Covariance Matrix Adaptation MAP-Annealing (CMA-MAE) Fontaine and Nikolaidis 2023 introduces a nuanced alteration in the ranking mechanism. This change gradually reduces the influence of elites in filled cells of the archive, ensuring that the optimization process does not prematurely shift focus from the objective to exploration. This issue is especially pertinent in cases involving flat objectives or low-resolution archives. Remarkably, this modification is compatible with both CMA-ME and CMA-MEGA, broadening its applicability.

5.5 Multiobjectivity

While quality diversity focuses on two objectives, one on performance and the other on diversity, multiobjective optimization in general is a good approach to maintaining diversity in evolutionary computation. The motivation once again comes from biology (Miikkulainen and Forrest 2021). Biological fitness is complex: animals must seek food and shelter, avoid predators, find mates, and care for the young, and often some of these objectives conflict. The problem can be solved in many ways, leading to multiple niches, and such diversity leads to powerful further adaptation.

Note however that biological objectives can be expressed simply as a single high-level objective: survival of the species. A similar approach can be taken in evolutionary computation, i.e. a complex optimization task can be expressed simply as winning a game, making a lot of money, or gaining a lot of publicity. Such objectives allow evolution to be creative; on the other hand, the fitness signal is weak and may not allow identifying good ideas until they are fully developed. This approach may need to be paired with neutral mutations, weak selection, and deep time, placing it closer to biological evolution (Section 9.1.1).

Multi-objective optimization can thus be seen as a practical approach one level below such a high-level specification. It is often possible to devise performance objectives, cost objectives, and secondary objectives such as simplicity, accuracy, or appearance, without specifying the desired solutions directly. In many cases it is useful to have a Pareto front as a result, i.e. a collection of solutions that each represents a different tradeoff between them such that no solution is better than any other across all objectives. One solution in the Pareto front can then be chosen according to other criteria, such as conditions at deployment time, or human preferences that are difficult to express as objectives.

Because evolutionary computation is a population-based search method, multiobjective optimization is natural, and several methods have been developed for it (Deb et al. 2002; Zhang and Li 2007; Coello Coello, Van Veldhuizen, and Lamont 2007). Typically they work well up to half a dozen objectives, after which the Pareto front starts to have too many solutions (fewer solutions are better than others across all objectives). Other techniques have been developed for many-objective optimization, up to hundreds or thousands of objectives, representing a large number of constraints or tests (Deb and Jain 2014; Ishibuchi, Tsukamoto, and Nojima 2008). Multiobjective formulation is often a natural way to approach the problem, as demonstrated many times in this book (e.g. Sections 6.2.3-6.2.4, 10.4-10.5, and 14.2).

Multiobjectivity is also a natural way to boost diversity: with multiple objectives, there are many ways of being successful. Niching or speciation may emerge in the population, and may be further separately encouraged with mechanisms such as those in NEAT. Species

can then be used to form ensembles, taking advantage of the diversity. Such methods are reviewed in Section 5.6.

5.6 Ensembling

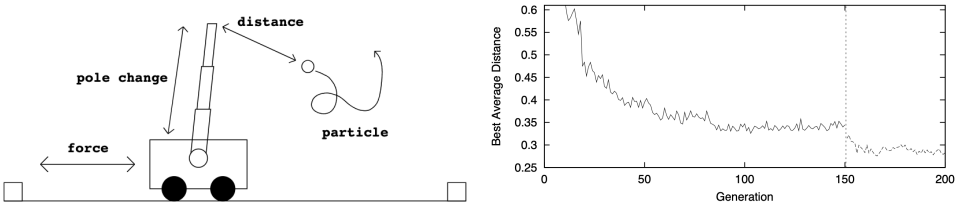
In general in machine learning, it is often a good idea to train multiple different models for the task, and then form the final system by ensembling them. The idea is that each model is somehow different, e.g. has a different architecture, is initialized differently, or is trained with different training samples. Thus, each of them may end up learning something the other models do not, and together they can perform better than any model alone. This idea is consistent with studies in psychology, social science, and business that suggest that diversity in human teams leads to improved decision-making (Rock and Grant 2016).

