Discovering Representations for Black-box Optimization

Adam Gaier, Alexander Asteroth, Jean-Baptiste Mouret

Discovering Representations for Black-box Optimization

Adam Gaier
Inria, CNRS, Université de Lorraine
Bonn-Rhein-Sieg University of Applied Sciences
adam.gaier@h-brs.de

Alexander Asteroth
Bonn-Rhein-Sieg University of Applied Sciences
Sankt Augustin, Germany
alexander.asteroth@h-brs.de

Jean-Baptiste Mouret
Inria, CNRS,
Université de Lorraine
Nancy, France
jean-baptiste.mouret@inria.fr

ABSTRACT

The encoding of solutions in black-box optimization is a delicate, handcrafted balance between expressiveness and domain knowledge — between exploring a wide variety of solutions, and ensuring that those solutions are useful. Our main insight is that this process can be automated by generating a dataset of high-performing solutions with a quality diversity algorithm (here, MAP-Elites), then learning a representation with a generative model (here, a Variational Autoencoder) from that dataset. Our second insight is that this representation can be used to scale quality diversity optimization to higher dimensions — but only if we carefully mix solutions generated with the learned representation and those generated with traditional variation operators. We demonstrate these capabilities by learning an low-dimensional encoding for the inverse kinematics of a thousand joint planar arm. The results show that learned representations make it possible to solve high-dimensional problems with orders of magnitude fewer evaluations than the standard MAP-Elites, and that, once solved, the produced encoding can be used for rapid optimization of novel, but similar, tasks. The presented techniques not only scale up quality diversity algorithms to high dimensions, but show that black-box optimization encodings can be automatically learned, rather than hand designed.

ACM Reference Format:
Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. 2020. Discovering Representations for Black-box Optimization. In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3377930.3390221

1 INTRODUCTION

The method of encoding solutions is one of the most critical design decisions in optimization, as the representation defines the way an algorithm can move in the search space [41]. Work on representations tends to focus on encoding priors or innate biases: aerodynamic designs evolved with splines to encourage smooth forms [39], Compositional Pattern Producing Networks (CPPNs) with biases for symmetry and repetition in images and neural network weight patterns [47, 48], modularity induced in evolved neural networks [15, 16, 38], or neural network structures which encode strong enough biases to perform without training [22].

GECCO '20, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7128-5/20/07.
https://doi.org/10.1145/3377930.3390221

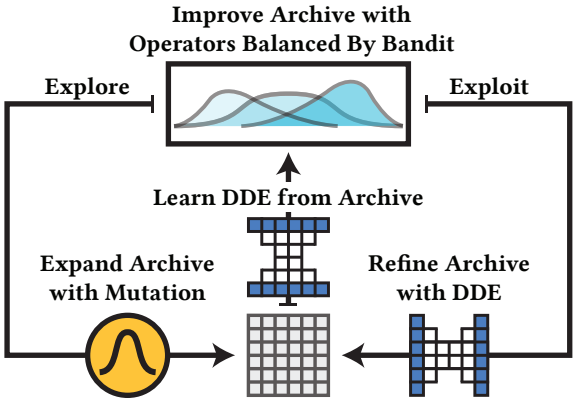

Figure 1: *Data-Driven Encoding MAP-Elites* (DDE-Elites) searches the space of representations to search for solutions. A data-driven encoding (DDE) is learned by training a VAE on the MAP-Elites archive. High fitness solutions, which increase the bias of the DDE toward performance, are found using the DDE. Novel solutions, which increase the range of solutions which can be expressed, are found using mutation operators. UCB1, a bandit algorithm, balances the mix of these explorative and exploitative operators.

The best representations balance a bias for high performing solutions, so they can easily be discovered, and the ability to express a diversity of potential solutions, so the search space can be widely explored. At the one extreme, a representation which only encodes the global optimum is easy to search but useless for finding any other solution. At the other, a representation which can encode anything presents a difficult and dauntingly vast search space.

