Discovering the preference hypervolume: an interactive model for real world computational co-creativity

Hagg, A.

CHAPTER 3

Divergent Search

One widely-used method to generate diverse solutions is to rely on random or semi-random search mechanisms to generate novel, unconventional ideas. Blind or unbiased variation is often used as part of an evolutionary algorithm (EA) (see Appendix A), which selectively retains perturbed solutions when indications exists that they are useful, as was described by Simonton (2007). Some stage theories of creativity, which were described in Section 2.1.1, contain a recombination step (see Santanen et al. (2004) and Nijstad and Stroebe (2006)) that causes high cognitive load in humans. To find more distant, novel, solutions, evolutionary computation should be used as the driver of variation in creative processes (see Lehman et al. (2020)). EA perform specifically well at finding 'haphazard recombinations' to generate novel solutions. In a recent article, Miikkulainen (2020) described that he expects evolutionary computation, combined with novel learning techniques, to be the 'next deep learning'. He pinpoints machine creativity as the upcoming milestone in artificial intelligence.

In this chapter I discuss divergent evolutionary optimization techniques that form the cornerstone of the objective participant in the process model introduced in Section 2.4. In the divergent search paradigm, we do not look to find the best solution, but instead find a diverse set of high-performing solutions. Diversity is the result of a process that finds novel solutions and is usually measured and enforced based on distance metrics on a set of descriptive feature dimensions. This chapter discusses solution diversity as treated throughout the history of evolutionary optimization algorithms. Three main paradigms are examined: multiobjective (or multicriterion) optimization, multimodal optimization, and phenotypic niching (also called quality diversity (QD)). The latter connects the concept of ecologic niches in biology to evolutionary optimization. Common diversity metrics are

used to compare these diversity maintenance techniques, whereby understanding the difference between genetic and phenotypic similarity helps us to create and select these divergent algorithms for use in co-creative processes. The solution sets that are created help to overcome the limits of our imagination (see Sartre and Elkaïm-Sartre (1946)), by showing a large variety of solutions.

In this chapter, the following requirements from Tables 2.1–2.4 are fulfilled:

- A3 ("The initial solution set presented to user should need as little guidance from the user as possible")
- A6 ("The process should contain evolutionary components")
- B1 ("Solutions should be categorized")
- B2 ("Knowledge should be organized hierarchically")
- B3 ("User can filter knowledge base dynamically")
- B4 ("Process should provide diverse perspectives on existing knowledge")
- C1 ("Search should efficiently generate a diverse set of novel, functional solutions")
- C3 ("Search should identify regions of design feasibility and optimality")
- C4 ("Search should support addition, removal and/or variation of constraints, objectives and variable parameter bounds")
- C5 ("Solution similarity should be based on their functional expression and presented along meaningful feature dimensions")

In order to achieve this, the following research questions need to be answered:

Research Questions

I Are solutions best compared using their genomes or their expressed phenotype or behavior? (requirements B1, B2, B3, and C5)
II What multi-solution optimization method produces the highest phenotypic diversity? (requirements A1, A6, B4, C1, C3, and C4)
III Can we produce more diverse solution sets when learning phenotypic niching from data instead of using predefined features? (requirements A3, B4, C1)
IV What are the limitations of generative models in terms of the possible diversity of the solutions they create? (requirements A3, B4, C1)

The main insight highlighted in this chapter is that in multi-solution optimization, a focus on providing functional diversity can be achieved by applying diversity maintenance based on artifacts' morphological and behavioral similarity.

What follows now is a connection between research in natural evolution and ecology on the importance of the phenotype, or expression of a genome, to evolutionary encodings. This is taken from work by Hagg (2021).

3.1 Extending the Phenotype

In this and the next section, an answer to research question I ("Are solutions best compared using their genomes or their expressed phenotype or behavior?") is sought. To achieve this, the manner in which solutions are encoded in (evolutionary) computation needs to be discussed. Let us take a step back and analyze what ideas from natural evolution, the origin and inspiration of evolutionary computation, tell us about computational encodings.

The two key features of natural evolution are survivability (quality, fitness, or optimality) and diversity. Quality is mainly driven by naturally occurring scarcity of nourishment, weather and safe environments. Only creatures that are fit enough will survive to create offspring. Mutations in genomes of the offspring of successful creatures cause the population to diverge and explore possible genomes. Diversity arises from the driving forces of random variation and protection by *niching* effects. Evolutionary niches shield species from having to battle for nourishment with all other species. Each species naturally converges to being fit enough within a niche, becoming specialized on surviving in that niche, or even by constructing their own niche (this last effect was described by Laland (2004)).

In classical EA, an emphasis is put on quality rather than diversity of outcome. Diversity only serves as a means to a goal. In the context of engineering optimization this makes sense. In the end we want solutions to perform as well as possible and computational methods are better at maximizing the optimality of large solution sets than human engineers are. Diversity measures like niching (see Schaaf et al. (2003)) are used to prevent a population-based optimization algorithm from 'getting stuck' in local optima, a sub-optimal region of the solution space. The population within the algorithm is artificially spread out by the combination of a mutation operator, which introduces random variations of the population, and a diversity

measure, subdividing the population between multiple attracting optima, as was discussed by Asteroth and Hagg (2015). Oftentimes a recombination operator is used as well to allow for jumps in the objective landscape, combining components of existing solutions into new artifacts.

Niching is often tightly connected to the concept of speciation. Only individuals that are located in or close to the same niche interact directly. These individuals are likely to be more genetically compatible and produce offspring. In evolutionary computation, speciation is often seen as a core property of artificial niching methods. "*Niching in evolutionary algorithms is a two-step procedure that a) concurrently or subsequently distributes individuals onto distinct basins of attraction and b) facilitates approximation of the corresponding (local) optimizers*", as summarized by Preuss (2006). Basins of attraction represent high fitness regions in the objective function out of which there is no 'escape path' that does not go through a low fitness region. A species will therefore 'nest' itself inside of a basin of attraction.

3.1.1 Genome, Phenotype and Behavior in Nature

Let us now look at the concept of fitness in an example from biology. A simple fitness metric can be defined as the amount of grass in kilograms eaten per day. This metric might be naive but it is not that different from the kind of naive metrics often used in computational evolution, for example the amount of wind resistance that is caused by an airplane wing or the number of crates stacked by a robot per hour. The amount of grass a cow eats is many times larger than what a cutworm is able to consume. Both species are specialized in their own niche, both have reached a local optimum, but when using the proposed fitness metric, their absolute fitness is vastly different. Yet, both species are in an evolutionary stable niche and have been able to survive for long periods of time. We can derive that a highly diverse set of solutions can emerge from natural evolution. In the example above, neither solution is 'preferred', as there is no invisible engineer selecting between those niches.

An ecologic niche can be seen as a basin of attraction in which only species with certain properties can survive. A species that is effective at surviving in a niche is less likely to adapt towards another basin, as it would first have to cross unfit genetic states, although this is not impossible and is often seen in bird species that migrate to islands and become fit with respect to another niche. This can lead

to large changes in the behavior of a species, for example birds losing their flight abilities due to genetic drift (see Wright et al. (2016)).

In computational evolution, a niche is often reduced to a basin of attraction in the objective function over the genome, or encoding. This is very common in benchmark comparisons between algorithms. However, this is a simplification compared to natural evolution. A natural niche can be filled by different genomes. It is not the cutworm alone that can eat grass. Cows can too. Both can digest grass and therefore have similarities in their phenotype, the expressed morphology (the ability to metabolize grass) and behavioral expression (searching grass for nourishment) of a genome. Describing a niche by looking at the genome's expression seems to make more sense than concentrating on its genome alone. The expressed genome, or phenotype, for example the number of jumps performed by a robot biped controller (a behavior), or the amount of curvature in an aerodynamic shape (shape morphology), surely is a closer description of the individual in terms of what niche it can occupy, how successful it can be inside the niche and how it achieves this success.

An example that entails more complex behavior is swimming. Even a genome that does not provide the ability to 'properly' swim, like a fire ant's, can cross rivers through the expression of that genome. Fire ants achieve this through cooperation with conspecifics by self-assembling into a raft, a population level behavior that was observed by Mlot et al. (2011). Thus, a beaver and a fire ant, both having genomes that occupy vastly different basins of attraction in genetic space, can occupy the same ecologic niche according to certain features of their phenotype (see Figure 3.1). They can both cross a river, so display behavioral similarity, and can occupy similar niches, at least those that contain rivers.

An important insight is that a beaver does not swim the same way a colony of fire ants does. The beaver's body performs swimming motions whereas the fire ants need to cooperate. But on a simplified descriptive level, the ability to cross water, the behavior is the same. On the other hand, when we take 'body size' as a description, a large body size will prevent the beaver from entering places fire ants can reach. Fire ants and beavers can be distinguished in terms of the two phenotypic diversity metrics. In one they fall into the same niche, in the other they do not.

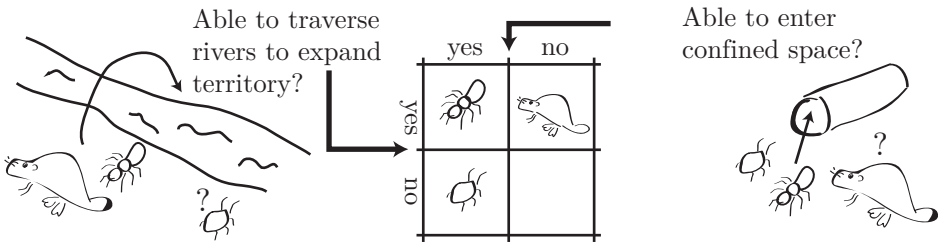

Figure 3.1: Some phenotypic features, like the ability to traverse a river, put the fire ant and the beaver into the same phenotypic niche (rows). Features like body size allows separation between the two species into separate niches (columns). Stink bugs can enter confined spaces like ants do, but they cannot swim, putting them into yet another niche. Both the morphology of the body as well as the behavior of a species determine what niche it can occupy.

It is the interaction between a creature and its environment, the behavior, not the genome, that ultimately defines a natural niche. Dawkins (1982) already posed that behavior can be subsumed into the *extended phenotype*. Non-behaviorial phenotypic traits influence the conditions under which a behavior can take place. A large animal will not be able to occupy niches that need a small body. The morphology of the species can limit or allow certain behaviors. Therefore, certain features of the extended phenotype, be it behavioral or morphological, are important for fitness and help to determine in what niche an individual can flourish.[1]

3.1.2 Encoding in Evolutionary Computation

What does the insight about the extended phenotype's importance mean for evolutionary computation? The distinction between genome and phenotype has been a key inspiration for evolutionary computing, as was described in Sterelny et al. (2001).

Solutions in evolutionary techniques are usually encoded as shown in Fig. 3.2. A discrete set of n parameters, the genome, constitutes the n-dimensional search

[1]Laland (2004) took Dawkins' idea one step further and introduces the idea of *niche construction*. By influencing its environment, a creature can create its own niche. This introduced the idea of evolution being a causally cyclical process. A creature creates a niche by acting on its environment, its genome adapts to the niche, which in itself causes changes in the environment, and so forth.

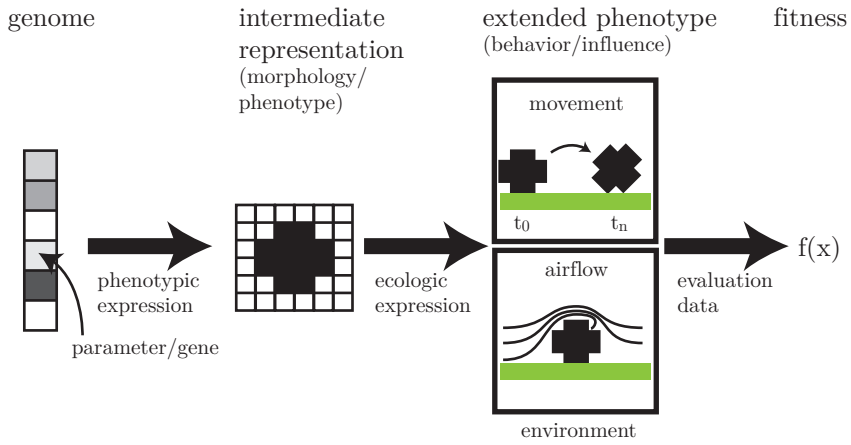

Figure 3.2: Evolutionary representation, an interpretation based on work by Whigham et al. (2017). A genome, consisting of a number of parameters is phenotypically expressed into an intermediate representation, e.g. a morphology. Through its ecologic expression, the behavior or influence on its environment, a solution is in a state where it can be evaluated using one or more fitness functions.

space in which solutions are evolved. Genomes are expressed into intermediate representations, the phenotype, for example a shape or a neural network. But the phenotype is not the full description of a solution. Only once the solution is inserted into the application environment on which it is supposed to act or interact with can we inform ourselves about its quality. One can argue that the 'genome-to-phenotype mapping'[2], the phenotypic expression, has to be extended with a 'phenotype-to-solution mapping'. From here on, separate terms for these mappings will be used: the phenotypic and ecologic expression. The fitness or quality of the phenotype is determined based upon a solution's ecologic expression, its interaction with the environment, or ecosystem.

Neutrality Although the expression function should ideally be one-to-one, it often does not prevent multiple genomes to be mapped to the same phenotype. The phenomenon of such a mapping is called genetic neutrality, which is akin to

[2]In evolutionary optimization, the term genotype is used instead of genome, although this constitutes a divergence from the nomenclature used in biology. There, a genotype is often used as a functional subset of a genome. In computer science, however, we do mean the mapping between the *entire* genome to a phenotype. Hence, the decision is made to use the less ambiguous biological term.

genetic neutrality in biology. In biology, a neutral mutation is understood to have no effect on the survivability of a life form. In evolutionary computation, a neutral mutation changes a genome in such a way that the phenotype is not changed. From here on out let us refer to genetic variants that lead to the same phenotype as neutral genomes (see Hu et al. (2011)).

