5 Searching for / utilizing diversity

A most remarkable outcome of biological evolution is the tremendous diversity of solutions it has produced. There is life in a large variety of environments: organisms thrive in extreme heat, and cold, thin atmosphere and deep ocean pressure, large and small scales, based on a variety of energy sources and chemical building blocks. The mechanisms that produce such diversity make it possible to both construct complex solutions over time and to adapt to the changing world. As a matter of fact, a new challenge can often be met by small modifications to already existing solutions, leading to the observation that evolution is a tinkerer (Jacob 1977).

The same is true of computational evolution: generating and maintaining diversity makes it possible to solve harder problems. Diversity does not arise naturally in most evolutionary methods but requires special mechanisms. Such methods usually focus on genetic diversity; however, with neuroevolution, behavioral diversity has an important role as well. This perspective leads to methods of balancing performance and diversity objectives, as will be discussed in this chapter.

5.1 Genetic Diversity

Evolutionary computation is often formalized as a process of finding an optimum in a fitness landscape. The process starts with an initial population that is widespread on the landscape and gradually converges around the highest peaks in it. In this sense, loss of diversity is an essential part of the process: It allows allocating search resources where they matter the most, eventually refining the solutions so that the best ones can be found reliably and accurately.

However, the process may sometimes converge too soon, before all the promising peak areas have been discovered. Some of the best solutions may have narrow basins, and may thus be missed. Such premature convergence is difficult to detect and guard against. Also, if the problem is dynamic, i.e. the fitness landscape changes over time, the converged population cannot keep up. Once the population has converged, there's little hope of finding anything better, or anything new.

The reason is that the most powerful and unique mechanism of evolutionary search, recombination, no longer works in a converged population. If all solutions are similar, recombining them generates nothing new, and progress stops. Mutation still remains, and can in principle create new material. However without an effective crossover, the process is essentially reduced to random search.

Thus, most evolutionary computation methods today are in direct conflict with diversity. The methods aim at making progress in a greedy manner, with a strong selection that converges quickly. As will be discussed in Section 9.1.1, this is not the case in biological evolution. The selection is weak, many genetic changes are neutral and remain in the population for a long time. Slowing down the process in this manner may result in more diversity and creativity. This is also an option in evolutionary computation, but it has not yet been fully explored. Taking advantage of weak selection, neutrality, and deep time is an interesting direction for the future.

The simplest approach to coping with premature convergence is to increase the mutation rate. If it is done early enough, it may give crossover enough material to operate. However, this material is essentially random, and at large enough levels will undermine evolutionary search. Another straightforward approach is to extend the current population with an archive of representative past individuals. The archive ensures that diversity is not lost, but it is infeasible to grow the archive indefinitely, and it is difficult to decide which individuals should be included in it.

Thus, evolutionary computation mostly relies on mechanisms that are added to search for the purpose of maintaining diversity. The first challenge in building such mechanisms is to measure diversity. At the level of genetic encodings, it is often possible through a distance metric between genomes. They are often vectors of values, so Euclidean distance (L1) is often sufficient. Manhattan distance (L1), Hamming distance, or edit distance may also work in various cases. With such a distance metric, diversity can be measured as the average distance between genomes in the population.

Diversity measures can be further focused on a local area of the space, or k nearest neighbors. Such an approach is useful in case it is important to identify which individuals in the population contribute to diversity more than others—those individuals can then be kept in the population or the archive longer.

Several methods have been developed to take advantage of these measures. In crowding (De Jong 1975), new individuals are allowed to replace existing individuals that are similar to them, or their parents. Note that this mechanism does not drive the creation of diversity, but slows down convergence: it is not as easy for similar individuals to take over the population.

In fitness sharing (Goldberg and Richardson 1987), the actual fitness of an individual is adjusted based on how similar it is to other individuals in the population. More specifically, the fitness $f(x)$ of individual x is adjusted by

$$f'(x) = \frac{f(x)}{s(x)}. \tag{5.22}$$

The similarity metric s is e.g.

$$s(x) = \sum_{j=1}^{n} d(x, y_j), \tag{5.23}$$

where the distance $d(x, y_j)$ is taken over all n members y_j of the population. In this manner, the fitness is reduced for individuals that are similar to many other individuals in the population. The adjustment makes them less likely to be chosen as parents and more likely to be discarded, thus slowing down convergence. The similarity metric is expensive to calculate.

It can be made more practical by reducing the calculation to a local neighborhood, or to a sampling of the population.

Fitness sharing in some domains can be implemented implicitly, avoiding the extra computation. In particular in cooperative coevolution, solutions are constructed by combining individual population members into a single structure, such as a neural network composed of several neurons (Potter and Jong 2000; Moriarty and Miikkulainen 1997). The entire solution is evaluated for fitness; the individual's fitness is the average fitness of all solutions in which it participated. It turns out that good solutions are usually composed of diverse individuals. If for instance a neural network is put together from a single neuron cloned many times, it would likely not perform well. Thus, evolution in cooperative coevolution populations maintains diversity as part of the evolution process itself. If one kind of neuron starts taking over the population, it will be selected too many times for the network, the network performs poorly, the neuron receives lower fitness, is likely to be discarded, and diversity returns. Thus, by making diversity implicitly part of the fitness evaluation, it can be maintained automatically.