Ensembling may be as simple as just averaging the outputs of multiple models, or combining them more intelligently, or selecting one model that's most likely to have the correct answer for each input. Methods have also been developed, such as Mixtures of Experts (Masoudnia and Ebrahimpour 2014) and RHEA (Section 6.2.5), to train and combine different models more systematically. The fact that ensembling works is statistically surprising and was controversial for a while, but there is now a good understanding of it, especially in classification tasks (Li, Wang, and Ding 2018). Ensembling intelligent agents requires more complex methods because behavior often depends on sequences of inputs and decisions and is often based on recurrent neural networks, but it is possible as well. Ensembling is thus part of the standard machine learning toolbox and can be used routinely to improve performance.

Ensembling is a particularly natural extension of evolutionary approaches. EAs create and maintain a population from which the ensemble can be drawn. Moreover, having a diverse set of candidates is crucial both for evolution and ensembling. Often the individuals in the final population end up with slightly different skills, from which an effective ensemble can be formed (Islam and Yao 2008). Examples of such diversity include e.g. the age-estimation network architecture (Section 11.3.6) and training with population culture (Section 5.7). Such diversity is even more pronounced when the task is multiobjective: Individuals in the Pareto front form a natural pool from which to select ensemble members.

The NEAT neuroevolution method also employs a speciation mechanism that encourages diversity in search (Section 3.4). In effect, NEAT runs multiple island-based evolutionary processes, i.e. separate subpopulations that only periodically cross over, and species that are created and removed dynamically as evolution progresses. The species are created and maintained based on topological (i.e. genetic) diversity, but they result in enough behavioral diversity for ensembling to be effective. Indeed, it is possible to use just the species champions as the members of the ensemble, and then add a voting, averaging, winner-take-all, or gating as the ensembling mechanism (Pardoe, Ryoo, and Miikkulainen 2005).

Note that ensembling is related to many neuroevolution ideas and mechanisms discussed in this book. For instance, the main idea of the ESP method (Section 7.1.1) is to evolve neurons for each location in the network in separate subpopulations; because good performance requires different neurons, diversity across populations is automatically maintained, and neurons are evolved that cooperate well together. Such a network can be seen as an ensemble with a very strong combination mechanism. Similarly to the hierarchical mixtures of



(*a*) Particle chasing task (*b*) Improvement through ensembling

Figure 5.8: **Effect of simple ensembling in a complex control task.** (*a*) When the cart-pole task is extended with an extensible pole, it becomes a fly-swatting task. The control dynamics change constantly as the pole changes, making control highly context dependent and well-suited to ensembling. (*b*) The population of controllers are first evolved with NEAT for 150 generations, and once the performance plateaus, a gating network to select among eight species champions. The performance improvement is significant and immediate, suggesting that ensembling is a simple and reliable way to boost performance of neuroevolution experiments. (Figures from Pardoe, Ryoo, and Miikkulainen 2005)

experts approach in machine learning, ESP can be extended hierarchically to construct a team of networks, where each network receives different inputs. For instance, each network can keep track of a different opponent, and at the highest level, a combiner neural network decides what action to take (Rajagopalan et al. 2011). This approach was used to evolve both the prey and the predator agents in the coevolutionary arms race example described in Section 7.2.2.

In MM-NEAT (Section 6.1.4), multiple modules emerge from the evolution of a single network. They can be seen as ensemble members, and the preference neurons in each module as the ensembling mechanism, suggesting how the module output should be combined. Such preference neurons can be evolved in separate networks as well: In essence, each network places a bet that they have the right answer (Bruce and Miikkulainen 2001). They are evolved to maximize the return from their bets, and as a result, the bets serve as confidence estimates. Ensembling then consists of simply selecting the network with the highest confidence. The context+skill approach (Section 6.1.2) can also be seen as an ensembling mechanism. There are two special ensemble members, one representing context and the other the most likely action, and a combiner network on top representing the ensembling mechanism.