Given a large set of example solutions, representations could be learned from data instead of being hand-tailored by trial-and-error: a learned representation would replicate the same biases toward performance and the same range of expressivity as the source data set. For instance, given a dataset of face images, a Variational Autoencoder (VAE) [31] or a Generative Adversarial Network (GAN) [25] can learn a low-dimensional latent space, or encoding, that makes it possible to explore the space of face images. In essence, the decoder which maps the latent space to the phenotypic space learns the "recipe" of faces. Importantly, the existence of such a low-dimensional latent space is possible because *the dataset is a very small part of the set of all possible images.*

However, using a dataset of preselected high-performing solutions "traps" the search within the distribution of solutions that

are already known: a VAE trained on white faces will never generate a black face. This limits the usefulness of such data-driven representations for discovering *novel* solutions to hard problems.

In this paper, we propose the use of the MAP-Elites algorithm [37] to automatically generate a dataset for representations using only a performance function and a diversity space. Quality diversity (QD) algorithms [12, 40] like MAP-Elites are a good fit for representation discovery: creating archives of diverse high-performing solutions is precisely their purpose. Using the MAP-Elites archive as a source of example solutions, we can capture the genetic distribution of the highest performing solutions, or elites, by training a VAE and obtaining a latent representation. As the VAE is only trained on elites, this learned representation, or Data-Driven Encoding (DDE), has a strong bias towards solutions with high fitness; and because the elites have varying phenotypes, the DDE is able to express a range of solutions. Though the elites vary along a phenotypic continuum, they commonly have many genotypic similarities [51], making it more likely to find a well-structured latent space.

Nonetheless, MAP-Elites will struggle to find high-performing solutions without an adequate representation. Fortunately, the archive is produced by MAP-Elites in an iterative, any-time fashion, so there is no "end state" to wait for before a DDE can be trained — a DDE can be trained *during optimization*. The DDE can then be used to enhance optimization. By improving the quality of the archive the DDE improves the quality of its own source data, establishing a virtuous cycle of archive and encoding improvement.

A DDE based on an archive will encounter the same difficulty as any learned encoding: the DDE can only represent solutions that are already in the dataset. How then, can we discover new solutions? Fundamentally, to search for an encoding we need to both *exploit the best known representation*, that is, create better solutions according to the current best "recipes", and also *explore new representations* — solutions which do not follow any "recipe".

In this paper, we address this challenge by mixing solutions generated with the DDE with solutions obtained using standard evolutionary operators. Our algorithm applies classic operators, such as Gaussian mutation, to create candidates which could not be captured by the current DDE. At the same time we leverage the DDE to generalize common patterns across the map and create new solutions that are likely to be high-performing. To avoid introducing new hyper-parameters, we tune this exploration/exploitation tradeoff optimally using a multi-armed bandit algorithm [23].

This new algorithm, DDE-Elites, reframes optimization as a search for representations (Figure 1). Integrating MAP-Elites with a VAE makes it possible to apply quality diversity to high-dimensional search spaces, and to find effective representations for future uses. We envision application to domains that have straightforward but expansive low-level representations, for instance: joints positions at 20Hz for a walking robot ($12 \times 100 = 1200$ joint positions for a 5-second gait of a robot with 12 degrees of freedom), 3D shapes in which each voxel is encoded individually (1000-dimensional for a $10 \times 10 \times 10$ grid), images encoded in the pixel-space, etc.

Ideally, the generated DDE will capture the main regularities of the domain. In robot locomotion, this could correspond to periodic functions, since we already know that a 36-dimensional controller based on periodic functions can produce the numerous joint commands required every second to effectively drive a 12-joint walking robot in many different ways [11]. In many domains the space of possible solutions can be vast, while the inherent dimensionality of interesting solutions is still compact. By purposefully seeking out a space of solutions, rather than the solutions themselves, we can solve high-dimensional problems in a lower dimensional space.

2 BACKGROUND

2.1 Optimization of Representations

In his 30 year perspective on adaptation in evolutionary algorithms, Kenneth De Jong identified representation adaptation as "perhaps the most difficult and least understood area of EA design." [14]

Despite the difficulty of creating adaptive encodings, the potential rewards have lured researchers for decades. Directly evolving genotypes to increase in complexity has a tradition going back to the eighties [2, 24]. The strategy of optimizing a solution at low complexity and then adding degrees of freedom has proved effective on problems from optimal control [19], to aerodynamic design [39], to neural networks [49]. Evolving the genome's structure is particularly important when the structure itself is the solution, such as in genetic programming [32] or neural architecture search [17, 22, 35].