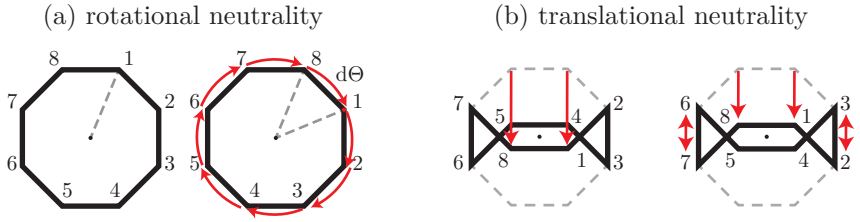

Figure 3.3: Genetic neutrality. The same phenotype is expressed when rotating the control points (numbered one through eight) by a $\frac{\pi}{8}$ angle (left) or by translating the control points as shown (right).

Fig. 3.3a shows an example of neutrality. For angles $\theta = 0°$ and $\theta = 45°$, phenotypically speaking, these shapes are the same. In this case, eight genomes all point to the same phenotype. Similarly, Fig. 3.3b shows how, through translations of the control points, a similar shape can appear based on different genomes.

A neutral genome-to-phenotype mapping (Fig. 3.4) can be highly non-linear. Neutral individuals contain different genomes but produce the same phenotype. A large change to the genome can lead to a small change in the phenotype.

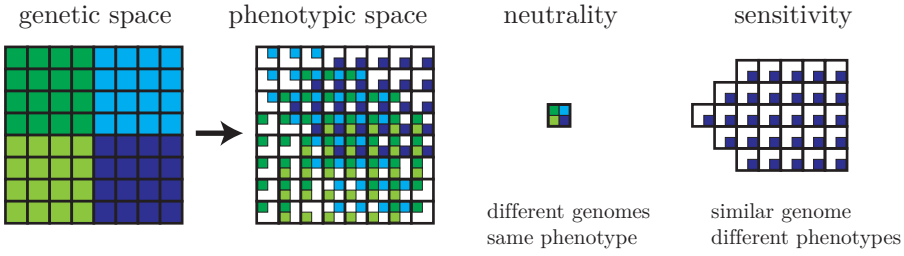

Figure 3.4: Due to genetic neutrality, different genomes (clarified by the use of different colors in the genetic space) can expose a similar phenotype and be put into the same phenotypic niche. Sensitivity can cause similar genomes to produce a relatively large range of phenotypes.

It is postulated that diversity maintenance in a neutral genetic space leads to lower phenotypic diversity than when using phenotypic niching. To increase diversity in phenotypic space, niching should be performed there.

Sensitivity A second effect arises when genomes are projected into phenotypic space in a non-linear way, as shown in Fig. 3.4. A small change in the genome can lead to large changes in the resulting phenotype (see Tarapore et al. (2016)). This effect is coined *sensitivity*[3]. The more sensitive a representation is, the larger phenotypic changes can be when applying small changes to the underlying genome.

Take a simple spline representation. The spline shapes are determined by control points. The shape of the spline can therefore extend beyond the control points' locations, creating a shape that is 'larger' than the space determined by the control points themselves. Fig. 3.5 shows three splines with very similar control point locations. Only the center control point's y value is varied, so the solutions are very close to each other in genetic space. The small changes in control point locations cause larger-scale changes in the phenotype. This, however, is not a mere scale difference, as it is obvious (for us humans) to see the qualitative differences between the solutions' expressed shapes.

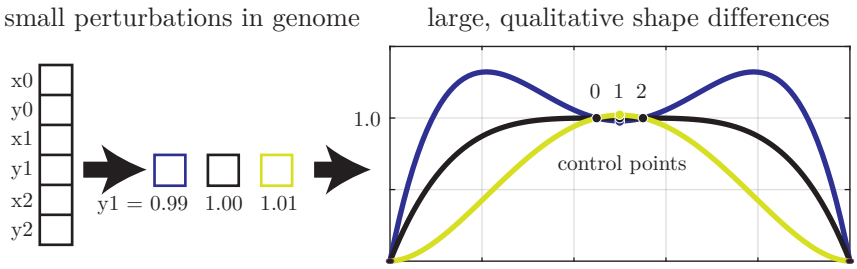

Figure 3.5: Three fifth order spline interpolations that have similar genomes.

The locations of control points relative to each other are more informative for predicting the final shape than their absolute locations. Hence, sampling the parameter space, which determines the control points' locations, does not give as much control over the search in morphological space, as when a search takes place in morphological space directly.

[3]This effect is connected to the concept of evolvability, described by Valiant (2009).

The observation that (genetic descriptions of) species sometimes fill more than one phenotypic niche is not restricted to non-closed form (non-parametric) or indirect encodings. The latter usually refers to neural representations in robotics where the mapping is highly non-linear and dependent on the environment. The effect can happen in any mapping between genome and phenotype. Some evidence was provided by Hagg et al. (2018), in which genomes are clustered in a low-dimensional space. The resulting solution *classes*[4] are not always mapped to a single phenotypic niche. In complex optimization tasks, like aerodynamics, bifurcations (abrupt large changes in the air flow when applying a small change to a shape within that flow) are very common in real world turbulent flow, as was shown by Carroll and Mehra (1982).

3.1.3 Conclusions

The evidence that neutrality and sensitivity occur in real world encodings and applications implies that, if we want to find a functionally diverse set of solutions, we need to search phenotypic or behavioral space, not genetic space. The concept of the extended phenotype and the effects of genetic neutrality and sensitivity provide evidence for an answer to research question I ("Are solutions best compared using their genomes or their expressed phenotype or behavior?"). I postulate that solutions are best compared using their expressed phenotype and behavior. To provide more evidence, the next section introduces the available multi-solution optimization paradigms after which, in Section 3.3, they are compared.

3.2 Multi-Solution Optimization Methods

In this section, three multi-solution paradigms are discussed to understand how the concept of diversity has been used in evolutionary computation. There is a myriad of arguments for diversity in optimization. The concept of diversity itself is considered to be the goal (see Basto-Fernandes et al. (2013)) for a number of reasons: generating alternatives for engineers to choose from, finding trade-offs between features, learning properties of the decision space (*innovization*, see Deb and Srinivasan (2006)), increasing robustness of optimization, allowing solutions

[4]Classes translate to species as defined by Basto-Fernandes et al. (2017).

to adapt to a changing fitness landscape (see Cully and Demiris (2017)), performing model-based diagnostics, or crossing the reality gap between simulation and reality (see Koos et al. (2012)). By understanding the functional diversity of high-performing solutions, we can find more unexpected solutions that provide users of co-creative systems with relevant dissonance effects to expand their intuition.

In this section, three multi-solution paradigms, depicted in Fig. 3.6, that have been developed in the last decades are shown. The paradigms are compared in the next section.

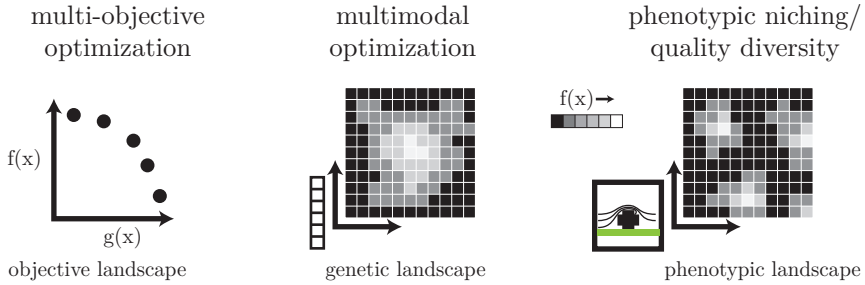

Figure 3.6: Three multi-solution paradigms. The multi-objective paradigm finds a Pareto front of non-dominating solutions along a number of objectives. The multimodal paradigm diversifies solutions according to their genetic distance. The phenotypic or quality diversity paradigm diversifies solutions in terms of phenotypic/behaviorial dimensions.

3.2.1 Diversity in Objective Space

In multi-objective (or multicriteria) optimization (MOO), the Pareto set of trade-off solutions with respect to two or more objective functions is discovered (see Fig. 3.6). One of the most successful methods is the non-dominated sorting genetic algorithm II (NSGA-II) that was introduced by Deb et al. (2002). The paradigm does not control the diversity of the genomes or their expression, only the diversity in objective space, by leveraging a crowding distance metric into the algorithm. However, solutions are expected to be 'different' as they fulfill different trade-offs between objectives. An assumption arises that the front consists of a diverse set of solutions, all with different qualities in terms of the objective functions. Genetic diversity has been established as an important factor in the success of

MOO algorithms, as was shown by Toffolo and Benini (2003) and Shir et al. (2009). As Ulrich (2010) noticed: "*sometimes it might be more important to find a structurally diverse set of close-to-optimal solutions than to identify a set of optimal but structurally similar solutions*".

Although MOO is arguably the first evolutionary technique that explicitly searches for a set of solutions, the field has taken most effort to find the Pareto optimal set and less into investigating the question of (phenotypic) solution diversity.

3.2.2 Diversity in Genetic Space

Computer science research up to the 1970's had already developed interest in population-based evolutionary approaches to optimization, but in that decade specifically, a number of approaches that increase genetic diversity were introduced. Arguably, genetic diversity was first used in evolutionary optimization not to find different solutions, but to deal with the problem of premature convergence to local optima. De Jong (1975) analyzed the problem of premature convergence to local optima in genetic space in his doctoral thesis in 1975. He recognized the need to integrate the concept of *niches* into the evolutionary optimization method. He introduced *crowding*, where child solutions only replace the most similar parent in the population, to improve performance on multimodal surfaces. Alternative techniques are *sharing*, introduced by Holland (1975), where fitness values are penalized due to proximity of other solutions, and *clearing*, introduced by Pétrowski (1996), where the population is subdivided into subpopulations according to their similarity in genetic space.

In the 1990s, multi-solution, multi-local or multimodal optimization (MMO) was put into focus. This paradigm has the explicit goal to find a diverse set of high quality locations in the search space, based on a single criterion (see Fig. 3.6). The first metric that was intended to be used on benchmark problems was peak ratio (see Ursem (1999)), the percentage of local optima that are found.

The field of MMO contains early work on restricted tournament selection to produce multiple solutions by Harik (1995) and was later more established by methods like basin hopping by Wales and Doye (1997), topographical selection by Törn and Viitanen (1992), restarted local search (RLS) by Pošík and Huyer (2012) and nearest-better clustering by Preuss (see Preuss (2010), Preuss (2012), Preuss (2015), and Wessing (2015)).

3.2.3 Diversity in Phenotypic Space

Section 3.1.2 gave evidence that it can make more sense to strive for diversity in phenotypic and behavioral space. A direct search is usually infeasible, due to the high dimensionality of the phenotypic space, or to the fact that it is not describable in closed form and can only be sampled through numerical or physical simulation. Instead, we can perform niching based on behaviorial or other phenotypic features. This is the idea behind quality diversity (QD) [5].

QD produces a set of solutions that are diverse in terms of their extended phenotype, yet still perform well. Instead of applying niching on the basis of genetic similarity, QD considers the behavior of a solution in its environment (see Fig. 3.6), like the average speed of a neural network controlled robot, the amount of turbulence caused by a car, or the amount of curvature of the surface of a complex water drain system, as the basis for niching. This enables QD to find unexpectedly good solutions, or make the optimization process more robust with respect to requirement changes. If one solution does not work well, engineers can take this new knowledge into account when selecting an alternative solution.

The search still usually takes place in genetic space. The high dimensionality of a closed form encoding, where we encode all points as separate parameters, would result in a very high dimensional search space which is not tractable. By intentionally confining the search into a lower dimensional representation, tractability is regained, while phenotypic niching ensures solution diversity.

The idea of measuring similarity on the phenotype level is not new. Pétrowski (1996) mentioned the possibility and the omni-optimizer by Deb and Tiwari (2008) took a more narrow view on phenotypic space, describing it in terms of objectives and constraint values. However, I argue that a fitness function should not be seen as an inherent property of the extended phenotype but rather a metric on the (implicit) goal of a creature or solution's ability to survive.

The first optimization paradigm that embraced behavior-based niching is novelty search (NS). In this new paradigm, Lehman and Stanley (2008) chose to ignore fitness, which can be deceptive and trap a search in a local optimum. Instead, they proposed to use behavioral similarity as an objective function and showed

[5] As in: diversity of qualities. Both quality diversity as well as *illumination* are used in the field, although one could view the deeper concept to be *phenotypic niching*.

that this can circumvent deceptive fitness functions, outperforming fitness-based optimization. They posed that many points in the search space collapse to a small set of simple behaviors, making novelty-based search feasible. NS introduces an archive of past, novel solutions, that serve as a basis for measuring novelty.

The approach was evaluated by evolving controllers for a biped robot. The controllers are able to use motor primitives of the robot's six degrees of freedom to assign poses, based on the inputs of two ground contact sensors only. The authors introduced a novelty metric based on a fixed-horizon sequence of changes in the horizontal coordinates of the robot's center of gravity. To compare two controllers, the sum of squared distances of both sequences was calculated. This way, robots that fall down are assigned the same niche (due to the fixed horizon). NS therefore ignores new solutions that fall down in the beginning and instead concentrates on finding new solutions that follow different trajectories. The results show that NS finds more biped controllers that do not fall down, and identifies better controllers than fitness-based search alone. This shows that NS is not just an exhaustive search method, but is able to ignore easy-to-reach local optima in a very hard search space.

Lehman and Stanley (2011a) posed that the problem of deceptiveness resides in the fitness function, based upon the fact that the underlying neuroevolution algorithm they used provides (genetic) diversity management and is not able to circumvent deceptiveness. The authors did acknowledge that removing the fitness function entirely is a problem in optimization, as it cannot be determined when solutions are fit or whether solutions are functional at all.

Phenotypic diversity allows us to judge different solutions in terms of how they behave in their environment. This can help ignoring solutions that show similar behavior and concentrate the search on novel behavior.