Further, when evolving neural networks, genetic diversity is often less important than the diversity of the behavior the networks generate. This perspective will be discussed next.

5.2 Behavioral Diversity

It is important to maintain genetic diversity in evolution so that the search process can cover enough of the search space to find good solutions, and can adapt to any changes in the landscape. This goal is important in neuroevolution as well, and genetic diversity maintenance methods are useful in it. However, neuroevolution is different from many other types of evolutionary optimization in that it aims to construct computational structures, i.e. neural networks, rather than static solutions. It is important that the behaviors of those networks are diverse as well. In many such domains the fitness landscapes are deceptive, i.e. the highest peaks are surrounded by valleys, or they are flat, i.e. many different behaviors lead to similar fitness. Methods that rely on hill-climbing, i.e. incremental improvement through small changes, such as reinforcement learning and mutation-based search, struggle in such domains. They are difficult for neuroevolution as well, but search based on behavioral diversity makes it more effective.

Creating and maintaining genetic diversity does not necessarily lead to diverse behaviors. The reason is that the mapping between the genotype and behavior is complex and unpredictable. First, the same behavior can be encoded by very different neural networks. One example of this phenomenon is competing conventions: The same neurons and weights in the network are encoded in a different order in the genome. As a result, the networks function exactly the same, but the encodings have no overlap, i.e. are maximally diverse. Second, a small change in the encoding can have a large effect on the behavior. Negating an activation function, for example, may cause the robot to turn left instead of right. Genetic diversity is thus not a good indicator of behavioral diversity.

Evolution of behaviors still takes place at the level of encodings, of course, and the genetic diversity needs to be maintained to prevent convergence. However the mechanisms for measuring, maintaining, and creating behavioral diversity are quite different, resulting in fundamentally different evolutionary processes.

Whereas genetic diversity could be measured in a relatively straightforward manner based on the distance between encodings, behavioral diversity is more complex. First, behavior needs to be characterized formally, taking into account what matters in the domain. This often involves creating a vector representation of the behavior, or a behavior characterization (BC; Mouret and Stéphane Doncieux 2012). For instance for a mobile robot, the BC could consist of a histogram of the sensory inputs, actions, and locations encountered during a number of sample runs. More generally, a collection of possible inputs to the network could be created, and the outputs corresponding to each of these inputs taken as the BC. If domain knowledge is not available, they can be generated randomly. With domain knowledge, it may be possible to define a collection of situations that forms a representative sample, or better yet, a sample of the most important decision points in the domain, thus creating a more meaningful BC (Mouret and Stéphane Doncieux 2012; Gomes, Urbano, and Christensen 2013).

It is difficult to form such a BC for recurrent neural networks where not only the current inputs matter, but also the history of the preceding inputs and actions. A common approach is to represent the actions as distributions, and the BC as a mapping: for a set of sensory states, it specifies the distribution of actions the agent is likely to take. Interestingly, with such a representation, it is possible to learn optimal BCs (Meyerson, Lehman, and Miikkulainen 2016a) for a set of multiple tasks in the same domain, such as robot navigation in multiple mazes. The BCs are adapted so that they represent the distributions of optimally behaving agents in known tasks, forming a powerful foundation for evolution of optimal behavior in new tasks.

Once a BC has been defined, the next step is to measure diversity among them. As in the case of genetic diversity, calculating the average distance between individuals is a common approach. A more formal way is to utilize entropy, an information-theoretic concept that measures the level of surprise or uncertainty in the outcomes of a random variable. Intelligent behavior in general can be described as resulting from entropy maximization (Wissner-Gross and Freer 2013). In evolutionary computation, it can be applied to the behavior of an agent or a population of agents, thus describing how diverse they are. For instance, the behavioral space can be divided into discrete intervals, and the number of agents visiting each interval counted (Kang et al. 2021). The entropy of this distribution then measures the behavioral diversity of the population.

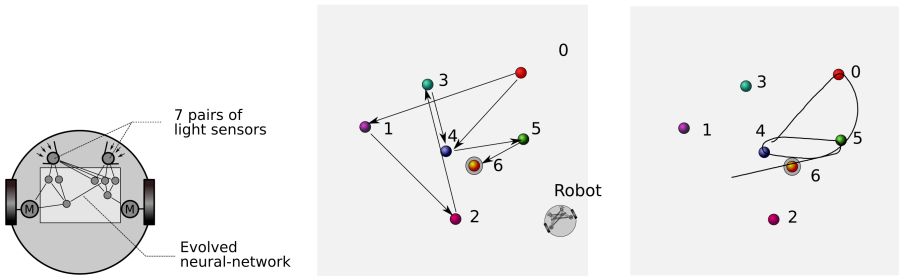
The information-theoretic approach can be developed further to measure empowerment, i.e. the ability of an agent to control its world (Salge, Glackin, and Polani 2014a). Empowerment can be defined as the channel capacity between the agent's actuators A_t at time t and its sensors S_{t+1} at the next time step:

$$E = \max_{p(a_t)} I(S_{t+1}; A_t), \tag{5.24}$$

where $p(a_t)$ is the probability of actuator value a_t at time t and $I(S;A)$ is the mutual information between S and A, i.e.