Recent approaches toward representation evolution have focused on genotype-phenotype mappings [5]. Neural networks, which map between inputs and outputs, are a natural choice for such 'meta-representations'. These mappings can evolve with the genome [43, 46], or fix the genome and evolve only the mapping [47, 48].

Supervised methods have been previously applied to learn encodings. These approaches require a set of example solutions for training. Where large, well-curated data sets are available this strategy has proven effective at creating representations well suited to optimization [6, 7, 53], but where a corpus of solutions does not exist it must be created. In [36, 44] these solutions were collected by saving the champion solutions found after repeatedly running an optimizer on the problem, with the hope that the learned representation would then be effective in similar classes of problems.

2.2 MAP-Elites

MAP-Elites [37] is a QD algorithm which uses a niching approach to produce high-performing solutions which span a continuum of user-defined phenotypic dimensions. These phenotypic dimensions, or behavior descriptors, describe *the way* the problem is solved, and are often orthogonal to performance. MAP-Elites has been used in such diverse cases as optimizing the distance traveled by a walking robot using different legs [11], the drag of aerodynamic designs with varied volumes and curvatures [20], and the win rate of decks composed of different cards in deck-building games [18].

MAP-Elites is a steady-state evolutionary algorithm which maintains a population in a discretized grid or 'archive'. This grid divides the continuous space of possible behaviors into bins, or 'niches' with each bin holding a single individual, or 'elite'. These elites act as parents, and are mutated to form new individuals. These child individuals are evaluated and assigned a niche based on their behavior. If the niche is empty the child is placed inside; if the niche is already occupied, the individual with higher fitness is stored in the niche and the other discarded. By repeating this process, increasingly optimal solutions which cover the range of phenotype space are found. The MAP-Elites algorithm is summarized in Algorithm 1.

Algorithm 1 MAP-Elites

```
function MAP-Elites(fitness(), variation(), X_initial)
    X ← ∅, F ← ∅                          ▷ Map of genomes X, and fitnesses F
    X ← X_initial                          ▷ Place initial solutions in map
    F ← fitness(X_initial)
    for iter = 1 → I do
        x' ← variation(X)                  ▷ Create new solution from elites
        p', b' ← fitness(x')               ▷ Get performance and behavior
        if F(b') = ∅ or F(b') < f' then    ▷ Replace if better
            F(b') ← f'
            X(b') ← x'
        end if
    end for
    return (X, F)                          ▷ Return illuminated map
end function
```

Though phenotypically diverse the elites are often genotypically similar, existing in an "elite hypervolume", a high performing region of genotype space [51]. Just as in nature, where species as diverse as fruit flies and humans share nearly 60 percent of their genome [1], the "recipe" for high performance is often composed of many of the same ingredients.

This insight was leveraged in [51] to create a new variation operator which considers the correlation among elites. Genes which vary little across the elites, and so are likely common factors that produce high performance, are also subject to the smallest amount of perturbation — lowering the chance their children stray from the elite hypervolume. Biasing mutation in this way ensures that exploration is focused on factors which induce phenotypic variation without drifting into regions of poor performance.

2.3 Variational Autoencoders

Autoencoders (AEs) [28] are neural networks designed to perform dimensionality reduction. AEs are composed of two components: an encoder, which maps the input to a lower dimensional latent space; and a decoder, which maps the latent space back to the original space. The decoder is trained to reconstruct the input through this lower dimensional latent "bottleneck". The encoder component can be viewed as a generalization of Principal Component Analysis [54], with the latent space approximating principal components.

Though the AE is able to represent the data at a lower dimensionality, and reproduce it with minimal loss, it can still be a poor representation for optimization. An important quality of representations is 'locality', that a small change in the genotype induces a small change in the phenotype [41]. When AEs are trained only to minimize reconstruction error they may overfit the distribution of the training data and create an irregular latent space. The low-locality of such latent spaces limits their usefulnesses in optimization: nearby points in latent space may decode to very different solutions, meaning even a small mutation could have a large effect.