The introduction of NS led to further studying of the search for novel, non-optimal solutions. Inspired by the work of Mouret (2011b) on the reconciliation of NS with fitness, Lehman and Stanley (2011b) extended NS by adding a competitive factor. They introduced novelty search with local competition (NSLC). It formulates fitness and diversity as a bi-objective problem, treating both goals as equally important. NSLC finds a diverse set of high quality optimizers by performing niching in phenotypic space. Shown in applications for developing artificial creatures and robot controller morphologies, NSLC only allows solutions that belong to the same

phenotypic niche to compete. To this end it keeps track of an archive of niches. Solutions are added if their phenotype is novel enough or better than that of a similar solution.

For any new candidate solution, its novelty score is calculated by the average Euclidean distance to a fixed number of k nearest individuals in the archive, where k is determined experimentally, and distance is measured in phenotypic space. Then, the candidate receives a local competition score, represented by the sum of morphologically nearest individuals it outperforms. Both the novelty and local competition scores are then used in an adapted version of the MOO algorithm NSGA-II. The Pareto front that is produced consists of solutions that show the trade-off between novelty and local competition. Parents for a new generation are selected from the individuals in the front to produce new solutions that are different from those saved in the archive and high-performing when compared to similar solutions.

Although NSLC is shown to better exploit morphological niches than fitness- or novelty-based optimization, a number of problems exist with the approach. For one, the archive keeps growing, which makes the comparison of new candidates ever more expensive. The reliance on NSGA-II also makes it difficult to scale up to multiple diversity metrics. These problems were addressed by Mouret and Clune (2012). They formulated multi-objective landscape exploration, which works in a similar way, but emphasizes the aspect of exploring the landscape of phenotypic traits with respect to their fitness. Phenotypic niching is performed by collecting elites mapped by behaviorial traits. Combining this with the idea of building behavior repertoires (see Cully et al. (2013) and Cully and Mouret (2013)), a simplification of the multimodal QD algorithm, multidimensional archive of phenotypic elites (MAP-Elites), was introduced by Mouret and Clune (2015). It *illuminates* the fitness landscape through the lense of phenotypic features.

Algorithm Description QD searches in genetic space for solutions to fill phenotypic niches (Figure 3.7). A low-dimensional phenotype space is defined by selecting a number of phenotypic features, that can be based on behaviorial, morphological, or other features. Solutions' feature coordinates determine what niche they are placed into. They compete only with the solution in that niche. The archive represents the result of the QD algorithm.

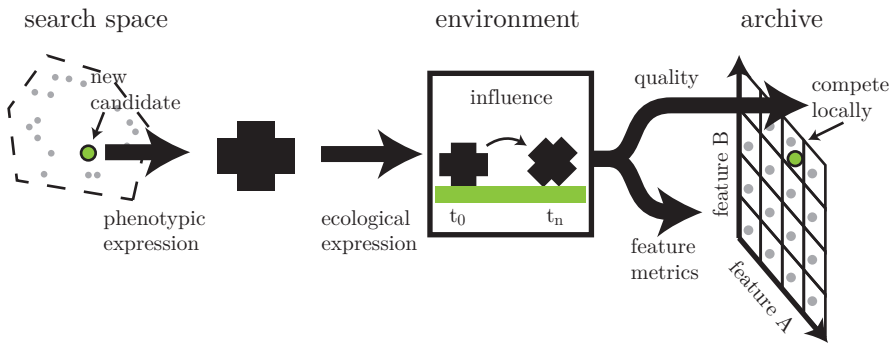

Figure 3.7: QD searches in genetic space but performs niching in a low-dimensional phenotypic archive. Candidate solutions are assigned to the niche based on their feature coordinates. These feature coordinates are calculated based on the expression of a solution, albeit phenotypic or ecologic (see Fig. 3.2). Candidates are only placed inside a niche if they improve its quality value.

QD algorithms follow the pattern described in Alg. 1. After initializing a population of solutions and setting an evaluation budget, parents are selected based on a scoring scheme. In NSLC, parents are selected from a separate population, whereas in MAP-Elites they are selected from the archive itself. Offspring is created, usually using mutation only. The performance of the children is evaluated and their feature coordinates are calculated. The individuals can be added to existing niches through local competition, or new niches can be created, depending on the archive structure that is used. Finally, the selection scores are updated.

Algorithm 1 Quality Diversity

Define and formalize phenotypic descriptors
Initialize genetic population $\mathcal{P}$
Initialize archive $\mathcal{A}$
for iter $= 1 \rightarrow$ generations budget **do**
  Select parents $\mathcal{P}$ to form new offspring based on scoring scheme
  Evaluate performance and phenotypic descriptors of offspring
  Add individuals (potentially) to niches in archive $\mathcal{A}$
  Update selection scores
end for

Cully and Demiris (2017) proposed a modular framework in which the main features of a QD algorithm are described: the type of container or archive and the

procedure for selecting the parents of the next generation.

Archive The archive is the central object in QD. It is constructed based on phenotypic features, and represents the current intuition of the diversity of high-quality solutions to a problem. The archives used in the first QD algorithms were either fixed (MAP-Elites) or unbounded (NSLC) (Figure 3.8). New archive designs allow them to be more efficient, effective, compact and capable of long term optimization.

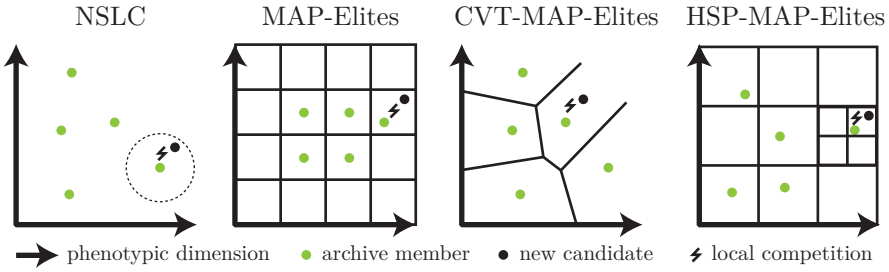

Figure 3.8: The first two QD algorithms introduced an unbounded (NSLC) and fixed-grid bounded (MAP-Elites) version of the phenotypic archive. The fixed grid in the latter can be replaced with a Voronoi or a hierarchical subdivision of the phenotype space. The feature space can also be subdivided in a hierarchical way.

The original definition of the grid, however, has a problem due to the growth of the number of niches, which is exponential in the number of feature dimensions. The ability of the MAP-Elites archive to deal with high-dimensional niching spaces was improved by Vassiliades et al. (2017b). The equidistant grid is replaced with a fixed number of predefined niches, independent of the dimensionality, using centroidal Voronoi tessellation (CVT) (see Figure 3.8). Smith et al. (2016) used hierarchical spatial partitioning (HSP), by which the speed of exploring the niching space was shown to be increased. HSP organizes optimization results in a hierarchical way, which would fulfill requirement B2 ("Knowledge should be organized hierarchically").

In work by Vassiliades et al. (2017b), the fixed extremes of the MAP-Elites archive were expanded during the QD process which leads to changing niche composition and possible removal of solutions. The now unbounded archive performs similarly to the bounded version, without having to manually predefine the extremes/bounds of the niching space.

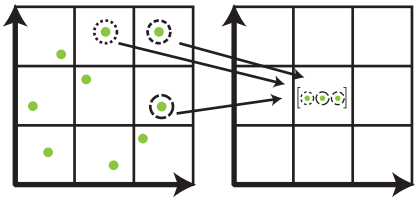

Figure 3.9: Hierarchical behavior repertoire. By building up a hierarchy of stacked archives, each archive can be filled using compositions of primitives from another layer.

The idea of using multiple archives was first introduced by Pugh et al. (2016), to give QD more options to sidestep local optima. Taking this idea one step further, Cully and Demiris (2018) introduced a hierarchical, stacked archive, where each layer uses the layer beneath because a set of primitives can decompose the problem of generating complex behaviors. This hierarchical organization of archives is another way to fulfill requirement B2 ("Knowledge should be organized hierarchically"). Finally, the archive can be decentralized and used in swarm robotics, as was done by Hart et al. (2018), which allows to simultaneously evolve multiple behaviors across multiple robots.

QD and its archive are similar to the idea of the morphological box by Zwicky (1969). They allow new creations by combining known solution configurations. Phenotypic features are an efficient method to represent the knowledge in the box, fulfilling requirements B1 and C5. The archive allows indirectly finding combinations of phenotypic feature, therefore it establishes an 'indirect recombination' operator, which makes the definition an explicit, encoding-dependent operation. obsolete.

Selection Procedure To form new offspring, parents are selected from the current set of solutions. Selection in MAP-Elites was originally introduced as an unbiased operator. Individuals are randomly selected from the archive, whereby each niche can be selected with equal probability. In NSLC the selection is based on both novelty as well as local quality and takes place in the accompanying population, not in the archive.

A number of alternative scoring schemes have been introduced as a basis for selection. Pugh et al. (2015) set the probability of selecting a parent in MAP-

Elites proportionate to the novelty of the solution within the niche of the parent. The authors analyzed when this selection method makes sense by changing the alignment (correlation) between the diversity and quality metrics. The higher this alignment, the more effective the selection procedure is to fill the archive. MAP-Elites can be adapted with this method to behave more like novelty search. The selection procedure automatically reverts back to random selection as soon as the archive is filled.

Another method to improve the speed at which MAP-Elites fills the archive is curiosity-based selection. Curiosity represents the propensity of an individual to generate offspring that gets successfully added to the archive. The metric is defined by counting how often a niche produces offspring that is accepted into the archive because of novelty or better performance. The niches that produce more offspring, or which are more novel, or higher performing, are selected with a higher probability. The method by Cully and Demiris (2017) is not only able to fill the archive more quickly than random selection, but it often finds better solutions in the niches. The selection method can have a significant impact on QD performance, although random selection as a default performs well.

One of the main reasons QD performs well at not only finding diverse solutions, but oftentimes even outperforms classical evolutionary optimization algorithms, can be ascribed to the archive. In the archive, *stepping stones*, solutions that are not high-performing themselves, can lead an evolutionary path to access high-performing regions of the search space, as was shown by Meyerson and Miikkulainen (2017).

Due to the decoupling of genetic search and phenotypic niching, QD supports addition, removal and/or variation of constraints, objectives and variable parameter bounds, fulfilling requirement C4, “Search should support addition, removal and/or variation of constraints, objectives and variable parameter bounds” (Table 2.3).

In QD, the archive becomes the central evolvable object. It can be seen as a lattice, whose points move through genetic space as elites are replaced over time. The QD algorithm used in this work is MAP-Elites, which is explained in detail below.

MAP-Elites MAP-Elites uses a fixed, discretized n-dimensional grid (Figure 3.7), with each dimension representing one phenotypic aspect. This prevents comparisons becoming more expensive as the algorithm progresses and emphasizes the phenotypic space. The number of features, which serve as diversity measures, is limited by the fact that the niching space becomes exponentially larger when adding dimensions. Contrary to NSLC, in MAP-Elites, parents are selected randomly from the archive. There is no distinction between the population and the archive, so it is updated in-place.

Algorithm 2 MAP-Elites (see Mouret and Clune (2015)). The population (X) in the input is usually initialized using a space-filling sequence or random distribution.

1: **function** MAP-ELITES($\mathcal{X}, d(), f(), \sigma$)
2: $\mathcal{D} \leftarrow d(\mathcal{X})$
3: $\mathbf{f} \leftarrow f(\mathcal{X})$
4: $\mathcal{A} \leftarrow (\mathcal{X}, \mathbf{f}, \mathcal{D})$
5: **while** $gens < maxGen$ **do**
6: $\mathcal{X} \leftarrow$ RANDOMSELECT($dim(\mathcal{X})$)
7: $\mathcal{X} \leftarrow$ MUTATE($\mathcal{X}, \sigma$)
8: $\mathcal{D} \leftarrow d(\mathcal{X})$
9: $\mathbf{F} \leftarrow f(\mathcal{X})$
10: $\mathcal{A} \leftarrow$ REPLACE($\mathcal{A}, \mathcal{X}, \mathbf{f}, \mathcal{D}$)
11: **end while**
12: **return** $\mathcal{A}$
13: **end function**

MAP-Elites is shown in Alg. 2. It receives the following inputs: the initial population's genomes $\mathcal{X}$, a descriptor function $d()$ that defines which niche an individual is assigned to, a fitness function $f()$ to determine the quality of an individual, and σ which determines the size of mutations. The archive (which is called 'map' in MAP-Elites) is filled with the initial solution set (line 4), after which an iterative loop is executed for a number $maxGen$ of generations. First, a random set of individuals is selected from the map (line 6), which can either be performed with a uniform random distribution or a space-filling sequence. Individuals are mutated (line 7, see Alg. 3), usually with a normal distribution with standard deviation σ.

The phenotypic descriptors and fitness values are updated for the perturbed individuals, which are then placed into the archive $\mathcal{A}$ using Alg. 4.

Algorithm 3 Mutation.

```
1: function MUTATE(𝒳, σ)
2:     for i = 1 to |𝒳| do
3:         for j = 1 to n do
4:             x_i^j ← x_i^j + 𝒩(0, σ)
5:         end for
6:     end for
7:     return 𝒳
8: end function
```

Algorithm 4 Replacement.

```
1: function REPLACE(𝒜, 𝒳, f, 𝒟)
2:     for i = 1 to |𝒟| do
3:         n ← DISCRETIZE(𝒟_i), 𝒟_i ∈ 𝒟          ▷ 𝒟_i is assigned to niche ID n.
4:         if exists(f(n)) && f(n) < f(i) then   ▷ Replace if i improves niche.
5:             𝒜(n) ← [x_i, f_i, 𝒟_i]
6:         end if
7:     end for
8:     return 𝒜
9: end function
```

Phenotypic descriptors are discretized to determine which niche in the grid-shaped archive they get assigned to. Then, the fitness value of a candidate is compared to the fitness value of the current member of the niche (if the niche is not empty). If the candidate improves the archive, it replaces (or gets assigned to) the niche. After *maxGen* generations, the archive is returned to the user.