However, the most straightforward ensembling approach can already be useful in neuroevolution: A NEAT population can be evolved in a control task first, and then a gating neural network evolved to select which controller to use at each step. The approach was applied to a more challenging version of the pole-balancing task where the pole is actually a telescope that can change its length, and the pole's tip chases a moving target particle–as if trying to swat a fly (Figure 5.8). Even though there's only a single pole and the controller sees the positions and velocities (so that recurrency is not needed), the response of the pole changes with its length. Thus, the actions change the dynamics of the task, requiring the controller to adjust its strategy continuously. Such flexible control is hard to achieve with a single neural network, but easier with an ensemble. After evolving a population of controller neural networks for 150 generations, the species champions were used as an

ensemble. A gating neural network was then evolved for another 50 generations to pick one network to control the system at each step. The performance improvement was significant and immediate, demonstrating how even simple ensembling can add value to an existing neuroevolution approach.

The approach could easily be extended with various techniques to fit particular problems. For instance, diversity of the ensemble population could be increased by making evolution multiobjective. Secondary objectives may be defined naturally in many domains (such as speed, or cost, in addition to accuracy), but novelty is always a possible such objective, and highly effective in promoting diversity (Section 5.3). Or, the ensemble members could be evolved to optimize not their own performance in isolation, but performance as a useful member of the ensemble (García, Hervás-Martínez, and Ortiz-Boyer 2005). This approach could boost the performance of even the simplest ensembling methods, like voting, averaging, or gating.

Further, the gating network could be evolved not simply to select, but to combine the outputs of the population members, similar to context+skill approach. The ensemble members could indicate confidence as part of their outputs, and the combiner could take that into account in constructing its actions (instead of simply selecting the most confident network). The ensemble and combiner networks could be co-evolved to maximize the performance of the ensemble, similarly to hierarchical ESP and CoDeepNEAT (Sections 7.2.2 and 10.3.2).

In this manner, the general idea of ensembling can take many forms in neuroevolution. However, it should always be part of constructing the solution. Without some kind of ensembling in the end, a neuroevolution experiment often leaves money on the table.

More broadly, the simple success of ensembling offers a powerful lesson to problem-solving and decision-making in general: Diverse teams with multiple viewpoints are likely to perform better than individual experts, provided that there is some principled way of combining these viewpoints. Ensembling provides a simple such way: egalitarian learning, described in the next section, extends it further with learning.

5.7 Utilizing Population Culture and History

The knowledge that exists in the population beyond a single individual can be seen as population culture. There are common elements to it, i.e. knowledge that many individuals share such as common behaviors, variations of this common knowledge, and also elements unique to single individuals. Generally culture operates at a time scale between learning and evolution, but can also emerge even during lifetime of individuals, and can last as long as the population. It can also include artifacts that exist outside the population. They may be essential in establishing open-ended evolution in that they permanently alter the environment where evolution takes place (Lehman et al. 2022).

In evolutionary computation, population culture can be utilized in many ways to make evolution more effective Belew 1990; McQuesten 2002; Spector and Luke 1996; Reynolds, Michalewicz, and Cavaretta 1995; Maheri et al. 2021. Just like in human societies, and essential element of it is diversity. The population includes many different kinds of solutions; the power of cultural algorithms comes from exploiting such diversity.

The simplest way is to utilize diversity in a single generation of offspring. That is, instead of generating the usual two offspring at each crossover, dozens or hundreds are created.

They are then quickly evaluated and only the post promising few are kept—and they are most likely better than those two resulting in the normal process. This mechanism, called culling, is based on the observation that most crossovers are awful (Whitley, Dominic, and Das 1991; Nordin and Banzhaf 1995), i.e. result in offspring that are weaker than the parents. This effect is especially severe in neuroevolution with competing conventions, where most crossovers are wasted on incompatible individuals. Some algorithms forgo crossover entirely and only rely on mutation. However, crossover is an important vehicle of adaptation in biology, so somehow our implementation of it is lacking. Culling is a way of trying to fix it. It is motivated by biology in that embryos that are not viable are discarded early in gestation, and litters are often much larger than one or two individuals. There are probably other mechanisms at work as well in biology that make crossovers more productive than crossover in computation, such as more complicated genotype-to-phenotype mappings (Miikkulainen and Forrest 2021). They can be partially modeled by making culling more extreme, i.e. generating more offspring and retaining only a few of them, which is easy to do in evolutionary computation.