Variational autoencoders (VAEs) [31] are AEs whose training is regularized to ensure a high-locality latent space. The architecture is broadly the same: an encoder and decoder mediated by a bottleneck, but rather than encoding the input as a single point it is encoded as a normal distribution in the latent space. When training the model a point from this input distribution is sampled, decoded, and the reconstruction error computed. By encoding the input as a normal distribution we induce the distributions produced by the encoder to be closer to normal. VAEs are trained by minimizing two terms: (1) the reconstruction error, and (2) the Kullback-Liebler (KL) divergence [33] of the latent space to a unit Gaussian distribution, giving the loss function:

$$loss = \|x - \hat{x}\|^2 + KL\left[N\left(\mu_x, \sigma\right), N(0,1)\right] \quad (1)$$

Inducing solutions to be encoded in the form of a normal distribution structures the latent space in a continuous and overlapping way, creating a local encoding better suited to optimization.

3 DDE-ELITES

Every representation biases optimization in some way, improving optimization by limiting the range of solutions that can be expressed to those which are valid or high-performing [41]. But finding a balance between expressivity and bias is an arduous task requiring considerable domain expertise. Our method, DDE-Elites, automates the process of representation design and learns new encodings in tandem with search — allowing optimization and representation learning to improve each other in a self-reinforcing cycle.

DDE-Elites learns an encoding from examples of high performing solutions. To create these examples we use MAP-Elites, which produces a variety of high performing solutions rather than converging to a single optima. The variety produced by MAP-Elites is critical — the expressivity of any learned encoding is limited by the variety of examples. That MAP-Elites not only produces a variety of solutions, but allows us to define the nature of that variety, makes it particularly powerful for crafting useful representations. By defining the type of variety we want to explore we are defining the biases and expressivity we encode in our representation.

DDE-Elites is a variant of the MAP-Elites algorithm. The core component of competition within a niched archive is maintained, but novel methods of producing child solutions are introduced. Child solutions are created using an encoding learned from the archive. This encoding is refined as the archive improves, which in turn improves the optimization process. DDE-Elites optimizes an archive of varied solutions by reframing optimization as a search for the best representation, rather than the best solution.

The DDE-Elites algorithm proceeds as follows (see Figure 2 and Algorithm 2): (1) a DDE and *reconstructive crossover* operator is created by training a VAE on the archive; (2) the probability of using each variation operator is determined by the UCB1 bandit algorithm; (3) MAP-Elites is run with the chosen variation operator probabilities. The success rate of the variation operators to create solutions is used to update the bandit and the improved archive is used to create a new DDE and reconstructive crossover operator.

Data Driven Encoding. The MAP-Elites archive is a record of the highest-performing solutions yet found in each bin. When the archive is updated the VAE is trained to reconstruct the individuals in the archive. Reconstruction is a mapping from one phenotype to another, mediated through latent space; and the mapping from latent space to phenotype space analogous to a genotype-phenotype mapping, which we refer to as a Data-Driven Encoding (DDE).