$$I(S;A) = H(A) - H(A|S) = H(S) - H(S|A), \tag{5.25}$$

where $H(X)$ is the entropy of X. The $I(S;A)$ thus measures how much of the state entropy measure above can be explained by actions. The resulting metric, channel capacity, stands for the maximum rate of information transmission from A to S. In essence, empowerment



(*a*) Controller (*b*) Full light sequence (*c*) Discovered sequence

Figure 5.1: **Using behavioral diversity to discover solutions in a domain with a deceptive or flat fitness function**. The robot (*a*) has to move to the lights in the order indicated by the arrows (*b*) to eventually turn on Light 6. Fitness is defined as the number of time steps to reach Light 6 and therefore does not indicate which behaviors are promising early on. In contrast, behavioral diversity rewards controllers that turn on more and more lights; thus, it encourages exploration that eventually makes the search successful (*c*). In this manner, behavioral diversity can be used to guide search even when the fitness function is flat (as in this case) or deceptive (more generally). (Figures from Mouret and Stephane Doncieux 2009)

E thus measures the causal influence of the agent's actions on its future sensory inputs, i.e. how much power the agent has in changing the world it perceives. Empowerment is a useful concept in many ways. It is possible to characterize the evolution of intelligent agents as a process that maximizes empowerment. Similarly, the evolved agents then behave in order to maximize their empowerment. Such behavior provides the agents an intrinsic motivation that results in various goal-oriented behaviors.

Empowerment is thus a general theory of evolution of intelligent behavior. It measures a general desirable quality of an evolved agent and can be used as an explicit evolutionary objective. While it does not measure diversity directly, it often correlates with it. Similarly to implicit fitness sharing described in the previous section, empowerment favors actions that have a large impact, regardless of other objectives. In that sense, it often serves to diversify the set of actions that are available for the agents, and thereby leads to diverse behaviors.

As an example of behavioral diversity at work, consider a task for an evolutionary robot that moves around in an environment where seven lights are on or off in fixed locations (Figure 5.1; Mouret and Stephane Doncieux 2009). The robot can sense each light, and it can move around by controlling its two wheels. When it steps on a light, one or two other lights turn on. The task is to discover how to turn on Light 6. In the beginning, only Light 0 is on. To turn on Light 6, it has to first go to Light 0, then to 4, 5, and 6; or else, go to lights 0, 1, 3, 4, 5, and 6. Fitness is defined as the number of time steps to reach Light 6; thus, unless the robot is successful, it receives no fitness, and no indication of whether its behavior is promising. It is therefore very difficult to discover successful behavior based on fitness only. Therefore, evolutionary search for the optimal behavior does not even get started.

However, it is possible to define BC as the collection of lights that are on, such as 1000000, 1001000, 1100000, and so on. An archive of discovered behaviors can then be formed, and evolution rewarded for exploring new behaviors. In this manner, evolution quickly discovers movement sequences that result in more lights being turned on, including eventually Light 6. Thus, behavior diversity makes search effective in this domain where the fitness function does not provide a hill to climb. In the same manner, behavioral diversity helps cope with fitness functions that are deceptive, i.e. fitness peaks are located behind fitness valleys.

This section has introduced and illustrated the fundamentals of behavioral diversity. The next two subsections push these concepts further in opposite directions: novelty search aims to maximize exploration and creativity through divergent search, and quality-diversity methods seek to combine diversity with performance objectives.

5.3 Novelty Search

The previous sections have shown how evolution with behavioral diversity objectives can discover solutions that are difficult to find. It is possible to take this approach one step further, and make it the only objective of search. That is, the entire aim of evolution is to keep generating new variation and never converge at all: it is divergent instead of convergent.

A good motivation for divergent evolution comes from biology. Unlike traditional evolutionary computation, biological evolution does not have a goal. Variation is generated continuously, and selection operates upon it. This selection pressure is much weaker than that used in evolutionary computation, and results in much broader diversity. Evolution can thus quickly adapt to new situations, taking advantage of niches that confer an advantage in survival. The results can sometimes seem extremely creative, like the anglerfish, which lures prey by generating light at the end of a long fin ray (Coleman 2019), or bacteria that evolve to utilize citric acid as their carbon source (Blount, Borland, and Lenski 2008). It is this kind of creativity that computational divergent search is aimed at capturing.

Divergent search can be formalized within the current EC framework simply by rewarding behavioral diversity instead of performance. This approach is called novelty search (Stanley and Lehman 2015; Lehman and Kenneth O. Stanley 2010). A novelty metric is defined that measures how different a candidate solution is from solutions that have been generated before, i.e. how novel it is. This novelty metric then replaces the usual fitness metrics that measure performance in a task.