3.2.4 Conclusions

The evidence on research question I ("Are solutions best compared using their genomes or their expressed phenotype or behavior?") that was provided in the previous section shows that the (extended) phenotype is a research object that is worthwhile to investigate. Research in evolutionary computation has indeed focused on the phenotype, specifically QD algorithms. QD algorithms fulfill requirements B1 ("Solutions should be categorized") and C5 ("Solution similarity should be based on their functional expression and presented along meaningful

feature dimensions").

Solutions are best compared using their expressed phenotype or behavior, instead of their genomes, due to the effects of neutrality and sensitivity in (evolutionary) representations. Requirement B2 ("Knowledge should be organized hierarchically") can be fulfilled by using hierarchical archives. Features can usually be easily switched, when their calculation can be done quickly, which partially fulfills requirement B3 ("User can filter knowledge base dynamically"). Requirement B3 can also be fulfilled by changing feature dimensions in QD.

In the next section, the main paradigms from multi-solution optimization are compared.

3.3 Comparing Divergent Search Methods

In this section, various techniques in multi-solution optimization are compared to answer research questions I ("Are solutions best compared using their genomes or their expressed phenotype or behavior?") and II ("What multi-solution optimization method produces the highest phenotypic diversity?"). The three main paradigms in multi-solution, divergent search methods, MOO, MMO and QD, are evaluated in terms of the diversity and performance of the solution sets they create. A new niching archive is introduced that allows comparing genetic and phenotypic diversity. State of the art diversity metrics are used in a new problem domain to evaluate all paradigms after which recommendations are made when to use which approach.

A number of survey and analysis articles have appeared in the last decade. In Basto-Fernandes et al. (2013) a taxonomy for diversity in optimization was introduced. Wessing and Preuss (2016) investigated how genetically diverse solution sets in MOO are found and showed that quality indicators used in MOO can be applied to MMO. Vassiliades et al. (2017a) compared two algorithms from MMO to two QD algorithms in a robotics task, showing that clearing performance can be comparable to that of QD. Finally, Li and Yao (2019) discussed 100 solution set quality indicators in MOO and Tian et al. (2019) discussed diversity indicators for MOO. A published scientific comparison between all three paradigms did not previously exist.

The algorithms that provide simple interpretations of diversity in objective space (MOO), genetic space (MMO) and phenotypic space (QD) are selected for this comparison, which answers the question which one of these interpretations of diversity produces a larger diversity of shapes. In this section, which is based on work by Hagg et al. (2020), a new, simple archive is introduced that allows comparing genetic and phenotypic niching with the same algorithmic mechanism.

3.3.1 Niching with Voronoi Tessellation

To remove variations in the search dynamics when comparing different algorithms, a simplified niching variant of NSLC and CVT-Elites is introduced. The Voronoi-Elites (VE) algorithm is illustrated in Fig. 3.10. Selection pressure is applied based on artifact similarity. In effect, VE tries to minimize the variation of distances between artifacts in the (unbounded) archive. The Voronoi niches are not predefined, niche generators do not have to coincide with the centroids and boundaries of the archive are not fixed, as opposed to the CVT-Elites algorithm by Vassiliades et al. (2017b). VE is explained in Alg. 5 and 6.

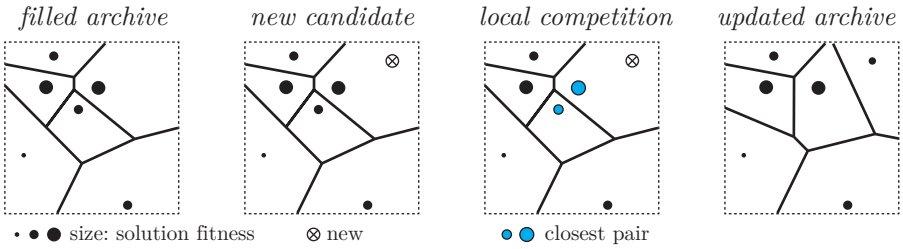

Figure 3.10: Updating the Voronoi archive. The VE formulation allows for a fixed number of archive members, independent of its dimensionality, which makes runtimes more predictable. In this example, the maximum number of artifacts in the archive is set to six. When a new candidate artifact is added, the pair of closest artifacts is compared. The worse of the two is rejected or removed from the archive and the artifact with higher fitness is kept or added to the archive. The borders between niches are drawn here to illustrate the range of influence of each item in the archive and how it changes after an update.

In the main procedure, the initial population of artifacts is created from a quasi-random, space-filling Sobol sequence (see Sobol (1967)) in line 2. The population is evaluated and its fitness values $\mathbf{f}$ and feature descriptors $\mathcal{D}$ determined based on

Algorithm 5 Voronoi-Elites.

```
1:  function VORONOI-ELITES(dim)
2:      𝒳 ← SOBOL(dim)
3:      f, 𝒟 ← EVALUATE(𝒳)
4:      𝒜 ← 𝒜 ∪ (𝒳, f, 𝒟)
5:      for 1 to numGen do
6:          𝒳 ← TOURNAMENT(𝒳, f)
7:          𝒳 ← MUTATE(𝒳, σ)
8:          f, 𝒟 ← EVALUATE(𝒳)
9:          𝒜 ← 𝒜 ∪ (𝒳, f, 𝒟)
10:         while |𝒜| > maxSize do
11:             𝒟 ← dist(𝒟, 𝒟) ▷ 𝒟 contains pair-wise distances between rows in 𝒟
12:             row, col ← argmin(𝒟)        ▷ Ignore diagonal in distance matrix 𝒟
13:             if f_row > f_col, with f_i ∈ f then
14:                 r ← col
15:             else
16:                 r ← row
17:             end if
18:             𝒜 ← 𝒜 \ {(x_r, f_r, 𝒟_r)}
19:         end while
20:     end for
21:     return 𝒜
22: end function
```

fitness and feature metrics, which have to be predefined by the user. The archive $\mathcal{A}$ is filled with tuples of genomes $\mathcal{X}$, their fitness values and feature descriptors. For *numGen* generations, the following pattern is applied. From the population, a set of parents is selected (with the same size as the population) using tournament selection. The function in Alg. 6 compares the fitness values of random parent pairs and selects the best performing parent. Then, the population is mutated by adding a normally distributed random value with a preset σ, shown in Alg. 3 (the same operator as used in MAP-Elites). The new population is evaluated to determine the new fitness and feature descriptor values. The archive $\mathcal{A}$ accepts all new solutions until the maximum number of archive niches is surpassed. The new population is assigned to the archive. The pair of elites that are phenotypically closest to each other are compared, removing the worst-performing from the archive, as shown in lines 10–19. By exerting selection pressure on the closest solutions, VE tries to

equalize the distances between individuals.

Algorithm 6 Tournament Selection.

```
1: function TOURNAMENT(𝒳, f)
2:     𝒲 = {}
3:     for 1 to |𝒳| do
4:         a ← random([1 : |𝒳|]), b ← random([1 : |𝒳|])
5:         if f_a > f_b, with f_i ∈ f then
6:             𝒲 ← 𝒲 ∪ f_a
7:         else
8:             𝒲 ← 𝒲 ∪ f_b
9:         end if
10:    end for
11:    return 𝒲
12: end function
```

Notice that, when genetic parameters are used as archive dimensions, VE behaves like an MMO algorithm by performing niching in genetic space. When phenotypic descriptors are used, VE behaves like a QD algorithm. With MAP-Elites this would theoretically be possible as well, yet the VE formulation allows for a fixed number of archive members, independent of the archive's dimensionality. This makes convergence times (time until an archive is filled) more predictable.

3.3.2 Diversity Metrics

A large number of metrics is available to compare the diversity of a set of solutions in optimization. Only those are considered that do not depend on precise domain knowledge, because no knowledge about actual local optima is available in real world applications. Three commonly used distance-based metrics are selected to evaluate the experiments.

The sum of distances to nearest neighbor (SDNN) measures the size of a solution set as well as the dispersion between members $\mathbf{s}$ of that set (see Wessing and Preuss (2016)). The metric is based on the total minimum dissimilarity between individual solutions, the nearest neighbors, in the set $\mathcal{X}$ (Eq. 3.1 and 3.2). Dissimilarity is measured using a distance metric, like the Euclidean norm.

$$d(\mathbf{s}, \mathcal{X}) = \min_{\mathbf{s}_i \in \mathcal{X}} (dissimilarity(\mathbf{s}, \mathbf{s}_i)) \tag{3.1}$$

$$SDNN(\mathcal{X}) = \sum_{\mathbf{s}_j \in X} (d(\mathbf{s}_j, \mathcal{X})) \tag{3.2}$$

Solow-Polasky Diversity (SPD), introduced by Solow and Polasky (1994), measures the effective number of species by using pairwise distances between the species in the set $\mathcal{X}$. To compute the metric, a matrix $\mathbf{R}$ has to be constructed, with entries $r_{ij} = exp(-\theta d(\mathbf{s}_i, \mathbf{s}_j))$. SPD is then calculated according to Eq. 3.3).

$$SPD(X) = \mathbf{1}^T \mathbf{R}^{-1} \mathbf{1} \tag{3.3}$$

If the solutions are similar with respect to each other, SPD tends to one. The sensitive parameter θ, which determines how fast a population tends to higher values of SPD with increasing distance, needs to be parameterized for every domain. It is set to one for genetic distances and to 100 for phenotypic distances in this section.

Pure Diversity (PD) is used in high-dimensional many-objective optimization (see Wang et al. (2016)). The metric is based on the minimum dissimilarity between individual solutions in the set X (Eq. 3.1). Eq. 3.4 shows that the metric is defined recursively. The PD value of a set X is equal to the maximum of the sum of its value on all but one of the members and the minimum distance of that member to the set. In the base case of the recursion, the set X contains only two members and the recursion stops.

$$PD(X) = \max_{s_i \in X} (PD(X \setminus \{s_i\}) + d(s_i, X \setminus \{s_i\})) \tag{3.4}$$

It does not have parameters, which makes it robust, and depends on a dissimilarity measure, which is usually the $L_{0.1}$-norm to deal with higher dimensional similarity spaces.

Publications in the field of QD have focused on a small number of metrics that do not seem to be very usable and certainly not independent of the used algorithms. The total fitness is used directly or through the *QD-score* by Pugh et al. (2015), which calculates the total fitness of all filled niches in a phenotypic archive. To

achieve this, the solutions from a non-QD algorithm are projected into a fixed phenotypic niching space. This score is domain-dependent and does not allow comparing QD algorithms that have different archiving methods. A comparison between archives created from different features introduces a bias towards one of the archives. The *collection size* indicates the proportion of the niching space that is covered by the collection, but again can only be used on a reference archive, as was done by Cully and Demiris (2017).

Archive-dependent metrics do not generalize well and introduce biases. Therefore, in this work only distance-based diversity metrics are used. The high dimensionality of phenotypic spaces is taken into account by using appropriate distance norms.

3.3.3 Evaluation

A simple polygon optimization domain is defined, in which the objective is to create symmetrical polygon shapes. Shape is an important basic component of design in art, architecture and engineering. On the one hand, shapes can serve as a medium for artistic expression which can also carry semantic meaning (e.g. the shape of a letter of a typeface) and on the other hand can help visualize an object during the design process before it is built or produced. Since the phenotypes can be expressed as binary bitmap images (Figs. 3.11b and 3.11c, resolution of 64x64 pixels), the Hamming (1950) distance is used in the diversity metrics to circumvent the problem of high dimensionality.

Three phenotypic features describing the polygons are defined that can be used either as criteria or as phenotypic features (Fig. 3.11d): the area of the polygon A, its circumference l and point symmetry P through the center. The polygon is sampled at $n = 1000$ equidistant locations on the polygon circumference. The sum of distances of all $n/2$ opposing sampling locations serves as a symmetry error E_s. The error on such a set of opposing samples $\mathbf{X}$ is calculated as shown in Eq. 3.5.

$$f_P(\mathbf{X}) = \frac{1}{1 + E_s(\mathbf{X})}, E_s(\mathbf{X}) = \sum_{j=1}^{n/2} ||\mathbf{x}_j, \mathbf{x}_{j+n/2}||, \text{ with } \mathbf{x}_i \in \mathbf{X} \tag{3.5}$$

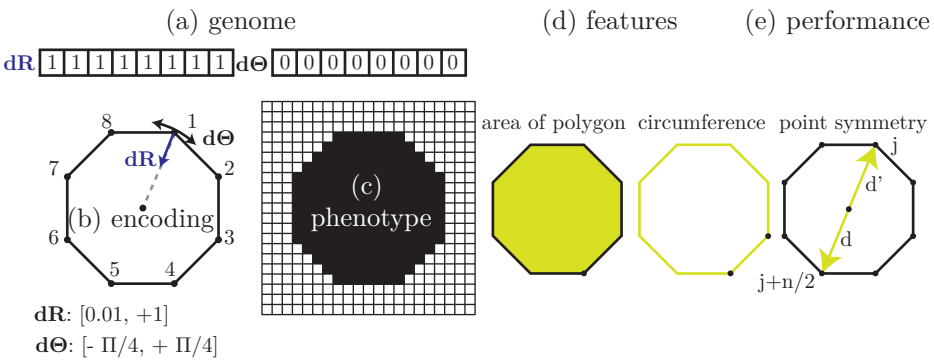

Figure 3.11: Free-form encoding of polygons. The genome (a) consists of 16 parameters that define angular and radial deformations (b). The range of possible gene values differs for the **dR** and **d**θ genes. The phenotype is considered to be the pixel representation of the polygon (c). Shown is a 20x20 phenotype, although 64x64 pixels are used throughout the experiments. Features and optimization criterion of the polygon domain are shown in (d) and (e).

For every opposing pair, the difference in distance to the center location is calculated. When the sum of these distance differences is zero, the shape is defined as fully symmetric.

Genetic or Phenotypic Diversity The Voronoi tessellation used in VE makes it easy to compare archives of different dimensionality by fixing the number of niches. VE is applied as an MMO algorithm, performing niching in 16-dimensional genetic space, and as a QD algorithm with a two-dimensional phenotypic space. The number of niches is increased to determine when differences between genetic and phenotypic VE appear (Fig. 3.12).