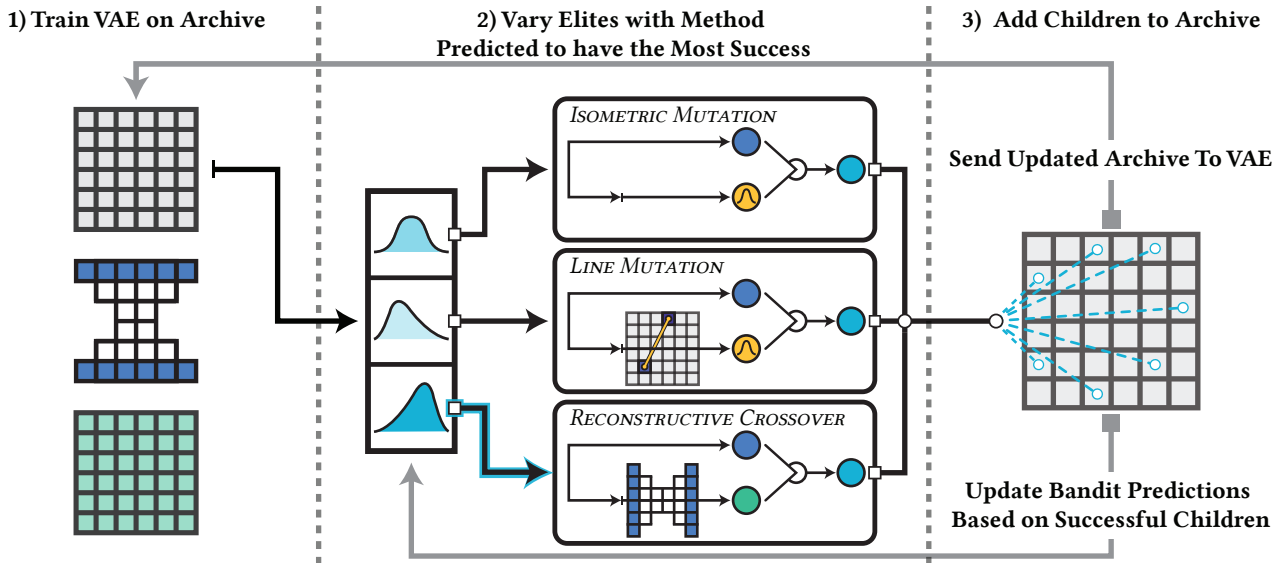

Figure 2: *DDE-Elites Algorithm*
(1) A VAE is trained on the archive, and used to create a 'reconstructive crossover' operator which creates new solutions by averaging the parameters of an individual with its own reconstruction; (2) the mix of exploitative and explorative variation operators predicted to have the most success is chosen by the multi-armed bandit algorithm UCB1 and used to create new solutions; (3) the new solutions are added to the archive and the success rate of the applied variation operator is updated.

Features common in high performing solutions will be the most successfully compressed and reconstructed — and features widely shared by high performing solutions are likely to lead to high performance. Critically, by training the encoding only on high-performing solutions we bias the space of solutions the DDE can express to those with high performance.

Reconstructive Crossover. By limiting the range of solutions which can be expressed by a representation, we are able to bias the solutions found during search. When a solution is reconstructed with the VAE it is mapped onto the restricted space of solutions expressible by the DDE — a space characterized by high performance.

Reconstructing individuals with the VAE can create new solutions with higher fitness than the originals, but cannot create novel solutions. Solutions created by the DDE are based on those already in the archive, so cannot reach solutions which lie outside of the encoded distribution. At early stages of optimization when there are few example solutions, using only reconstruction to create new solutions would doom our encoding to a small region of expression.

Rather than completely replacing individuals with their reconstructions we instead shift them closer to forms expressible by the DDE with a new variation operator, *reconstructive crossover*. Child solutions are created by performing crossover with two parents: a parent chosen from the archive and its reconstruction. Crossover takes the form of an element-wise mean of the parameter vectors.

$$\mathbf{x}_i^{(t+1)} = \frac{1}{2} * (\mathbf{x}_i^{(t)} + VAE.Decode(VAE.Encode(\mathbf{x}_i^{(t)}))) \quad (2)$$

The reconstructive crossover operator slows the loss of diversity by only moving an individual *toward* the distribution of solutions encoded by the DDE, not directly into it. By only shifting solutions rather than replacing them, we allow exploration outside of the distribution to continue. Even when there is little gain in fitness, solutions that are the result of reconstructive crossover have a lower inherent dimensionality, on the account of having parents pass through the compressive bottleneck of the VAE. In this way the reconstructive crossover operator not only spreads globally advantageous genes throughout the archive, but also pulls the archive towards more easily compressed solutions.

Line Mutation. Reconstructive crossover enables effective optimization within the range of solutions that the DDE can express, but explorative operators are required to widen the pool of example solutions and improve the DDE. So when creating new solutions we choose to either produce them through reconstructive crossover, or through random mutation.

In addition to isometric Gaussian mutation commonly used in MAP-Elites, we apply the line mutation operator proposed in [51]. Line mutation imposes a directional component on the Gaussian perturbations. During mutation the parent genome is compared to a random genome from the archive. The variance of mutation in each dimension is then scaled by the difference in each gene:

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \sigma_1 \mathcal{N}(0, \mathbf{I}) + \sigma_2 \left(\mathbf{x}_j^{(t)} - \mathbf{x}_i^{(t)}\right) \mathcal{N}(0, 1) \quad (3)$$

where σ_1 and σ_2 are hyperparameters which define the relative strength of the isometric and directional mutations. Intuitively, when two genes have similar values the spread of mutation will be small, when the values are very different the spread will be large.