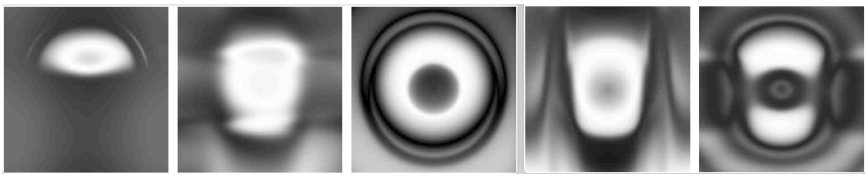
A common novelty metric is the sparseness of the behavior space around the individual, i.e. the average distance to its k nearest neighbors. Similarly to Equation 5.23,

$$\rho(x) = 1/k \sum_{j=1}^{k} d(x, y_j), \tag{5.26}$$

where $\rho(x)$ stands for the novelty of individual x, y_j is the jth nearest neighbor of x, and d is the distance metric between their behavioral characterizations. This novelty is computed against the current population as well as an archive of prior solutions. The archive is first initialized randomly, and new individuals are then added to it with a low probability. In this manner, the archive constitutes a sampling of the behavior space, guiding the search to new areas.

Gen 12 Gen 20 Gen 36 Gen49 Gen 74

(*a*) Intermediate images in the evolution of the skull image



Run 1 Run 3 Run 7 Run 15 Run 17

(*b*) Attempts to evolve the skull image directly

Figure 5.2: **Stepping-Stone-Based vs. Direct Evolution of a Skull Image.** How can a a CPPN be evolved to create a particular image, such as the skull? (*a*) Human players of Picbreeder selected images that looked interesting on their own, without the goal of generating a skull, which emerged serendipitously toward the end of the evolution from these stepping stones. (*b*) When evolution is directed to evolve the skull image directly with a distance-based fitness, it falls short of most of the details; shown are the final results of five such example runs. In this sense, the discovery of stepping stones is crucial in generating complex solutions. (Figures from Woolley and Stanley 2011)

Novelty search indeed leads to diverse solutions. However, and most remarkably, it sometimes also discovers solutions that are useful in the domain–even though there is no provision in the search for preferring them in any way. One potential explanation is that in order to be most different from what has been created before, it is a good idea to utilize structure in the domain. That is, search may discover stepping stones that can be combined effectively into more complex solutions, thus creating more diversity than a random search.

The motivation for this idea comes from the Picbreeder game (Section 8.3), where human players select the most interesting images and evolution creates more images by crossing over and mutating the CPPN neural networks that generated them (Secretan et al. 2011b). It turns out that the human players do not usually have a goal in mind in what they are trying to generate, but instead, use the raw material serendipitously: They come up with ideas of what to create on the fly, depending on what interesting shapes and images are currently in the population. For instance in creating a skull image, they utilized, over time, many images that looked nothing like the skull. There were images that could be described as a a crescent moon, a comet, a drop on water, and a mask ((Figure 5.2a; Woolley and Stanley 2011). These images served as stepping stones that eventually came together to generate the skull.

Interestingly, if evolution is set up with the goal of generating the skull image, it fails (Figure 5.2b). The images approach the skull shape overall but never get the elements right.

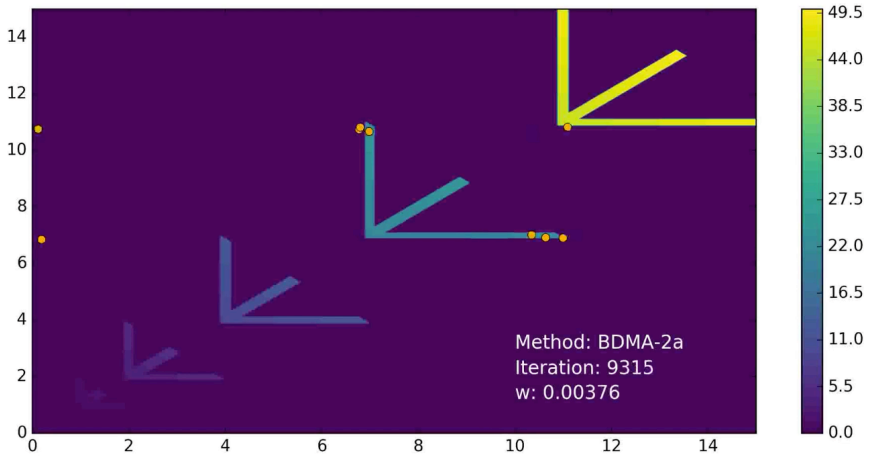
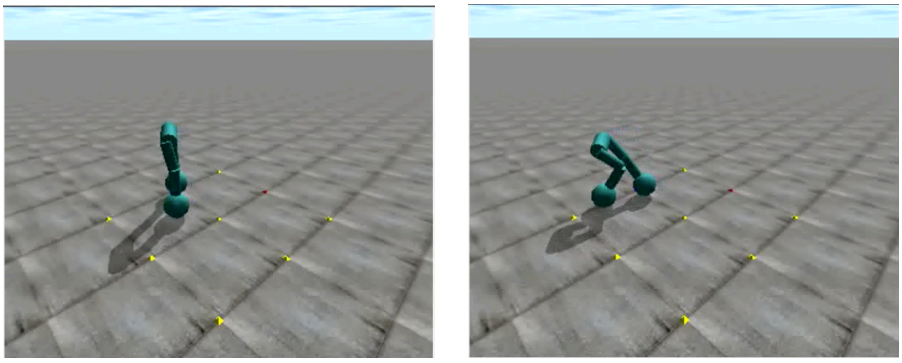