At 25 solutions, the approaches produce about the same diversity, but genetic VE finds higher quality solutions. As the number of niches is increased, based on where niching is performed (genetic or phenotypic space), the diversity in that space becomes higher. Phenotypic VE beats genetic VE in terms of phenotypic diversity, which gives us evidence that the answer to research question II should tend towards preferring phenotypic niching over genetic niching. At the same time, the average fitness values of genetic VE are higher than those of phenotypic VE, although the difference gets lower towards 400 solutions.

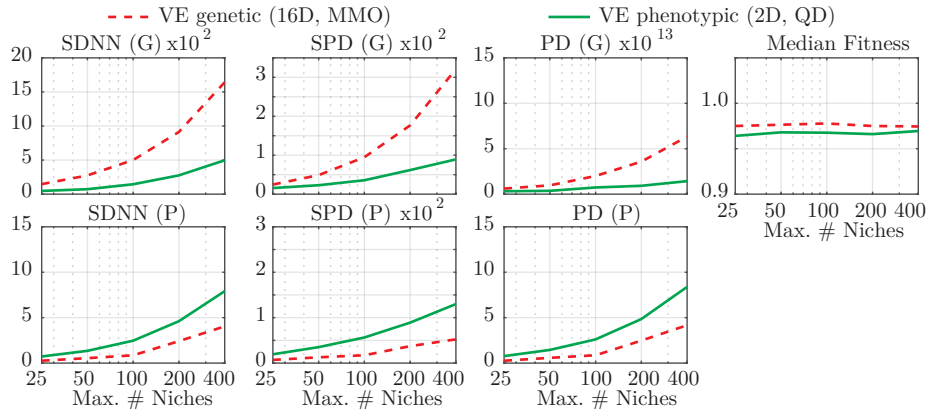

Figure 3.12: VE performed in 16D genetic and two-dimensional phenotypic space. Top: genetic diversity (SDNN, SPD, and PD) and median fitness, bottom: phenotypic diversity. The number of niches/solutions is increased (x-axis).

Comparison of Results The main question to answer is which paradigm provides the highest phenotypic diversity of shapes. VE, RLS (see Section 3.2.2) and NSGA-II (see Section 3.2.1) are compared in multiple experiments. Throughout these experiments the number of function evaluations and solutions is fixed. Five replicates per configuration are produced. In NSGA-II the features are used as optimization criteria, maximizing A and minimizing l. The true Pareto set consists of circles with varying sizes. The number of generations is set to 1024 and mutation strength to 10% of the parameter range. The probability of crossover for NSGA-II is 90% and the probability of mutation is $\frac{1}{dof} = 0.0625\%$, with $dof = 16$ degrees of freedom. VE's archive size is varied throughout the experiments. The number of children (64) and population size is set to the same value. RLS uses as many restarts as the size of the VE archive, and is restricted to the bounds of the genome's parameter ranges. Broyden-Fletcher-Goldfarb-Shanno (BFGS) (see Broyden (1970), Fletcher (1970), Goldfarb (1970), and Shanno (1970)) is used as a local search method. The initial solution set for VE and NSGA-II is created with a Sobol sequence – the initial RLS solution is in the center of the parameter range but RLS' space-filling character assures a good search space coverage.

Phenotypic VE is compared to NSGA-II and RLS. When the genes dr, which encode the radial deviation, are bounded between 0 and 1, and $d\theta$, which encode the angular deviation, are set between $+/-0.125 \times \pi$, genetic neutrality can be minimized. Neutrality is increased by expanding those bounds (Table 3.1). In

contrast to VE, the phenotypic diversity of RLS' solutions is expected to decrease as genetic neutrality increases. Since there is no mechanism to distinguish between similar shapes with different genomes, there is an increasing probability that RLS finds phenotypically similar solutions. It is expected that the solution set produced by RLS is more diverse than using NSGA-II, due to its space-filling character.

Table 3.1: Parameter settings in order of increasing genetic neutrality.

case	angular min.	angular max.	radial min.	radial max.	neutrality
A	0	1	-0.05	0.05	-
B	0	1	-0.125	0.125	+
C	-0.25	1	-0.25	0.25	++
D	-0.5	1	-0.5	0.5	+++
E	-1	1	-1	1	++++

It is expected that NSGA-II will easily find the Pareto set, which consists of circles of various sizes, maximizing the area while minimizing the length of the circumference, while QD should find a variety of shapes that can be any combination of large and small A and l. I postulate: allowing all criteria combinations, instead of using a Pareto approach, leads to higher diversity, while still approximating the Pareto set.

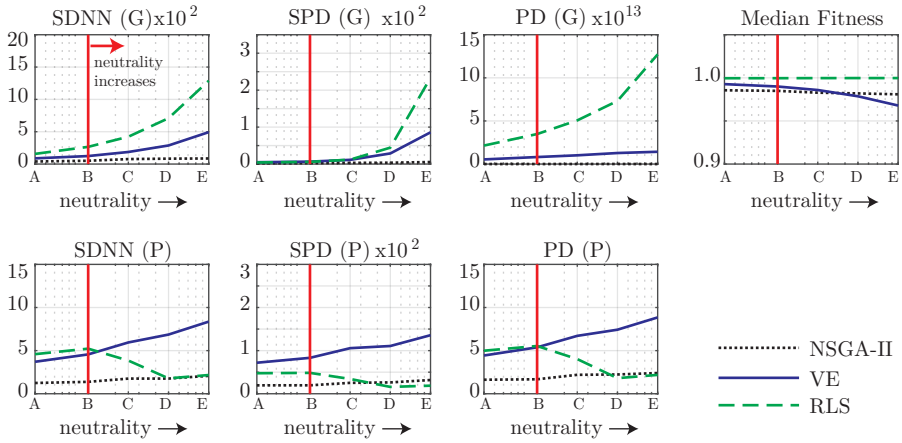

Figure 3.13: Genetic (top) and phenotypic (bottom) diversity, and median fitness. Right of red marker: neutrality increases, using parameter bounds shown in Table 3.1.

The number of solutions is set to 400. A result similar to Fig. 3.12 appears for the standard algorithms in Fig. 3.13. While phenotypic diversity of VE is highest, especially after the genetic neutrality threshold is crossed (at B), diversity of NSGA-II is lowest, as is expected for this setup. Although the diversity of VE is higher than that of RLS, the latter's solutions are all maximally symmetric (see fitness plots), for all settings of the encoding. This makes RLS more appropriate when quality is more important than diversity. The results give us more evidence to answer research question II. (Phenotypic) VE indeed provides a more diverse solution set than any of the other paradigms.

The ground truth Pareto set can be calculated a priori, as we know that circular shapes maximize area while minimizing circumference. The members of the Pareto set adhere to the following genome: $r_1 = r_2 = \cdots = r_8$ and $\theta_1 = \theta_2 = \cdots = \theta_8$. To create 100 shapes from the ground truth Pareto set, ten equidistant values for r and θ are combined.

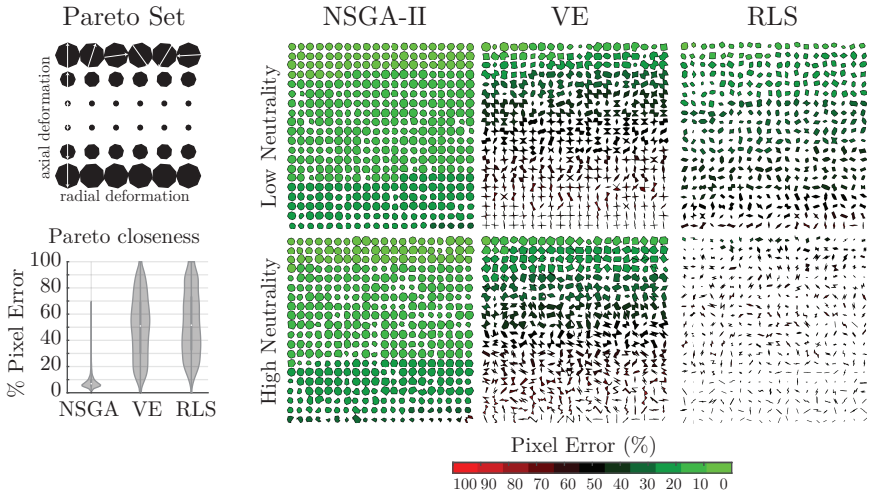

Figure 3.14: The ground truth Pareto set is shown over the entire parameter range, with negative as well as positive values for the radial deformation. Bottom left: closeness to Pareto set, measured as pixel errors. The six figures on the right show example solution sets for low and high neutrality.

Part of the ground truth Pareto set is shown in Fig. 3.14. The distance to the Pareto set is determined in phenotypic space, by measuring the smallest pixel error, the sum of pixel-wise differences, between a solution and the Pareto set. We see

that a number of solutions in VE and RLS are close to the Pareto set (Fig. 3.14 bottom left). Example results with low and high neutrality are shown on the right. Solutions that are close to the Pareto set are shown in the brightest green color. VE again seems to be more robust w.r.t. genetic neutrality, as it finds more solutions close to the Pareto set in high-neutrality domains (bottom row) than RLS. This provides us with even more evidence that QD should be the preferred over MOO and MMO methods because we aim to generate a phenotypically diverse solution set.

3.3.4 Conclusions

This section gave evidence for answers to research questions I ("Are solutions best compared using their genomes or their expressed phenotype or behavior?") and II ("What multi-solution optimization method produces the highest phenotypic diversity?"), fulfilling requirements B4 ("Process should provide diverse perspectives on existing knowledge"), C1 ("Search should efficiently generate a diverse set of novel, functional solutions"), C3 ("Search should identify regions of design feasibility and optimality"), and C4 ("Search should support addition, removal and/or variation of constraints, objectives and variable parameter bounds").

The highest phenotypic diversity is reached by comparing solutions based on their phenotypes, using a QD algorithm instead of MMO or MOO. QD is less sensitive to genetic neutrality than MMO. While the diversity of solution sets of QD and RLS is higher than that of MOO, they both find some solutions close to the ground truth Pareto set. The section included the introduction of the VE algorithm, which allows the comparison between genetic and phenotypic niching.

The next section discusses the phenotypic features that are used in QD and compares predefined features to those that are learned from data.

3.4 Phenotypic Features

Requirements A3 ("The initial solution set presented to user should need as little guidance from the user as possible") and B4 ("Process should provide diverse perspectives on existing knowledge") beg the question, whether we should use predefined features or learn them from data. There certainly is a tension between

explainable, predefined features and the need to maximize solution diversity. In this section I therefore investigate research question III ("Can we produce more diverse solution sets when learning phenotypic niching from data instead of using predefined features?").

3.4.1 Predefined Features

Phenotypic features can be viewed as the concretization of the idea of conceptual spaces, a term introduced by Gärdenfors (2004). The term formalizes concepts on the basis of a geometric structure that consists of a number of quality dimensions, basic features by which objects can be compared. The term is motivated by the notions of conceptual similarity and prototype theory.

The phenotypic features used for niching can be based on solutions' behavior, as was done by Cully et al. (2015), morphology, by Gaier et al. (2018), or other properties of the extended phenotype. For example, Cully et al. (2015) evolved an intuition of well-performing neural controllers for a hexapod robot. By expressing each robot controller, certain phenotypic traits can be measured, like the amount of time each leg touches the ground, along which we perform phenotypic niching. Each niche accepts only those controllers whose legs touch the ground in certain temporal patterns. Within that niche, the best solutions compete to survive. The resulting archive of locally well-performing behaviors provided the authors with many different walking gait strategies, which allows quick switching between strategies when necessary.

Selecting phenotypic features is very much a domain-dependent issue. Yet, we seek to minimize the amount of guidance by the user in the generative stage. Some work has been done on unsupervised and adaptive learning of features. A major breakthrough in unsupervised determination of features are *innovation engines*. The technique, introduced and applied on image generation by Nguyen et al. (2015, 2016), uses a generative model (GM) to learn phenotypic features that are 'interesting'. By training the network on 1000 image classes, its output confidence can be used to determine whether an image generated by a QD algorithm is a good representative of one of the classes. Similarly, the features used in the archive are found using a deep neural network or autoencoder in an unsupervised manner (see Cully and Demiris (2018) and Cully (2019)). Another idea is to measure how well a network is able to compress an image. The better it is able to do this, the

less interesting or novel an image is. Gaier et al. (2019) evolved indirectly encoded images. They periodically trained an autoencoder on all images in the archive thereby allowing themselves to measure a kind of novelty score represented by the image reconstruction error. Higher errors indicate more novel images. If an image is novel enough, it gets saved in the QD archive. The diversity metric is therefore not only unsupervised, but adaptive as well.

In some cases, unsupervised descriptors might help us to find phenotypic features that are hard to define in particular high-dimensional domains. This does reintroduce the issue of explainability, because there is no direct way to understand what the features actually represent. We can only perform a posteriori analysis on the deep neural networks, often only by using simple metrics, like a confidence level on the classification output of the network. Using deep neural networks to find features is not something that can be done in every domain, due to model's need of a large number of training examples. It is no coincidence that innovation engines were introduced for images, a domain in which a large amount of data generally is available.

It is important to take into account the cost of calculating the phenotypic descriptors, which has to be performed for each candidate solution. In this respect, an aspect that is based on e.g. morphology might be less expensive than one that is based on behavior in a simulation.

3.4.2 Learning Features

We might not always want or be able to define phenotypic features that are as influential as possible on diversity. Still, data-driven feature models can help us discover features in more complex domains, although issues around cost and explainability might prevent this.

Up to this point in this thesis, domain knowledge was used to construct a phenotypic niching space with VE. Intuitively, the area and circumference seem like good indicators for phenotypic differences in the used domain. But the comparison between QD and MMO is not completely fair, as the latter does not get any domain information. On the other hand, the features used in QD might not be the most diversifying.

Domain knowledge is now removed from QD and a phenotypic niching space is constructed by using a well-known dimensionality reduction (DR) technique to map the phenotypes to a latent space, as was done by Meyerson et al. (2016) and Cully (2019). This data-driven phenotypic niching approach, named Automatic Voronoi-Elites (AutoVE) and shown in Fig. 3.15, has never been applied to shape optimization.