In many cases certain parameter values will be correlated to high fitness, regardless of the individual's place in behavior space.

The line operator is a simple way of exploiting this similarity, but in contrast to reconstructive crossover does not limit expressivity – allowing it to be used as a method of exploring new solutions. Though both the reconstructive crossover and line mutation operators take advantage of the similarities between high performing individuals, their differing approaches allow them to be effectively combined as explorative and exploitative operators.

Parameter Control. DDE-Elites explores the space of representations with the exploitative operator of reconstructive crossover, which finds high performing solutions similar to those already encoded by the DDE, and explorative operators of mutation, which expand the space of solutions beyond the range of the DDE.

The optimal ratio to use these operators is not only domain dependent, but dependent on the stage of the algorithm. When the archive is nearly empty, it makes little sense to base a representation on a few randomly initialized solutions; once the behavior space has been explored, it is beneficial to continue optimization through the lens of the DDE; and when the archive is full of solutions produced by the DDE it is more useful to expand the range of possible solutions with mutation. These stages are neither predictable nor clear cut, complicating the decision of when to use each operator.

Faced with a trade-off between exploration and exploitation we frame the choice of operators as a multi-armed bandit problem [3]. Multi-armed bandits imagine sets of actions as levers on a slot machine, each with their own probability of reward. The goal of a bandit algorithm is to balance exploration, trying new actions, and exploitation, repeating actions that yield good rewards. Bandit approaches are straightforward to implement and have been previously used successfully to select genetic operators [13].

We define a set of possible actions as usage ratios between reconstructive crossover, line mutation, and isometric mutation. The ratio of $[\frac{1}{4}, \frac{3}{4}, 0]$, for example, would have solutions created by reconstructive crossover with a probability of $\frac{1}{4}$, line mutation with a probability of $\frac{3}{4}$, and never with isometric mutation. Each action is used to create a batch of child solutions and a reward is assigned in proportion to the number of children who earned a place in the archive. At each generation a new action is chosen, and the reward earned for that action recorded.

Actions are chosen based on UCB1 [3], a simple and effective bandit algorithm which minimizes regret. Actions with the greatest potential reward are chosen, calculated as:

$$Q(a) + \sqrt{(2 \log t)/(N_t(a))} \tag{4}$$

where $Q(a)$ is the reward for an action a, t is the total number of actions that have been performed, and $N_t(a)$ the number of times that action has been performed. UCB1 is an optimistic algorithm which rewards uncertainty — given two actions with the same mean reward, the action which has been tried fewer times will be chosen.

Our archive is in constant flux, and so the true reward of each mix of operators changes from generation to generation. To handle the non-stationary nature of the problem we use a sliding window [23], basing our predictions only on the most recent generations.

Algorithm 2 DDE-Elites

function DDE-Elites($fitness()$ $\mathcal{X}_{initial}$)
  $\mathcal{X} \leftarrow \mathcal{X}_{initial}$
  $\mathcal{V}$: Possible Variation Operator Probabilities (vector)
  (e.g., [0,0.5,0.5], [0.8,0.0,0.2], [1.0,0.0,0.0] for [xover,line,iso])
  successes $\leftarrow zeros(len(\mathcal{V}))$ ▷ # successes for each option
  selection $\leftarrow zeros(len(\mathcal{V}))$ ▷ # selections for each option
  for iter = $1 \rightarrow I$ **do**
    — Train VAE on Current Archive —
    VAE.Train ($\mathcal{X}$)
    — Choose Variation Based on UCB1 —
    $i \leftarrow \arg\max\left(\frac{\text{successes}[s]}{\text{selected}[s]} + \sqrt{\frac{2\ln(\text{sum(successes)})}{\text{selected}[s]}}\right)$
    — Run MAP-Elites Using Chosen Variation —
    $variation() \leftarrow \mathcal{V}[i]$
    $\mathcal{X}' \leftarrow$MAP-Elites($fitness(), variation(), \mathcal{X}$)
    — Track Performance of Chosen Variation —
    $selection[i] \leftarrow selection[i] + 1$
    $successes[i] \leftarrow successes[i] + nImproved(\mathcal{X}', \mathcal{X})$
  end for
  DDE $\leftarrow$ VAE.Decode()
  return $\mathcal{X}$, DDE
end function