The challenge in culling is to recognize the few most promising offspring without having to run a full fitness evaluation on the whole set. If that is possible, then culling can speed up evolution. It turns out that such approximate evaluation is possible through culture. A set of inputs can be formed, i.e. a set of questions, or a syllabus if you will, that's then given to each offspring see how they respond. Those answers can then be compared to answers that other prominent population members would create, such as the parents or population champions. Those offspring whose answers are very different from the culture can then be culled. Even though the hope is that some offspring's answers differ because they are better than anything seen before, this process is effective in identifying offspring that are the worst, i.e. nonviable. Most crossovers are awful, it is enough to discard only those. This process can be very effective, for instance speeding up evolution by a factor of three or more in neuroevolution for the pole-balancing task (McQuesten 2002).

Similar cultural mechanisms can be applied to other parts of the evolutionary process. For instance in selecting parents for crossover, the main goal is to combine good traits of both parents. This goal is challenging because fitness alone does not tell the full story. Sometimes good genes are incompatible with or dominated by other genes in the individual, resulting in poor fitness overall (as will be seen in Section 6.2.5). Therefore, parents should be chosen not only based on fitness, but also distance. That is, the parents should be close enough in the genotypic space to be compatible, but different enough so that crossover will generate something new. In this manner, combining the strengths of both parents becomes more likely.

One practical implementation of this idea is to select the first parent based on fitness only, as usual, and the second to complement it—that is, while still competent in fitness, to be as different from the first as possible. The difference can be measured based on the answers in the syllabus, as in culling. It turns out that in neuroevolution for the acrobot task (i.e. swinging the jointed pole upright), a better offspring is generated twice as often as without such parent selection (15% of the time instead of 7%) (McQuesten 2002). Note that the second parent is usually much worse in fitness, so such high fitness is likely achieved by combining complementary strengths.

Culture can also be used to maintain diversity directly by focusing on which individuals are discarded from the population to make room for new offspring. Usually the individuals with the poorest fitness are removed, but diversity can be used as a secondary measure. One way to implement this idea is to find two pairs that are the closest in the population in terms of the answers to the syllabus, and then discarding the less fit of them. Again in acrobot neuroevolution, such a mechanism resulted in populations that were three times as diverse (in average distance in answers to the syllabus), making evolution 30% faster (McQuesten 2002).

A fourth way of taking advantage of culture is to use it to leverage learning in evolution. As discussed in Section 4.2.3, the syllabus of inputs can be paired up with answers of the parents or population champions, and then used as a training set for gradient descent. In this manner, those offspring that has the best learning potential can be identified. Even when the learned weights are not coded back into the genom, evolution becomes more effective through the Baldwin effect, i.e. more informative selection of offspring. In pole balancing, this mechanism can make neuroevoultion an order of magnitude faster (McQuesten 2002).

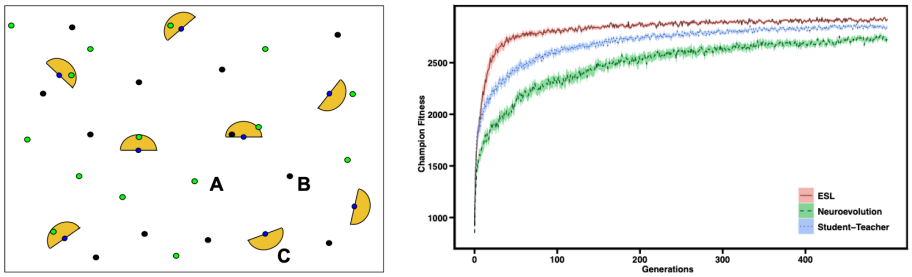
However, even better use of this idea can be made by taking advantage of diversity in the population culture. That is, the behaviors of all individuals in the population serve as the cultural heritage; individuals can learn from any of these behaviors, and such learning can guide genetic evolution in a more diverse and effective way.

At the outset, it is not clear that this idea would work. To be sure, dividing the population into teachers and learners, and utilizing parents and population champions as teachers, makes sense: The new and poorly performing individuals in the population are trained to be more like those that are known to perform well. However, such training is also bound to reduce diversity. Much of the population starts copying a few good individuals, which may make it more difficult for evolution to discover new solutions.