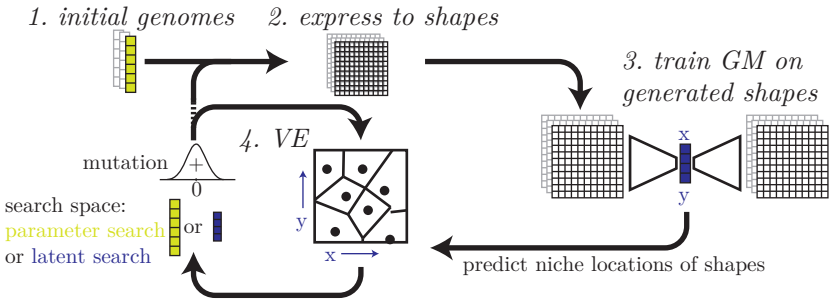

Figure 3.15: Generative model and VE combined into a generative system in two phases. First, initialization: (1) an initial random set of genomes is generated and (2) converted into shape bitmaps which are used to (3) train a GM. Second, optimization loop: (4) VE iteratively updates the archive of candidates. Two setups of this loop are compared: the VE performs search either in parameter space or in the GM's latent space.

An initial set of genomes is drawn from a space-filling Sobol sequence. The genomes are expressed into their phenotypes, which are then used to train a GM. A convolutional autoencoder (cAE), introduced by Hinton (1994), is used here (Fig. 3.16).

The bottleneck in the cAE is a compressed, latent space that assigns every phenotype to a coordinate tupel. The encoder predicts these coordinates of new shapes in the latent space, which are used as phenotypic features. VE is used to search through genetic space and to produce phenotypes that expand and improve the cAE archive.

The reasoning behind using VE as a QD algorithm is as follows. In the original MAP-Elites used by Mouret and Clune (2015), the archive consists of a fixed grid of niches, which leads to an exponential growth of niches with the number of phenotypic feature dimensions. The creators of CVT-Elites, Vassiliades, Chatzilygeroudis, Clune, and Mouret (Vassiliades et al.), dealt with this problem by predefining fixed

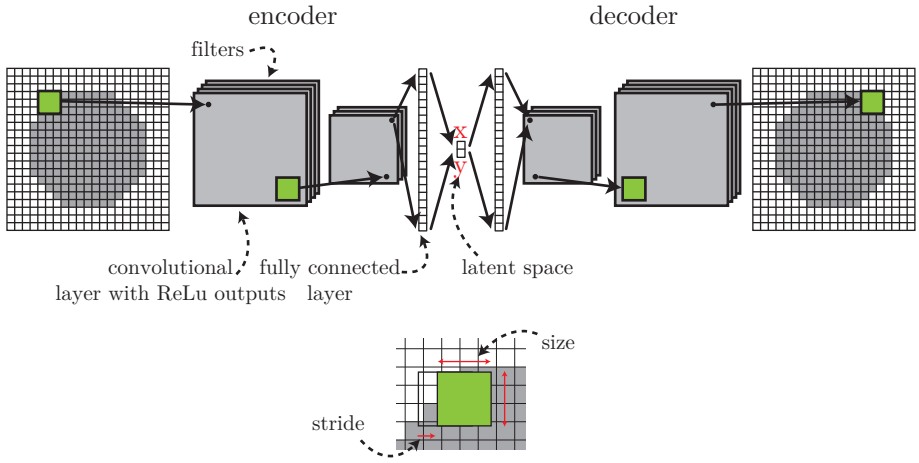

Figure 3.16: A convolutional autoencoder that compresses information, like an image of a polygon, by providing a bottle-neck in the center. The network is trained to reproduce the input at the output. The convolution operates on filters with a 3x3 size and a stride of one.

niches using a Voronoi tessellation of the phenotypic space. Due to their fixed archive, both methods tend to reduce the genetic variance of the solution set in the first iterations. Initial (random) samples tend to not cover the entire phenotypic space and thus competition is harsher, leading to the excluding of many solutions in the beginning. To maximize the number of available training samples for the cAE, the VE algorithm is therefore more appropriate. In contrast, VE, introduced in Hagg et al. (2020), does not precalculate the niches. It accepts all new artifacts until the maximum number of niches is surpassed.

The cAE is retrained with the new samples. It consists of two convolutional layers in the encoder and four transposed convolutional layers in the decoder. The filter size is set to three pixels, the stride to two pixels, and the number of filters to eight. The cAE is trained using ADAM, introduced by Kingma et al. (2015), with a learning rate of 0.001 and 350 training epochs and a mean square error loss function. Latent coordinates are normalized between zero and one.

3.4.3 Evaluation

AutoVE is compared with VE using manual features, continuing the experiment from Section 3.3.3. The number of generations (1024) is divided over two iterations for AutoVE and the number of latent dimensions is set to two (to compare with manual VE), five or ten.

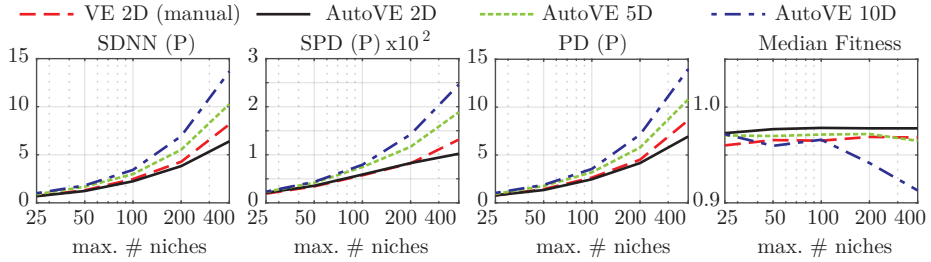

Figure 3.17: Phenotypic diversity and fitness of manually crafted features (VE) compared to using an autoencoder (AutoVE) with two, five or ten latent dimensions.

Fig. 3.17 shows that the two-dimensional manual and autoencoded phenotypic space (AutoVE 2D) produce similar diversity, whereby the quality of solutions from AutoVE 2D is higher. The higher-dimensional latent spaces increase the solution set diversity at the cost of fitness. This is to be expected, as lower-fitness optima are protected in their own niches. Finally, the diversity of higher-dimensional AutoVE is around 50% higher than that of any of the other tested methods in this thesis. Using an autoencoder produces higher diversity than manually defined features, making AutoVE a strong choice for high-diversity multi-solution optimization.

3.4.4 Conclusions

This section gave evidence for an answer to research question III ("Can we produce more diverse solution sets when learning phenotypic niching from data instead of using predefined features?"), fulfilling requirement A3 ("The initial solution set presented to user should need as little guidance from the user as possible") and indicating the approach's usefulness in fulfilling requirement B4 ("Process should provide diverse perspectives on existing knowledge").

More diverse solution sets can be produced when learning phenotypic niching from data instead of using predefined features. An autoencoder was used to discover phenotypic features in a shape optimization problem, showing that we do not need to manually predefine features to get a highly diverse solution set, which increases the evidence that QD is superior in terms of quality and diversity when compared to MOO and MMO.

It is often easy to manually define two or three phenotypic descriptors, but human imagination can run out of options quickly. Automatic discovery of phenotypic features is a more attractive option for increasing solution diversity. Real world multi-solution optimization and understanding solution diversity are important steps towards increasing the efficacy and efficiency at which engineers solve problems.

Requirement A5 ("The process should be adaptable along the resonance-dissonance dimension") is not fulfilled yet. The resonance-dissonance trade-off is not configurable, although this can be accomplished by using e.g. the reconstruction error of the cAE as a feature. This will be evaluated in the next section.

3.5 Limitations of Generative Models

The expectation of Miikkulainen (2020) that the combination of evolutionary computation with deep learning techniques can lead to the next big jump in AI development, machine creativity, begs the question how powerful GM actually are. In this section, research question IV is answered: "What are the limitations of generative models in terms of the possible diversity of the solutions they create?". In order to do this, a comparison is made between multi-solution evolutionary search in the parameter space of a generative system and the search in the latent space of a variational autoencoder (VAE), which were introduced by Kingma and Welling (2014), that was trained with samples from the same system. A generative system is described that produces two-dimensional shapes through the manipulation of eight control points. The system is evaluated in multiple scenarios, including one where it is forced to extrapolate away from solutions that are describable by its latent feature dimensions. This analysis is necessary to estimate, what effect requirement A5 (a resonance-dissonance trade-off should

be configurable) would have on such a system. This is based on work published in Hagg et al. (2021).

Deep generative models such as VAE find application in the context of optimization for their ability to extract patterns from raw data, learn meaningful representations of artifacts and accurately produce more samples with the same properties. Disentangled representation learning can furthermore equip a model's latent space with linearly separated factors of variation, revealing the underlying parameters of a generative process (see Burgess et al. (2017)).

The feature compression networks can be used to compute descriptors for QD algorithms. Defining similar descriptors by hand is a non-trivial task which requires expertise and intuition and, depending on the domain, often cannot compete with an automated solution. Evidence for this was already shown in Section 3.4. While the advantage of learning from data lies in the recognition of complex patterns, the expressiveness of the resulting GM is entirely dependent on the quality and representativeness of the data samples provided. This is especially critical when relying on such a model to produce novel examples and diverse sets of outputs. In fact, artists who employ a generative adversarial network (GAN) in their work often use a variety of strategies to actively diverge from the intended purpose of these models and to produce outputs significantly different from the original data (see Berns and Colton (2020)).

The evaluation is relevant in two scenarios: 1) when the generative process is manually defined but a VAE is used to compare artifacts (e.g. distance/similarity estimation), and 2) when only data is available and the underlying patterns are unknown or too difficult to extract manually and have to be learned by an appropriate model.

3.5.1 Variational Autoencoders

VAEs are a likelihood-based method for generative modeling in deep learning. They follow the standard architecture of an autoencoder: a compressing encoder network, mapping data samples to latent space, and a decoder network which is trained to generate the original samples from the corresponding latent codes (Fig. 3.18). A VAE can generate new samples by interpolating between training locations in the latent space. While common autoencoders draw from an unrestricted range of

latent code values, the latent space of a VAE is typically modelled to be a centered isotropic multi-variate Gaussian ($\mathcal{N}(0, I)$).

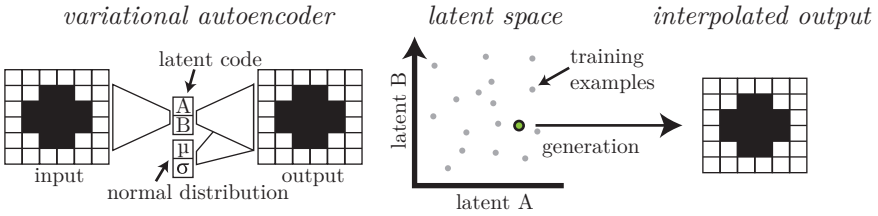

Figure 3.18: Left: variational autoencoder, center: sampling from latent space, right: interpolated output.

The VAE training objective is to optimize a lower bound on the log-likelihood of the data. A beta-annealing variant of the loss term is used to improve disentanglement with improved reconstruction (see Burgess et al. (2017)). This variant of the evidence lower bound (ELBO) calculates the loss function over the predicted output x and the ground truth $\hat{x}$ as follows:

$$ELBO(x, \hat{x}) = C(x, \hat{x}) + \beta \cdot (KL(x, \mathcal{N}(0, 1)) - \gamma) \tag{3.6}$$

Eq. 3.6 consists of a reconstruction loss term, in this case the binary cross-entropy $C(\cdot, \cdot)$, and a regularization term, which penalizes a latent distribution that is not similar to a normal distribution with $\mu = 0$ and $\sigma = 1$. The regularization term is calculated using Kullback-Leibler divergence (KL) and scaled by the parameter β. The annealing factor γ is increased from zero to five during training to focus on improving the distribution in latent space in the beginning of the training and then gradually improve the reconstruction error. The internal latent space of a converged model provides meaningful representations in which distances between data points correspond to their phenotypic similarity. The VAE's internal representation is used to estimate the similarity of artifacts.

Previous work employed autoencoders for dimensionality reduction and the encoding of behavioral descriptors in a control task (see Cully (2019)). In a robotics domain, the approach allows robots to autonomously discover the range of their capabilities, without prior knowledge. GM were used to distinguish parameterized representations in shape optimization (in Section 3.4). They have also been employed to automatically learn an encoding during optimization, using them

as a variational operator (see Gaier et al. (2020)). Other GMs, like GANs, have been used in latent variable evolution by Bontrager et al. (2018) to generate levels for the video games Super Mario Bros. (see Volz et al. (2018)) and Doom (see Giacomello et al. (2019)). A model's latent space is searched with an evolutionary algorithm for instances that optimize for desired properties such as the layout or difficulty of a level. While some authors view the generated levels as novel, none have studied exactly how novel or diverse of an output such a system can produce. The first attempt at such a comparison was made in Section 3.4, but here the comparison is made specifically between using a VAE as a generator and as a method to perform niching in a QD algorithm.

3.5.2 Study Setup

When a VAE is used to generate artifacts, the diversity of its output is bound by the expressiveness of its latent space. The objective of this section is to analyze the generative capabilities of a VAE's latent space and give empirical evidence for its limitations.

This section outlines the details of the study's subject matter, the generation of two-dimensional shapes. It lists the general configurations of the VAE and VE algorithm. Specific settings for experiments can be found in the experimental setups below. The section explains how the two methods are combined to build two versions of a generative system, one using VAE as a search space and one using VAE only to compare solutions and provide a niching space for VE.

Shape Generation For this study, a focus is put on the generation of two-dimensional shapes, an extension of the domain that was previously defined in Section 3.3.3. The shapes in the extended domain are similar to a data set which has been proposed for the evaluation for the quality of disentangled representations (see Higgins et al. (2016)). The setup of the shape generating system in the context of its later use with the VE algorithm is explained here. The shapes are again generated by connecting eight control points which can be freely placed in a two-dimensional space. Each control point is defined by two parameters, the radial (dr) and angular deviation ($d\theta$) from a central reference point, resulting in a total of 16 parameters (Fig. 3.19b). These 16 parameters serve as genomes (Fig. 3.19a),

encoding the properties of each individual and defining the parameter space of the VE algorithm.