function Isometric Mutation($\mathcal{X}$)
  $\mathbf{x} \leftarrow random\_selection(\mathcal{X})$
  return $\mathbf{x} + \sigma\mathcal{N}(0, \mathbf{I})$
end function

function Line Mutation($\mathcal{X}$)
  $\mathbf{x}, \mathbf{y} \leftarrow random\_selection(\mathcal{X})$
  return $\mathbf{x} + \sigma_1\mathcal{N}(0, \mathbf{I}) + \sigma_2(\mathbf{x} - \mathbf{y})\mathcal{N}(0, 1)$
end function

function Reconstructive Crossover($\mathcal{X}$)
  $\mathbf{x} \leftarrow random\_selection(\mathcal{X})$
  $\mathbf{y} \leftarrow VAE.Decode(VAE.Encode(\mathbf{x}))$ ▷ VAE Reconstruction
  return $(\mathbf{x} + \mathbf{y})/2$
end function

4 EXPERIMENTS

Planar Arm Inverse Kinematics. [1] We demonstrate the effectiveness of DDEs and DDE-Elites on in the inverse kinematics (IK) problem of a 2D robot arm, a common QD benchmark problem [12, 51]. Given target coordinates a configuration of joint angles should be found to place the end effector at the target. To solve this task, a discretized behavior space is defined over the x,y plane and MAP-Elites finds a configuration of joint angles which places the end effector in each bin. The location of the end effector is derived for an arm with n joints with angles y with using the forward kinematics equation:

$$\mathbf{b}(\mathbf{y}) = \begin{bmatrix} l_1\cos(y_1) + l_2\cos(y_1 + y_2) + \cdots + l_n\cos(y_1 + \cdots + y_n) \\ l_1\sin(y_1) + l_2\sin(y_1 + y_2) + \cdots + l_n\sin(y_1 + \cdots + y_n) \end{bmatrix}$$

There are many solutions to this IK problem, but solutions with lower joint variance are preferred to allow for smoother transitions