Figure 5.3: **Illustration of search based on stepping stones.** In this experiment, a population of points are evolved on the 2D rectangle. Fitness is zero in the background, and increases in each claw from left to right and from bottom to top. The population starts at the bottom left and has to discover the top fitness at the top right. While fitness-based and novelty-based searches are not much better than random, a search method that discovers and utilizes stepping stones performs much better. It discovers the local optima at the end of each finger of the claw-like pattern, and then combines them to make the jump to the next claw. In this manner, stepping stones can be identified as local optima, and recombined to make discoveries that would otherwise be difficult to make. For an animation, see https://neuroevolutionbook.com/neuroevolution-demos. (Figure from Meyerson and Miikkulainen 2017)

Perhaps the evolution of something that complex relies on discovering the proper stepping stones, i.e. discovering the solutions that represent the prominent structure in the domain?

One way to characterize the stepping stones is that they are local maxima in the search space wrt. a metric different from novelty. This metric could measure how impressive they are (Lehman and Kenneth O. Stanley 2012), or it could be related to performance in the domain (Meyerson and Miikkulainen 2017). Stepping stones can then be identified as those solutions that dominate other solutions in terms of novelty and fitness. In this manner, the search discovers global novelty and local refinement. For instance in the domain of Figure 5.3, neither novelty-based nor fitness-based search is much better than random search in finding the high fitness region on top right. However, the claw-like areas form stepping stones: The fitness increases horizontally and vertically in each toe, and by combining the end solutions of each toe, it is possible to jump to the next claw (with superior fitness). A search mechanism that takes advantage of local fitness and global novelty can utilize such stepping stones, and discover useful solutions in the domain.

Stepping stones can be found in complex real-world domains as well (Stanley and Lehman 2015; Lehman and Kenneth O. Stanley 2010). For instance, consider evolving a controller network for a bipedal simulated robot (Figure 5.4). It is possible to reward the networks simply by the distance the walker can travel before falling over. Such evolution is rewarded by incremental progress, and results in movement that is limited and aims to be



(*a*) Fitness-based search (*b*) Novelty search

Figure 5.4: **Contrasting the creativity of solutions in convergent and divergent search.** Gaits for the bipedal walker are evolved in two ways. (*a*) Convergent (fitness-based) evolution favors small, safe improvements that allow the robot to travel incrementally further. The resulting gait is rigid and slow and often fails. (*b*) In contrast, divergent (novelty-based) evolution discovers dynamic behaviors such as falls and jumps that are different from others. They serve as stepping stones in exploring a larger space which eventually includes robust dynamic gaits. In this manner, superior solutions can be discovered even when (and because!) they are not directly rewarded. For animations, see https://neuroevolutionbook.com/neuroevolution-demos.

stable, but is also vulnerable to disturbances and variations that might occur in the environment. In contrast, when such walking is evolved through novelty search, many behaviors that have little to do with walking are discovered, such as falling flat, jumping forward, taking a few steps before falling, and ultimately, leaning forward and moving legs fast to prevent falling. It turns out that such walking is more robust and more effective. It emerged from many different kinds of failures, and avoids them effectively. Evolution utilizes these failures as stepping stones, combining them effectively into more comprehensive solutions.

Quality diversity methods can be seen as a way to take advantage of stepping stones in a more general framework. The idea is to combine novelty search with fitness-based search in a way that allows finding better solutions and finding them faster, presumably taking advantage of stepping stones along the way. Quality diversity methods will be discussed in the next section.

5.4 Quality Diversity Methods

Quality Diversity (QD) (Pugh, Soros, and Stanley 2016) represents a significant shift in evolutionary computation, focusing not on finding the single best solution but on discovering a wide array of viable solutions, akin to the diverse species found in nature. This paradigm prioritizes the exploration of a rich behavior space, seeking a collection of individuals that are not only diverse but also exhibit high performance in their respective niches. The core idea of QD is to balance the quest for diversity with the search for quality, mirroring the natural competition seen within biological niches. Diversity in QD is evaluated based on the behavior of individuals throughout their lifetimes, with the aim of covering all regions of the

behavior space. Unlike traditional optimization approaches that focus on high-performing areas, QD aspires to sample the entire behavior space, striving for the best performance within each region. Its broader implications extend into areas like computational creativity and open-ended evolution, where the goal is to continually generate uniquely interesting artifacts. This approach represents a departure from the convergent, objective-driven processes that dominate traditional machine learning and optimization techniques, offering a new perspective on exploring and exploiting the vast potential within diverse solution spaces.