Key differences to the last domain are the final phenotype and the symmetry calculation. To form a final smooth outline, the points are connected by locally interpolating splines, as introduced by Catmull and Rom (1974), and Fig. 3.19c). A discretization step renders this smooth shape onto a square grid resulting in a bitmap of 64×64 pixels (Fig. 3.19d).

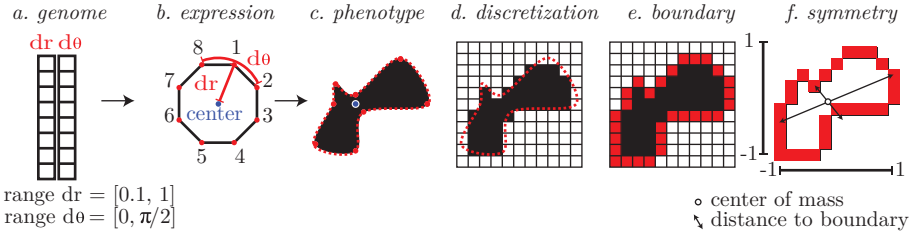

Figure 3.19: Generation of shapes: from (a) 16 genomes to (b) eight control points, freely placed in two-dimensional space, to (c) a smooth interpolated spline and (d) a final bitmap rendering. Quality evaluation: (e) the boundary of the shape is determined and (f) the shapeâĂŹs symmetry is measured from its center of mass.

Fitness As a simple objective and fitness criterion, an adaptation of the point symmetry metric that was used in Section 3.3.3 is used. To determine an artifact's quality, first, the boundary of the artifact is determined (Fig. 3.19e). Second, the coordinates of the boundary pixels are normalized to a range of minus one to one in order to remove any influence of the shape's size (Fig. 3.19f). Third, the center of mass of the boundary is determined to serve as the center of point symmetry. Fourth, the distances to the center of pixels opposite of each other w.r.t. the center of mass are compared. Finally, fitness is calculated as the symmetry error E_s, the sum of Euclidean distances of all $n/2$ opposing sampling locations to the center (see Eq. 3.5). A maximally symmetric shape is one for which this sum equals zero.

VAE Configuration A convolutional variational autoencoder (cVAE) with beta-annealing loss term (see Burgess et al. (2017)) and its decoder is used as a mapping network from latent codes to phenotype bitmaps (see Fig. 3.20). The

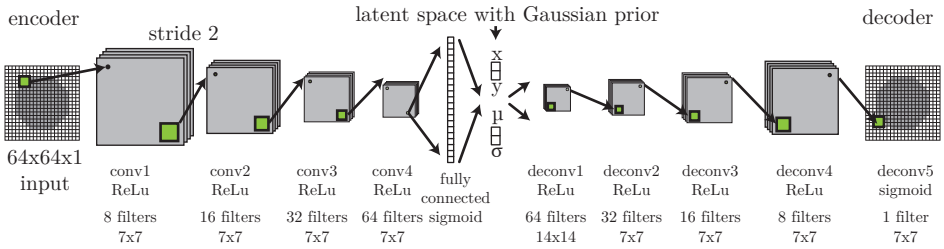

Figure 3.20: Architecture of convolutional variational autoencoder.

model's encoder network is made up of four downscaling blocks, each consisting of a convolution layer (8, 16, 32 and 64 filters respectively; kernel size 7×7; stride two) followed by a rectified linear unit (ReLU) activation function. The set of blocks is followed by a final fully-connected layer. The decoder network inversely maps from the latent space to bitmaps through five transposed convolution layers, which have 64, 32, 16, 8 and 1 filters respectively, kernel size 7×7 and stride two, except for the first layer which has a kernel size of 14×14. The last layer is responsible for outputting the correct size (64×64 pixels). The weights of both networks are initialized with the Glorot initialization scheme Glorot and Bengio (2010). The regularization term scaling factor β was set to four and the annealing factor γ was increased from zero to five over the course of the training, to have the training focus on improving the distribution in latent space in the beginning of the training, and improving the reconstruction error later in the process. Each model was optimized with the Adam optimizer, which was introduced by Kingma et al. (2015), with a learning rate $\mu = 0.001$ and a batch size of 128.

VE Configuration VE starts with an initial set of samples, generated from a Sobol sequence (see Sobol (1967)) in parameter space. Sobol sequences are quasi-random and space-filling. They decrease the variance in the experiments but ensure that the sampling is similar to a uniform random distribution and easily reproduced. In all experiments, VE runs for 1024 generations, producing 32 children per generation. Children are produced by adding a small mutation vector, drawn from a normal distribution centered around zero with $\sigma = 0.1$, to selected parent individuals. The selection is drawn at random from the archive. The number of artifacts in the archive remains constant, identical to the initial population size, over the entire experiment.

AutoVE with a Convolutional VAE The GM in AutoVE, which was introduced in Section 3.4.2, is simply replaced with a convolutional VAE to form a generative system with the objective to produce point symmetric two-dimensional shapes. The full generative process is illustrated in Fig. 3.15 and can be separated into two phases: 1) initialization and 2) an evolutionary optimization loop. At initialization time, a set of random genomes is drawn and translated into bitmaps, their phenotypic counterpart. The VAE is trained to convergence on this set of bitmap data. The learned latent space is then used in the following evolutionary process and the model's encoder and decoder networks serve as mapping functions between the phenotypic bitmap representations and the model's latent representations and vice versa.

In the evolutionary optimization loop, the VE algorithm iteratively updates the archive and tries to increase the diversity as well as the quality of the archive through local competition. To compare two candidates to each other, it relies on the VAE's lower-dimensional latent representations, which preserve semantically meaningful distances. This optimization process is performed in two different search spaces for the central comparison in this study: 1) parameter space (the explicit genome encoding) and 2) the VAE's latent space (the learned representation). This allows evaluating the expressiveness of a VAE's latent space and its capability to generate a diverse set of artifacts in comparison to the full space of possibilities which is reflected by the 16 predefined genetic parameters. The performance of the two approaches is measured in terms of the produced set's diversity. This setup permits studying the limitations of the latent space of a VAE and comparing it to the baseline diversity of searching for candidate solutions over the possible parameters.

This setup can easily be adapted beyond the binary case to gray-scale images represented by floating point values and to images with multiple channels. It further might be extendable to accommodate three-dimensional volumes. As is common in evolutionary computation, the difficulty may lie in mathematically expressing the appropriate fitness functions and measures.

3.5.3 Experiments

It is commonly assumed that a GM, like a VAE, has good interpolative and reasonable extrapolative capabilities, which makes itslatent space a potentially

appealing search space. But how well a search in this space performs in terms of generating a diverse output has not yet been adequately investigated. This section provides a first evaluation and hopes to contribute to a better understanding of GM latent spaces.

In the setup of the generative system the latent space of a VAE is used to search for and generate two-dimensional shapes in the form of square bitmaps. The output diversity of this process is compared to the baseline diversity of a search performed on the explicit genome encoding (parameter space). The pure diversity (PD) metric is used to measure diversity within a set of artifacts (see Section 3.3.2). By calculating PD on a set of bitmaps, diversity can be measured directly, independent of the representation in parameter space or the VAE's latent space.

In this context, insight is gained into two questions: 1) how accurately can a VAE represent a variety of shapes, that is to say how useful are its latent representations, and 2) how well can a VAE generate unknown shapes?

All data sets in the first experiment consist of samples which have been produced by varying two generating factors: scale and rotation (Fig. 3.21). In two experiments, a series of corresponding tasks is evaluated:

(a) With a complete set of samples as a baseline data set the standard reconstruction error of the model is evaluated in order to determine the general quality of latent representations.

(b) In the recombination task, a subset of artifacts in the center of the ranges of values of both generating factors is left out, leaving sufficient examples at either end of the ranges.

(c) In the interpolation task, the left-out subset of artifacts covers the complete range of one of the two generating factors, while some examples at the end of the range remain for the other.

(d) The extrapolation task consists of omitted samples at one end of values of one factor of variation, which affects the complete range of the other factor.

(e) The expansion task focuses on generating artifacts beyond the two given generating factors from the complete data set.

The VAE is expected to perform reasonably well in (b) recombining, (c) interpolating between and (d) extrapolating beyond the available variations to reproduce

the samples missing from the training data. In the expansion task (e), the VAE's latent space is expected to only produce artifacts of poor quality outside of the generating factors present in the training data.

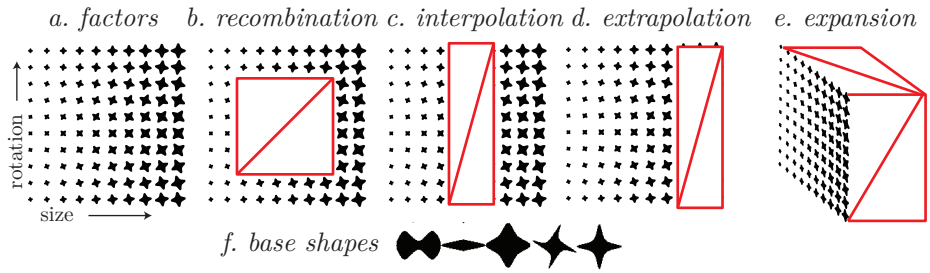

Figure 3.21: (a) generative factors used to create data sets; (b–e) four tasks on which to compare the performance of latent space search with parameter search, the red rectangles indicate artifacts that either have been left out of a data set (b, c, d) or are not available (e); (f) all base shapes used in this experiment.

Recombination, Interpolation and Extrapolation One baseline VAE is trained on the complete shape set (Fig. 3.21a) (256 shapes, scaled by factors of 0.1 to 1.0 and rotated by zero to $\frac{\pi}{2}$ in 16 steps each) and three additional models, each one on the data set of one special task (b–d) with held-out samples. The VAEs are trained for 3000 epochs, after which the models with the lowest validation error are chosen. The validation error is calculated on 10% of the input data.

To determine whether the VAE can correctly reproduce, and thus properly represent, the given shape, the models' reconstruction error is measured. For the baseline model this is done over the complete data set. For the task models (b–d) the reconstruction error is calculated only on the held-out examples. The error is defined as the Hamming distance between an input bitmap and a generated bitmap, normalized by the total number of pixels. A high reconstruction error would indicate that the model cannot properly generate the shapes and that its latent space does not provide an adequate search space for VE. While generating shapes to which there are no corresponding training examples, the reconstruction errors of unseen shapes that can be created with recombination and interpolation (b and c) are expected to be lower than for extrapolation (d).

To determine the resolution of the models, the distances in the latent space between the training examples for the baseline model and between the training and the unseen examples for the task models (b–d) are measured. If the latter are of a similar order of magnitude as the first, the models are able to distinguish unseen shapes from the training examples and from each other. This would indicate that the model's resolution is high enough to provide features for VE.

This experiment was performed on each of five base shapes (Fig. 3.21f) and for three different sizes of the VAE's latent space (4, 8, and 16 dimensions), as it is assumed that the model would not able to perfectly learn the two generating factors. Over the resulting 15 runs, average results are reported.

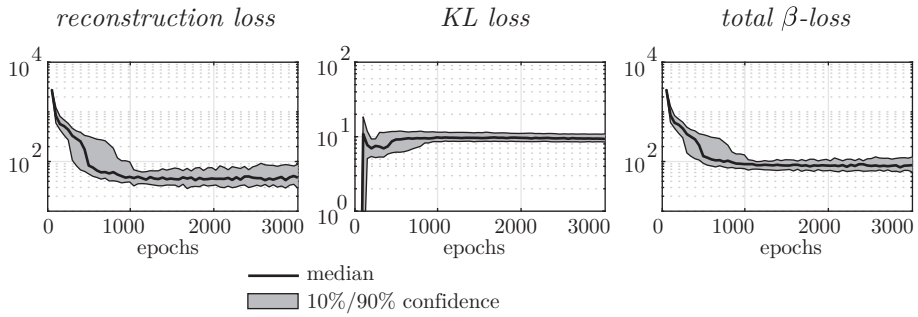

Figure 3.22: VAE validation losses during training.

Fig. 3.22 shows the reconstruction, KL and total β-loss on the validation data during training of the models. The training does not need much more than 1000 epochs to converge.

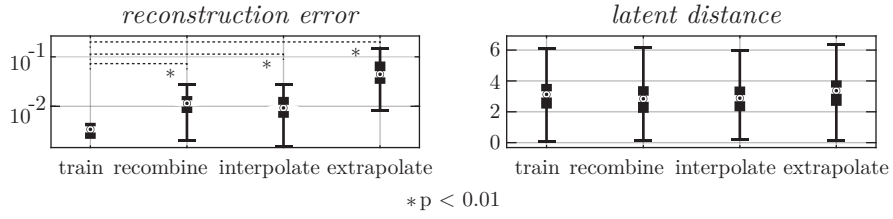

Figure 3.23: Reconstruction errors and latent distances for tasks a–d. Two-sample t-tests are performed comparing the reconstruction errors of the VAE's prediction of the training images to the recombination, interpolation and extrapolation data sets. Comparisons are marked with dotted lines.

Fig. 3.23 (left) shows the reconstruction errors for the training, recombination, interpolation and extrapolation sets. All significant results (two-sample t-test, $p < 0.01$) are marked with an asterisk. The error the models produce on the training samples is significantly lower than when reproducing the recombination and interpolation sets. As expected, the error on the extrapolated shapes is highest. The latent distances between the shapes in the four sets are shown in Fig. 3.23 (right). The distance distributions are similar.