Also, even though the parents and champions perform well overall, some of their actions can still be quite poor during evolution. Conversely, there may be other individuals in the population who perform very well in specific situations, even though they do not perform that well overall. In more broad terms, in evolutionary computation as in society in general, any individual may have something useful to teach to any other individual. This is one reason why diverse teams in general may be more innovative than teams that are not (Rock and Grant 2016).

This principle can be captured computationally in a method called Egalitarian Social Learning (Tansey, Feasley, and Miikkulainen 2012). The idea is that each agent A observes the performance of each other agent B in various situations in the task. If B receives a high reward in a situation x where A receives a low reward, there is a learning opportunity for A. A training example is formed with x as input, agent B's action y as output, and gradient descent is used to modify agent B. In a sense, the entire set of agents and their behaviors forms a population culture. Each agent is then trained to adopt those aspects of the culture that are the most successful.

This approach works in domains where rewards can be obtained frequently, and associated with partial behaviors. To enhance diversity, it is possible to divide the population into subcultures. Agents in each subculture teach and learn from the other agents in the same subculture, making it less likely for the population to converge prematurely. The approach



(*a*) The foraging domain (*b*) Foraging fitness over evolution

Figure 5.9: **The effect of diversity and egalitarian learning.** A population of agents need to forage in an environment with good bad objects. (*a*) The agents gain fitness by consuming food items of various positive values (A), and avoiding items of negative values (B). They have a limited view (C), requiring them to move around a lot to find the items. With direct neuroevolution, several strategies develop, some taking advantage of covering a lot of ground, and others taking advantage of being careful not to miss anything. (*b*) With egalitarian social learning (ESL), the evolved agents can also learn from each other during their lifetime. ESL achieves higher fitness by generation 50 than direct neuroevolution or a student-teacher approach by Generation 500. This experiment thus demonstrates both the value of diversity and of learning from population culture. For animations of these behaviors, see https://neuroevolutionbook.com/neuroevolution-demos. (Figures from Tansey, Feasley, and Miikkulainen 2012)

can be implemented through Lamarckian evolution or the Baldwin effect. When diversity is maintained through subcultures, Lamarckian evolution may be more effective.

The approach was demonstrated in a foraging domain where food items are randomly scattered and vary in their value from very good to poor to outright poisonous (Figure 5.9). The agents sense these items in eight 22.5^o sectors in front of them and also sense their own velocity. As their output, they control their velocity and orientation. With egalitarian learning, many different strategies evolve. Some subcultures focus on high-speed exploration in order to utilize high-valued food. Others move slower and carefully consume all positive food items. Overall, the egalitarian population is significantly more effective in utilizing the available food resources than a comparable student-teacher model, and direct neuroevolution. The experiment thus illustrates the value of diversity in a team of agents, as well as the value of egalitarian learning.

Instead of using the diverse solutions in a population for training, the knowledge in such solutions can be abstracted into a statistical model that then guides evolution. The model predicts how likely the different combinations of elements in these solutions are to result in high fitness. The approach is similar to CMA-ES (Section 2.1.3), which uses a model to make intelligent mutations, and estimation of distribution algorithms (EDAs; Lozano et al. 2006), where solutions are constructed step by step using a statistical model such as a Bayesian network or a Markov random field. At each step, the model is used to determine which further elements would be most likely to result in good solutions, given the elements chosen so far (Pelikan, Goldberg, and Cantú-Paz 1999; Alden and Miikkulainen 2016; Prior 1998).

Instead of building a model of gene statistics, it can be built for neurons or modules that form a network in approaches such as SANE, ESP or CoDeepNEAT (Sections 7.1.1

and 10.3.2). In such a process, the neuron that correlates most significantly with high fitness is selected first. When selecting the next neuron, a measure of epistasis (i.e. dependence) is first used to decide whether the fitness correlations of the next neuron candidates should be calculated based on only those networks that contain the previous neuron, or all networks in the population. The neuron with the highest correlation is then chosen as the next neuron. In this manner, a single offspring is constructed at a time in a probabilistic process that does not employ crossover or mutation. In problems such as double pole balancing, this approach, called Eugenic neuroevolution, can find solutions several times faster and more reliably than methods that evolve partial solutions without it (Polani and Miikkulainen 2000; Alden, Kesteren, and Miikkulainen 2002). Note that diversity in the population is crucial to form a good model–and the model is a good way to take advantage of such diversity.