Two early realizations of the QD paradigm are Novelty Search with Local Competition (NSLC) (Lehman and Kenneth O Stanley 2011) and Multi-dimensional Archive Phenotypic Elites (MAP-Elites) (Mouret and Clune 2015). These algorithms embody the QD approach by combining the drive for behavioral diversity with a localized search for performance quality. NSLC and MAP-Elites have demonstrated that this focus on diversification, rather than pure optimization, can yield impressive results in various domains, including those where traditional optimization methods fall short.

5.4.1 Novelty Search with Local Competition

In this seminal work, Lehman and Kenneth O Stanley (2011) tackle the complex challenge of generating diverse, capable artificial creatures within simulated environments, as illustrated in Figure 5.5. A crucial factor in addressing this issue is the representation of virtual creatures. Although there is a strong incentive to develop richer and more tractable representations, the authors argue that "the full potential of even the best representation may remain unachieved if evolution is handicapped by a deficient system of rewards." Independent of the chosen representation, traditional evolutionary algorithms often prematurely converge on a limited set of solutions, which restricts diversity and adaptability, leading to ecological impoverishment and limited speciation. Novelty search has been proposed as a remedy, rewarding divergence from past designs to enhance ecological diversity. However, focusing solely on novel morphologies can lead to functionally impractical designs, indicating the necessity of balancing the pursuit of morphological novelty with functionality to ensure that evolved creatures are not only diverse but also capable of effective performance within their environments.

To address this problem, the authors propose to combine novelty search with mechanisms of local competition (NSLC) that is motivated by the biological principle that ecosystems thrive on diversity and ensures resilience and adaptability. Novelty search, rewarding uniqueness rather than just fitness to a task, effectively prevents convergence on premature solutions. Local competition, simulating a more natural selection environment where creatures compete against others in their immediate vicinity rather than against a global fitness standard, promotes performance localized within morphological niches. Such a dual approach leads to high diversity while also maintaining the functional capabilities of the creatures.

The authors implement their approach using a genetic algorithm where each individual in the population is assessed both for its novelty and its competitive ability. Novelty is measured based on a multi-dimensional feature descriptor that quantifies how different an individual is from the rest of the population and from those stored in an archive

of historically novel individuals. The local competition is implemented by having individuals compete for survival against a subset of the population within their niche (e.g., competition among the k nearest neighbors, where k is a fixed parameter that is determined experimentally), rather than the entire population. The genetic representation of the creatures includes both morphological traits (such as body size and shape), allowing for a rich variety of possible evolutions.

NSLC leads to several beneficial effects. First, the ecosystem of evolved creatures shows a much higher level of diversity compared to systems evolved with traditional fitness-only approaches, as is illustrated in Figure 5.5. This diversity is not just superficial; the creatures exhibit a wide range of functional behaviors and physical forms. Secondly, the local competition model ensures that while diversity is maintained, the creatures also develop practical abilities to survive and thrive in their environments. This method effectively balances the exploration of the genotype space (through novelty search) with the exploitation of successful strategies (through local competition).

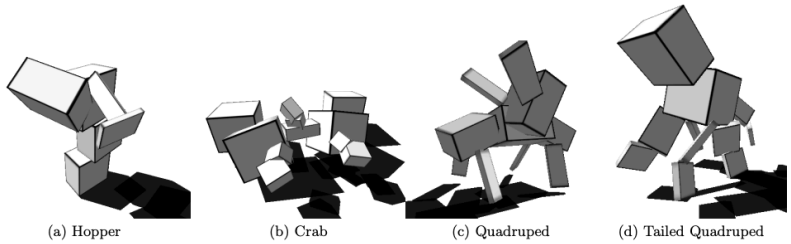


Figure 5.5: **Diverse competent morphologies discovered within a typical single run of local competition.** Various creatures are shown that have specialized to effectively exploit particular niches of morphology space. These creatures were all found in the final population of a typical run of local competition. The hopper (a) is a unipedal hopper that is very tall, (b) is a heavy short crab-like creature, and (c) and (d) are distinct quadrupeds. Creature (c) drives a large protrusion on its back to generate momentum, and (d) has a tail for balance. (Figure from Lehman and Kenneth O Stanley 2011)

5.4.2 MAP-Elites

MAP-Elites distinguishes itself within the QD domain by explicitly defining niches, a stark contrast to the passive emergence seen in NSCL. MAP-Elites operates by partitioning the search space into a grid of niches, each defined by specific feature dimensions that describe meaningful characteristics of possible solutions. These characteristics are also known as behavior characterization (BC). Each cell in this grid represents a niche corresponding to a unique combination of these features. The algorithm iteratively generates new candidates and places them into the appropriate niche based on their BCs. If a niche is empty, the candidate automatically becomes the elite of that niche. If the niche already has an elite, the new candidate replaces the existing one only if it has higher fitness. This process ensures that

each niche retains the best solution found so far according to the fitness function, but crucially, also captures a diverse array of solutions across the entire range of defined features. Figure 5.6 gives the pseudo-code of MAP-Elites.