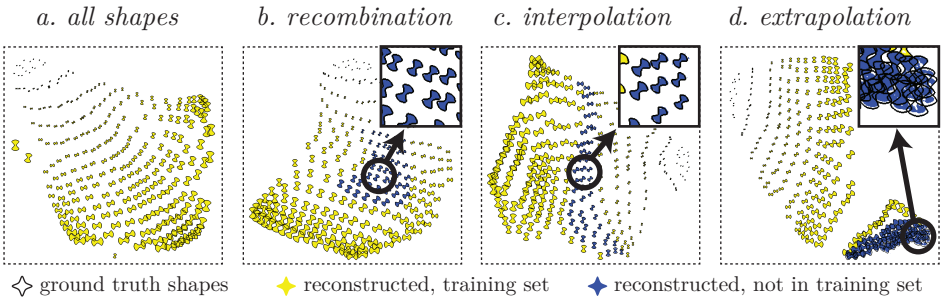

Figure 3.24: Examples of samples and latent spaces produced by the VAE with a latent dimensionality of eight (projected to two dimensions by t-SNE). Shapes in yellow represent samples that were present in the data set, while blue ones were not and have been generated by the model. Black outlines show the ground truth shapes, the difference between the outlines and shapes accounts for errors in reconstruction.

Four exemplary latent spaces are shown in Fig. 3.24 from models with a latent space dimensionality of eight, which has been projected to two dimensions with the dimensionality reduction method t-distributed stochastic neighbourhood embedding (t-SNE). This method was introduced by Maaten and Hinton (2008) and is further described in Appendix B. The first latent space (a, left) corresponds to the baseline model, trained on the complete training set. The other visualizations (b, c, and d) show the three tasks in which some shapes were omitted.

Expansion The last task, expansion (e), cannot be treated as per the previous experiment, because an a priori ground truth shape set 'outside of latent space' cannot be properly defined. Instead, the two search spaces (parameter search (PS) and latent search (LS)) are compared using AutoVE (see Fig. 3.15). It is measured which one of the two search spaces produces the most diverse set of artifacts using the PD metric. The experiment is split up into two configurations.

In the first configuration (**R**), both of the compared search approaches start from the same random initial set of genomes, which is common in many optimization problems. The size of the set is increased to 512, as this experiment poses a more difficult optimization problem. The genomes are translated into bitmaps, which serve as the training data for a VAE model. VE is then performed in both search spaces to fill two separate archives of 512 shapes each. The resulting shape sets are compared w.r.t. their diversity and average fitness, which are often in conflict with each other. As the translation from genome to bitmap always produces a contiguous shape, it is reasonable to expect that a VAE would learn to produce shapes, and not only random noise, even when starting with a randomly generated set of examples.

Often, a generative system does not start from a random set of data, but rather a set of examples that has been observed in the real world. A second configuration, continuation (**C**), is defined to reflect this. An experiment is performed to answer whether the diversity improves when training a VAE with a set of high-quality generated artifacts from a previous VE search. The archive of shapes produced by PS from the random initial set (R) is used as training data for a new VAE model. Both PS and LS are then performed again with this improved model.

It is expected that LS will interpolate between training samples, but will not be able to expand beyond the generative factors in the data, except through modeling errors. Since PS is performed on the explicit genome encoding, this search approach should be able to produce a much higher diversity of artifacts in both configurations **R** and **C**.

The number of latent dimensions of the VAE has been set to 8, 16 and 32 to analyze the influence of the degrees of freedom in latent space, when it is lower than, equal to, or higher than the number of parameters of the genome representation. A higher number of degrees of freedom gives an advantage to the latent model, a lower number would give it a disadvantage. When using 16 latent dimensions, VE deals with the same dimensionality in PS and LS. The number of filters in the VAE is quadrupled to give the model a better chance at learning the larger number of variations.

This experiment has been repeated ten times per configuration: 1) random initial set **R** in PS, 2) continuation **C** in PS, 3) **R** in LS and 4) **C** in LS. Average results are reported over the total of 10 experiment repetitions.

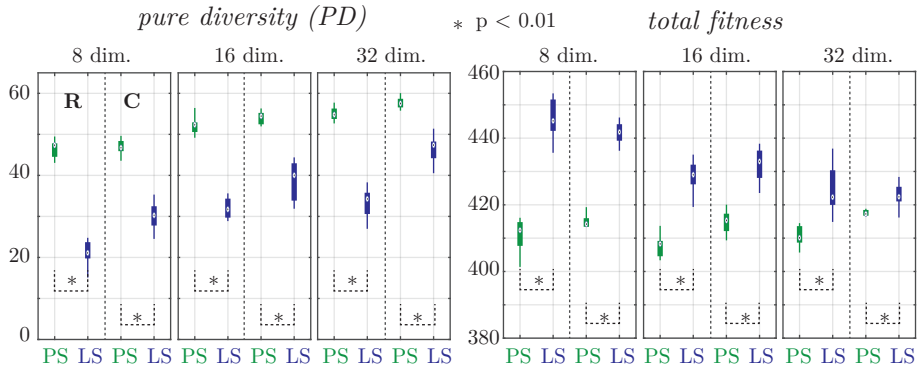

Figure 3.25: Diversity (left) and total sum of fitness (right) of artifact sets of both parameter (PS, green) as well as latent search (LS, blue). VAEs were trained with 8, 16 and 32 latent dimensions , respectively. Both the random initialization (**R**) and continuation (**C**) configurations of the experiments are shown (in every box the two left-hand bars correspond to **R** and the two right-hand to **C**). Significant differences (two-sample t-test, $p < 0.01$) are marked with an asterisk.

Fig. 3.25 compares the diversity and total fitness of the generated artifact sets. The diversity of PS is significantly higher than LS. In turn, LS produces artifacts with higher levels of fitness. Although the difference between PS and LS gets smaller when continuing search from an updated model (configuration **C**), it is still significant. All significant results (two-sample t-test, $p < 0.01$) are marked with an asterisk. Fig. 3.26 shows the expansion away from the latent surface achieved by PS, analogous to the hypothesis in Fig. 3.21e. The PS and LS artifacts' position in 16-dimensional latent space is reduced to two dimensions using t-SNE. The reconstruction error of the model's prediction of the PS artifacts is used as a distance measure to the latent surface.

An example of the resulting shape sets of PS and LS is shown in Fig. 3.27 to illustrate the effective difference in pure diversity.

3.5.4 Discussion

VAEs are able to reproduce recombined and interpolated artifacts quite well (Section 3.5.3). Extrapolation beyond the extremes of the generative factors is more difficult, which the higher reconstruction error in Fig. 3.23 gives evidence for.

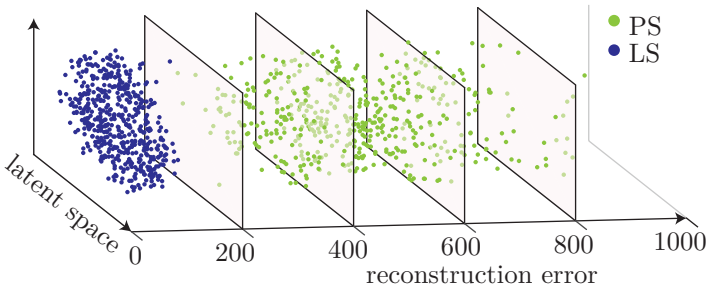

Figure 3.26: Expansion in a 16-dimensional latent model (projected to two dimensions with t-SNE). We interpret the reconstruction error of a shape as its distance from the latent surface. Samples from parameter search (PS, green) tend to extrapolate away from the latent distribution (LS, blue).

The distributions of latent distances between all four data set variants were similar. This provides some evidence that, even when VAEs are not able to reproduce the extrapolated shapes, they can still distinguish them from the training data and from each other. The position of examples in the latent space reflects the semantic relationship between shapes, as visualized in Fig. 3.24. Shapes are not properly reconstructed in the extrapolation task, but they are still positioned in a well-structured manner. The evidence leads to the hypothesis that expansion away from the latent surface will be even more difficult when searching the latent space directly. It also provides evidence that the generative system would be able to fulfill requirement A5 and allow a configurable resonance-dissonance trade-off, as long as the VAE is not used as a search space directly.

Section 3.5.3 attests to an answer to the last task, expansion. The ability of a VAE to find new shapes is indirectly measured by comparing the diversity of the artifact sets created by a parameter and a latent search (PS and LS). The diversity of PS is significantly higher than that of LS, as is shown in Fig. 3.25. This effect holds when the number of latent dimensions is increased beyond the number of degrees of freedom in the original explicit encoding or when the VAE is updated after a first VE run (**C**). Although a trade-off between diversity and fitness is expected, it becomes less pronounced in the 32-dimensional model. This provides evidence that parameter search actually finds both a more diverse as well as a higher performing set of artifacts than latent search. It can therefore be concluded that, when combining VAE and VE, it might make sense to use more powerful

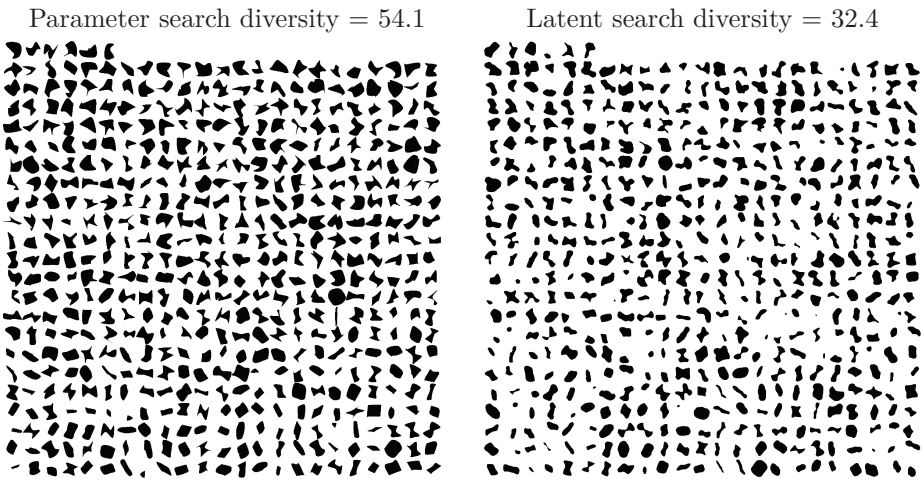

Figure 3.27: The artifact set generated by PS exhibits a higher diversity than the one generated by LS with a 16-dimensional latent model (projected to two dimensions with t-SNE).

encodings than those provided by a GM, but still take advantage of the VAE's ability to distinguish shapes and amplify diversity.

3.5.5 Conclusions

To answer research question IV ("What are the limitations of generative models in terms of the possible diversity of the solutions they create?"), a comparison was made between the use of latent spaces in GMs as a base for divergent search methods, specifically the AutoVE algorithm. The findings shown in this section quantify a VAE's ability to generate samples through recombination, interpolation and extrapolation within and expansion beyond the distribution of a given data set. The diversity of generated artifacts was compared when AutoVE is run either in latent space or parameter space. Based on these observations evidence was given that, in the context of divergent optimization, VAEs should be preferred to be used as a judge of similarity, instead or at least alongside their use as a creator of diverse artifact sets. Using GMs to distinguish solutions in evolutionary computation combines the advantages of automatically discovering the best way to describe similarity and diversity, while still allowing the use of powerful representations that produce more diverse sets of artifacts than data-driven encodings.

The usefulness of a GM's ability to interpolate, extrapolate or expand has to be discussed in the context of its application. In one setting, it might be ideal to perfectly reproduce a given domain and a model might be considered as working well if it can fit a distribution accordingly. In another context, however, and here artistic applications are included, a model's ability to surprise through its unexpected outputs can be of more value and desirable. Here we can use extrapolation to configure a resonance-dissonance effect in the mind of the user.

In the real world, the assumption that a GM can learn a 'perfect' representation does not hold. A perfect model might not produce anything unexpected, only creating high quality artifacts with low diversity. A broken model, on the other hand, might not produce anything useful. The use of modeling errors to find novel artifacts is certainly a mechanism that allows us to find novel solutions within the model. An important question for future work is whether early stopping can be used when training models to create novel yet useful artifacts: is there a correlation between training loss and diversity? This question is not answered in this thesis.

It is safe to assume that generative models are always limited by the training data, which biases models towards what they have learned. Of course, in very high-dimensional domains for which we can collect large data sets, like image and video data, a search in latent space already presents us with a vast amount of possible outcomes, which might be sufficient for some artistic contexts.

3.6 Chapter Summary

This chapter connected evolutionary computational encodings to ecology and showed that solutions are best compared using their (extended) phenotypes, due to neutrality and sensitivity effects. Now, different ways to solve a problem can be provided to users, which should increase the dissonance they experience. This answers research question I ("Are solutions best compared using their genomes or their expressed phenotype or behavior?").

In a critique by Whigham et al. (2017), the authors argue that biological evolution and its 'encoding' are not optimal and should not be blindly applied to evolutionary computation. They reason that genetic search operators must be aware of the consequences resulting from the genome-to-phenotype mapping. In support of

that criticism, I pose that, quite opposite to the emphasis on genetic search, search should take place in the phenotypic space. This space can oftentimes not be searched directly, for example in image generation, numerical or robotic simulations. Instead, a knowledge base needs to be built or learned that contains information about the common features of high-performing solutions.

Taking advantage of this with QD, performing niching in phenotypic space, enables an indirect search in those high-dimensional spaces. QD outperforms other multi-solution paradigms in generating diverse solution sets. A simple, self-expanding implementation of QD, Voronoi-Elites, was introduced, which makes comparisons between genetic and phenotypic search easier. This answers research question II ("What multi-solution optimization method produces the highest phenotypic diversity?").

QD's niching dimensions can be automatically extracted from data with GM, alleviating the user's input prior and increasing solution diversity even more than using predefined features. This answers research question II ("Can we produce more diverse solution sets when learning phenotypic niching from data instead of using predefined features?").

A further critical analysis was performed to highlight the limitations of GM and to answer research question IV ("What are the limitations of generative models in terms of the possible diversity of the solutions they create?"). Evidence was given that indicates that in some cases, GM might be better used as niching dimensions along which an EA can create a diverse set of solutions, instead of using the GM's latent dimensions for search directly. This is how we can enforce dissonance in the mind of the user and discover truly novel solutions.