So far the idea of utilizing culture has relied on the current population only. But culture can extend over multiple generations, and there is no reason why populations from prior generations couldn't be utilized in evolutionary algorithms as well. The more solutions there are to define culture, the more diversity there is also likely to be, making cultural algoritms more effective. Of course, an efficient way to store the solutions and select parents among them is needed.

Neuroannealing (Lockett and Miikkulainen 2013) provides such a mechanism. All solutions ever encountered in the evolutionary run are organized into a partition tree of solutions. There are four levels: the first one is partitioned according to number of layers in the network, the second according to the number of nodes in each layer, the third according to connectivity patterns between layers, and the fourth according to the weight values. A parent is selected by traversing the tree using a Boltzmann distribution on average fitness of each branch, as in simulated annealing. Once a parent is selected, NEAT-like mutations are performed to generate new solutions based on it.

Compared to standard NEAT, the neuroannealing process provides more ways to increase complexity without forgetting any of the previous solutions. It can thus construct larger and deeper networks than NEAT. Such networks are particularly useful in particular in fractored domains such as those that require evolving a behavioral strategy (Section 6.1.4. It outperforms NEAT in many such problems, including multiplexer design, concentric spirals, and double pole balancing.

Neuroannealing can be seen as implementing an extreme form of elitism: any solution can have useful information in it, and therefore nothing is ever discarded. Thus the population grows big over time, and is likely to include more diversity in solutions than smaller and constant-size populations can. With all this information, it is possible to reprsent the fitness function more comprehensively.

Each of the methods reviewed in this section point out opportunities for utilizing diversity in population culture in neuroevolution. An interesting challenge for the future is to find synergies between them: for instance, neuroannealing could be combined with eugenic evolution to build better models; culling, mate selection, and intelligent discarding with any generation-based methods; egalitarian learning with eugenic or neuroannealing systems. In this manner, diversity can be utilized in many more ways than simply powering search based on crossover.

5.8 Chapter Review Questions

1. **Biological and Computational Diversity:** Explain why diversity is a cornerstone of both biological evolution and computational neuroevolution. How does diversity enable complex solutions to emerge over time and adapt to changing environments?
2. **Genetic Diversity:** What role does genetic diversity play in evolutionary computation? Discuss the problems that arise when a population converges too quickly and how these issues hinder recombination and exploration.
3. **Behavioral Diversity:** Why is behavioral diversity particularly important in neuroevolution? Contrast it with genetic diversity, and describe a scenario where behavioral diversity could improve the search process.
4. **Diversity Maintenance Techniques:** Compare and contrast two methods for maintaining genetic diversity: fitness sharing and crowding. How do these techniques work, and what are their limitations?
5. **Behavioral Characterizations:** What is a behavior characterization (BC), and why is it essential for measuring and promoting behavioral diversity? Provide an example of how a BC could be defined in a robot navigation task.
6. **Multiobjectivity:** Explain how multiobjective optimization fosters diversity in neuroevolution. What are the benefits of having a Pareto front, and how does it relate to boosting population diversity?
7. **Ensembling:** Why is ensembling particularly well-suited for evolutionary algorithms? Describe how the NEAT method uses speciation to facilitate ensembling, and provide an example of its application.
8. **Cultural Diversity:** What is the role of population culture in neuroevolution? How can cultural mechanisms, such as culling, mate selection, discarding, and training, improve the efficiency and outcomes of evolutionary processes?
9. **Egalitarian Learning:** Define egalitarian social learning in the context of neuroevolution. How does it differ from a student-teacher approach, and why does it enhance diversity in a population?
10. **Neuroannealing:** How does neuroannealing utilize solutions from previous generations to enhance evolutionary outcomes? Discuss how its hierarchical storage mechanism supports diversity and facilitates complex problem-solving.