procedure MAP-ELITES ALGORITHM (SIMPLE, DEFAULT VERSION)
  $(\mathcal{P} \leftarrow \emptyset, \mathcal{X} \leftarrow \emptyset)$ ▷ *Create an empty, N-dimensional map of elites: {solutions $\mathcal{X}$ and their performances $\mathcal{P}$}*
  for iter $= 1 \rightarrow I$ **do** ▷ *Repeat for I iterations.*
    if iter $< G$ **then** ▷ *Initialize by generating G random solutions*
      $\mathbf{x}' \leftarrow$ random_solution()
    else ▷ *All subsequent solutions are generated from elites in the map*
      $\mathbf{x} \leftarrow$ random_selection($\mathcal{X}$) ▷ *Randomly select an elite x from the map $\mathcal{X}$*
      $\mathbf{x}' \leftarrow$ random_variation($\mathbf{x}$) ▷ *Create x', a randomly modified copy of x (via mutation and/or crossover)*
    $\mathbf{b}' \leftarrow$feature_descriptor($\mathbf{x}'$) ▷ *Simulate the candidate solution x' and record its feature descriptor $\mathbf{b}'$*
    $p' \leftarrow$performance($\mathbf{x}'$) ▷ *Record the performance p' of x'*
    if $\mathcal{P}(\mathbf{b}') = \emptyset$ or $\mathcal{P}(\mathbf{b}') < p'$ **then** ▷ *If the appropriate cell is empty or its occupants's performance is $\leq p'$, then*
      $\mathcal{P}(\mathbf{b}') \leftarrow p'$ ▷ *store the performance of x' in the map of elites according to its feature descriptor $\mathbf{b}'$*
      $\mathcal{X}(\mathbf{b}') \leftarrow \mathbf{x}'$ ▷ *store the solution x' in the map of elites according to its feature descriptor $\mathbf{b}'$*
  return feature-performance map ($\mathcal{P}$ and $\mathcal{X}$)

Figure 5.6: A pseudocode description of the simple, default version of MAP-Elites. (Figure from Mouret and Clune 2015)

The effects of applying MAP-Elites are profound and multi-faceted: First, MAP-Elites preserves a diverse set of solutions, each excelling in different parts of the feature space. For example, in its original paper, MAP-Elites managed to evolve various virtual creatures, each representing the pinnacle of their respective behavior niche (refer to the surrounding illustrations in Figure 5.7). This diversity is crucial for robustness and adaptability, as it provides a spectrum of potential solutions to unforeseen challenges or changes in task requirements. Second, MAP-Elites effectively "illuminates" the search space (see the heat map at the center in Figure 5.7), providing insights into how different features of solutions contribute to their success and interrelate with each other. This is particularly valuable in complex domains where the relationship between features and performance is not well understood.

5.4.3 Nuts and Bolts of QD Implementation

Since the establishment of QD as a powerful concept, exemplified by algorithms such as NSLC and MAP-Elites, numerous studies have emerged to analyze and enhance various facets of QD. A selected set of works are introduced below to showcase the intricacies implementing QD from three main perspectives:

Behavior Characterization BC not only determines the form of diversity during the search process but also significantly influences the efficacy of the optimization algorithm. Therefore, it should be meticulously chosen to enhance the QD's performance Pugh, Soros, and Stanley 2016. While there is complete freedom in determining BC for a QD task, it is preferable and necessary to choose those closely related to the desired objective. This approach provides additional benefits, such as improved model interpretability, and is crucial for achieving reasonable performance. For instance, Pugh, Soros, and Stanley (2016) examined the impact of using BCs that are both highly aligned (e.g., final coordinates at the trial's end) and misaligned (e.g., the most frequent direction of orientation) with the quality metric (e.g., goal achievement) in solving maze navigation tasks through various QD implementations. Their findings indicate that BCs misaligned with the quality metric not only underperform but also fail to match the efficacy of pure optimization-based

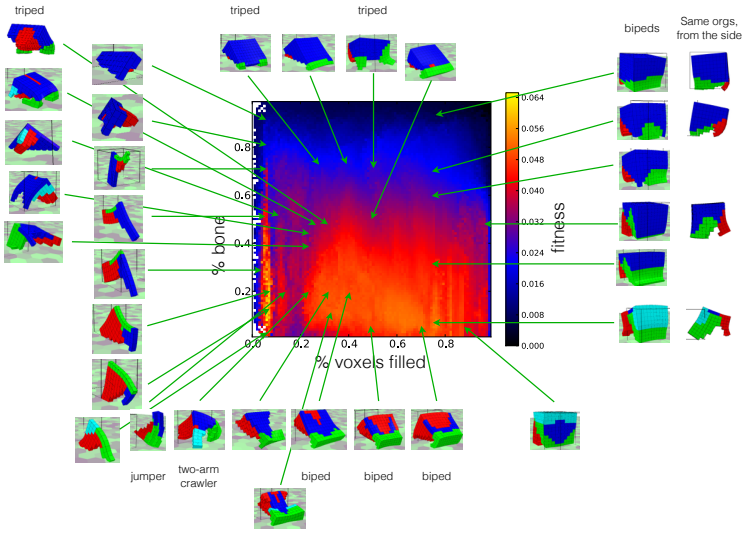


Figure 5.7: Example maps annotated with example organisms from different areas of the feature space. Within a map, MAP-Elites smoothly adapts a design theme along the desired dimensions of variation. Between maps, one can see that there is some variation between maps, both in the performance discovered at specific points, and in the types of solutions discovered. That said, in general each map generally paints the same overall picture of the performance capabilities of each region of the feature space. Note the different scale of the bottom color map. (Figure from Mouret and Clune 2015)

