate a signal at the end of the trial, and the agent in the next trial will receive it at the start. A successful communication thus indicates whether the agent should turn left or right at the intersection. In a memory task, the agent will receive an A or B signal and then an X or Y signal, before it can start to move. The AX combination indicates the reward is at left, others indicate that it is at right. The agent thus has to remember the combination of two signals in order to act appropriately. In the learning task, the agent can adapt the network's connection weights through modulated learning rules (Risi, Hughes, and Stanley 2010) after each trial to make a successful outcome more likely. These weight changes persist throughout the evaluation.

Indeed, fitness-based evolution in this domain developed a reactive strategy of always going to the left or right, depending on frequency and recency. This strategy was successful only in less than 20% of the trials. Even when communication, memory, and learning were available, evolution could not find a way of taking advantage of them—in other words, it could not overcome deception. However, with novelty search, evolution was able to discover communication, memory, and learning strategies that were successful in approximately 79%, 81%, and 57% of the trials. Analysis of the lineages of eventual solutions show that novelty search was indeed utilizing stepping stones, i.e. behaviors that received lower fitness on their own, but turned out useful in constructing the final communication, memory, or learning-based strategy.

Although the behaviors in the T-maze are simple, they are intended to capture the essential challenge of discovering cognitive structures. The results thus suggest that straightforward objective-based evolution is unlikely to discover cognitive behaviors, and thus novelty search and perhaps quality diversity methods are essential.

6.1.6 Utilizing stochasticity, coevolution, and scale

In many virtual domains, whether games or training environments, it is important that the virtual agents are not entirely predictable. That is, their behavior should be nondeterministic (or stochastic) to some degree, so that the simulation leads to a wider variety of situations and challenges. Similarly during training, the agents then encounter a wider variety of situations and may learn more robust and comprehensive behavior.

The action-unit coding at the output of the agent is generally a powerful approach: The action represented by the most highly activated output unit is chosen at each time step. Especially early in evolution, it is easier to find such networks rather than networks that would output continuous values (representing a range of actions) accurately.

If the agent networks were trained with backpropagation, such value-unit encoding would result in a probability distribution, i.e. for each input, the activations across the output units would indicate the probabilities of the correct action (Morgan and Bourlard 1989). However, such distributions do not develop automatically in neuroevolution. The networks may be able to identify the winner, i.e. develop the highest activation on the correct output unit, but the activations of the other units do not develop into probabilities: They do not matter for performance, and therefore can be anything, as long as they are lower than that of the winning unit.

However, evolution can be guided to develop probabilities with the simple technique of stochastic sharpening (Bryant and Miikkulainen 2006). From the beginning, the output activation values are treated as probabilities: They are normalized to sum up to 1.0, and the action to be performed is selected stochastically weighted by these values. For instance in the Legion-II domain, initially the action values were relatively uniform, generating a lot of randomness, but over evolution they became sharper, leading to more effective performance. However, the performance even in the end was somewhat stochastic, resulting in the kind of believable and interesting gameplay that would be difficult to achieve otherwise.

Interestingly, stochastic sharpening also improves the search for effective behaviors, and such agents eventually outperform those evolved without it. They are exposed to more situations during evolution, and thus evaluated more comprehensively. Their behavior becomes more consistent because unexpected situations do not throw them off. They also avoid output race conditions, i.e. situations where two output unit activations are almost exactly the same, resulting in unreliable choices. Thus, stochastic sharpening is one simple tool that can make behavior more effective, so much so that it may even be worth converting continuous domains to action-unit coding just to take advantage of it.

One important principle in evolving complex behavior that has not yet been discussed is coevolution, i.e. evolving the behavior in competition with other agents, or in cooperation with other agents. This is the topic of Chapter 7, and in a sense it thus continues the discussion of this section.

Another important topic for the future is the evolution of behavior in large-scale networks. In particular, transformer architectures have shown surprising power when scaled up

to billions of parameters, or million times more than many of the networks discussed in this section (Ouyang et al. 2022). One way to characterize this power is that such scale solves the problem of variable binding, or dynamic inferencing, that has limited the generality of smaller networks. For example, if trained with sentences of type 1 composed of words of type A, and sentences of type 2 composed of words of type B, such networks would not generalize to 1-sentences with B-words, and 2-sentences with B-words. Large language models perform such generalization routinely, if they are large enough: For instance, they can write technical instructions in the style of Shakespeare, never seen together in the training corpus.

Interestingly, a large scale is necessary for this ability to emerge. Transformers are based on attention, i.e. discovering useful relationships between input tokens. While the performance of large language and image models is not yet fully understood, it is possible that with a large enough scale, such models start learning relationships between abstractions as well. It would be interesting to see if scale has a similar effect in generating complex, robust, multimodal behavior. It may be possible to use existing pre-trained foundation models in language or vision as a starting point, and evolve behavior generation as a modification or augmentation to them. Or perhaps it will be possible to construct a foundation model for behavior from scratch through the imitation of massive datasets? Or maybe neuroevolution methods can be scaled to large models, and behavior discovered through massive simulations? Research on such scaleup forms a most interesting direction for future work.

6.2 Decision Making

Intelligent behavior, as discussed above, focuses on agents that are embedded in a real or simulated physical environment and interact with it through physical sensors and effectors. In contrast, intelligent decision making focuses on behavior strategies that are more abstract and conceptual, such as those in business and society. Neuroevolution can play a large role in decision making as well, but the approaches and opportunities are distinctly different. They often need to take advantage of surrogate modeling, and take advantage of human expertise, as discussed in this section.

6.2.1 Successes and challenges

To begin, note that human organizations today have vast amounts of data that describe their operation: Businesses record interactions with their customers, measure effectiveness of their marketing campaigns, track performance of their supply chains; health-care organizations follow the behavior of patients, measure effectiveness of treatments, track performance of providers; government organizations track crime, spending, health, construction, economy, etc. Such data has made it possible to predict future trends. Predictions are then used to decide on policies, i.e. decision strategies, i.e. prescriptions, in order to maximize performance and minimize cost.

Discovering optimal decision strategies is an excellent opportunity for neuroevolution. Optimal policies are not known; they involve a large number of variables that interact nonlinearly; the observations and outcomes are often partially observable and noisy; often several conflicting objectives, such as performance and cost, must be optimized at the same time. They are therefore well suited for representation in neural networks, and discovery through evolution.