methods. Conversely, BCs aligned with the task's objectives enhance performance, achieving state-of-the-art results at the time. Even when paired with misaligned BCs, the overall performance still surpasses pure fitness searching methods. The key takeaway from their study is that BCs aligned with the quality concept are essential to overcome deception in challenging problems. However, crafting BCs manually requires domain knowledge of the problem and the solution. For problems with limited information, one approach is to use a pure fitness searching method as a baseline, then iteratively incorporate and test candidate BCs for alignment with the quality metric, based on performance improvement over the baseline. Recent studies also suggest the feasibility of learning BC. For instance, meta-learning has been employed to discover optimal BD definitions, enhancing success rates in multiple tasks Meyerson, Lehman, and Miikkulainen 2016b. In robotic locomotion tasks, AURORA Grillotti and Cully 2022 uses dimension reduction models like PCA and Auto-Encoders to encode a robot's sensory data, treating the encoded vector as the BC during learning. These methods have shown promising results and point toward a more generalized approach for BC design.

Niches Representation After establishing BCs, the subsequent task is to develop a technique for segmenting solutions into niches based on these BCs. The approach to niche representation notably differentiates NSLC from MAP-Elites. In NSLC, niches emerge dynamically, defined by the k-nearest neighbors among a generation's peers and the elites in the archive. This results in an evolving archive, where the number and specifics of the 'cells'

are neither predetermined nor known in advance. Conversely, MAP-Elites divides the BC space into discrete behavioral cells. This division is based on BC range and user-defined granularity, offering a complete overview of the archive's size and cell characteristics. However, this method grapples with the curse of dimensionality, as the cell count escalates exponentially with the increase in BCs and their granularity. To mitigate this issue, Vassiliades, Chatzilygeroudis, and Mouret (2017) introduced Centroidal Voronoi Tessellation MAP-Elites (CVT-MAP-Elites). This variant employs a clustering approach like k-means to segment the archive space into k Voronoi tessellations. While CVT-MAP-Elites shares core functionalities with MAP-Elites, it diverges in two key operations: archive definition and cell querying. For defining the archive, CVT-MAP-Elites deploys $K \gg k$ vectors in the BC space to identify k centroids representing the cells, unlike MAP-Elites' straightforward discretization of BCs. When querying a cell to store a phenotype, CVT-MAP-Elites requires checking distances to centroids, potentially increasing computational complexity to $O(k)$ in the worst case, compared to the $O(1)$ complexity in MAP-Elites. Despite this increase in computational load, CVT-MAP-Elites proves advantageous, capable of scaling up to 1000 dimensions in maze experiments, a significant leap from MAP-Elites' limitation to around 20 dimensions.

Optimization Algorithm Although NSLC and MAP-Elites have shown impressive results, their most successful applications have predominantly been in robotic locomotion tasks with simple, low-dimensional controllers Colas et al. 2020. In addition, both QD implementations commonly employ a mutation-based Genetic Algorithm (GA) as their foundational optimization algorithm, leaving the potential of Evolutionary Strategies (ES) family members largely unexplored. Consequently, investigating new optimization methods to achieve scalability and enhance learning efficiency is a logical next step. In this context, Colas et al. (2020) introduced MAP-Elites with Evolution Strategies (ME-ES), utilizing the efficiency of ES to extend MAP-Elites to high-dimensional controllers managed by large neural networks. ME-ES demonstrated the ability to learn a neural network controller with approximately 10^5 parameters—significantly larger than those in previous studies—outperforming GA-based methods even with triple the computation time. Simultaneously, Fontaine et al. (2020) developed Covariance Matrix Adaptation MAP-Elites (CMA-ME), which integrates the high-performing CMA-ES algorithm from the ES family into the QD framework. The authors crafted a fitness function that prioritizes exploration (i.e., populating empty cells) over optimization (i.e., enhancing performance in filled cells) as the primary objective for CMA-ES. When the archive remains unchanged, CMA-ES's initial parameters and internal states are reset using a randomly chosen individual from the archive. In comparative experiments, CMA-ME outperformed MAP-Elites by not only doubling the solution quality but also providing a broader diversity of solutions. Building upon these advancements, Fontaine and Nikolaidis (2021) introduced MAP-Elites via a Gradient Arborescence (CMA-MEGA). Unlike traditional ES methods, which treat objective and BC functions as black boxes, MEGA integrates directional perturbations into MAP-Elites based on gradients of these functions, provided they are first-order differentiable. It employs CMA-ES to optimize the factors within the perturbation function. CMA-MEGA significantly surpasses traditional QD algorithms by not treating objective and BC functions as black boxes, and it demonstrates its efficacy in generating a diverse array of high-quality images by searching the latent space of a StyleGAN. Further building on these innovations,

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