[1] see Figure 5 for a visualization of this domain

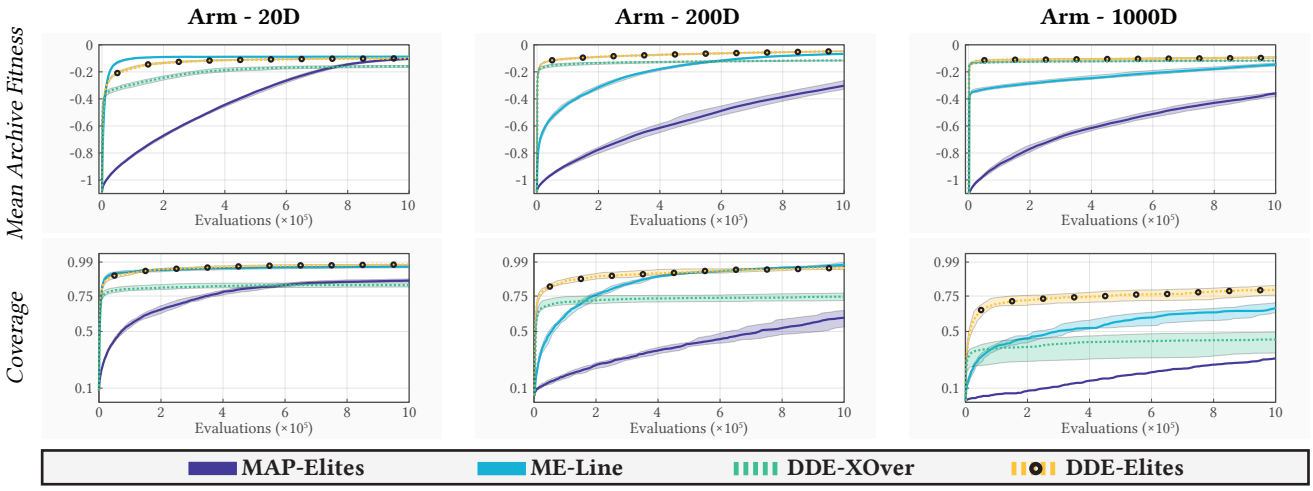

Figure 3: *Archive Illumination*
Archive illumination performance of MAP-Elites with different variation operators: standard isometric mutation (*MAP-Elites*), line mutation (*ME-Line*), reconstructive crossover (*DDE-XOver*) and DDE-Elites, which uses the UCB1 bandit algorithm to choose between the three at every generation. We measure fitness as the mean fitness of all solutions in the archive; coverage as the fraction of behavior space bins which contain solutions. Results over 20 replicates with lines indicating medians and quartile bounds shaded. The median of DDE-Elites, our approach, is additionally noted with black dots. All final results are significantly different ($p < 0.01$ Mann-Whitney U) in fitness and coverage. Progress is shown in evaluations (0 to 1 million); a batch size of 100 evaluations per generation was used, so this scale corresponds to generations from 0 to 10,000.

between configurations. We define fitness as the negative joint variance: $-\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu)^2$ where $(\mu = \sum_{i=1}^{n} y_i)$.

To summarize: the phenotype is the angle of each joint, the behavior is the x,y coordinates of the end effector, and the fitness the negative variance of the joint angles. The difficulty of the problem can be easily scaled up by increasing the number of joints in the arm: we solve this task with 20, 200, and 1000 joints. When a DDE is used 10 latent dimensions are used for the 20D arm, and 32 dimensions for the 200 and 1000D arms. The same archive structure is used for all domains. A unit circle is divided into 1950 bins, with each bin defined by the Voronoi cell [50] with centers placed in a ring formation[2].

4.1 Archive Illumination

We first demonstrate the ability of DDE-Elites to scale up illumination to high-dimensional problems. The performance of DDE-Elites is compared to three algorithmic variants: the canonical MAP-Elites algorithm using isometric mutation (*MAP-Elites*); MAP-Elites using line, or directional, mutation (*ME-Line*); and MAP-Elites using the reconstructive crossover (*DDE-XOver*). Our proposed approach *DDE-Elites* uses all operators at a ratio determined by the UCB1 bandit algorithm. These treatments are summarized in Table 1.

These variants are compared based on the quality of the archive at each generation (Figure 3). Archives are judged based on two metrics: (1) coverage, the number of bins filled, and (2) performance, the mean fitness of solutions.[3]

[2]See supplementary material for a visualization of this structure

[3]Sixty-four core machines were used to evaluate 100 individuals in parallel, requiring ~0.2s, ~0.8s, ~1.6s, for the arm at 20d, 200d, and 1000D arm respectively. In every case the VAE required ~2.4s to train on a single CPU core.

	Isometric Mutation	*Line Mutation*	*Reconstructive Crossover*
MAP-Elites	X		
ME-Line		X	
DDE-XOver			X
DDE-Elites	X	X	X

Table 1: Algorithm variants. DDE-Elites is our approach.

In the 20-dimensional case ME-Line quickly fills the map with high performing solutions. In only a one hundred thousand evaluations ME-Line creates an archive unmatched by MAP-Elites even after one million evaluations. When only the reconstructive crossover operator is used, despite promising early progress, a chronic lack of exploration results in archives which are worse than the standard MAP-Elites. DDE-Elites, with access to all operators, explores as quickly as ME-Line and creates archives of similar quality.

When the dimensionality of the arm is scaled up to 200D, we see the convergence rate of ME-Line slow down considerably. While still reaching high levels of performance it does so only after one million evaluations, a tenth of the evaluations required in in the 20D case — suggesting that the effectiveness of ME-Line scales linearly with the dimensionality of the problem. In contrast DDE-Elites is barely affected by a ten-fold increase in parameters — exploration is only slightly slowed, and high-performing solutions are found from the very earliest iterations. The effects of scaling can be observed even more clearly in the 1000D case: ME-Line illuminates the archive only very slowly, while the performance of DDE-Elites is marked by the same burst of exploration and consistently high

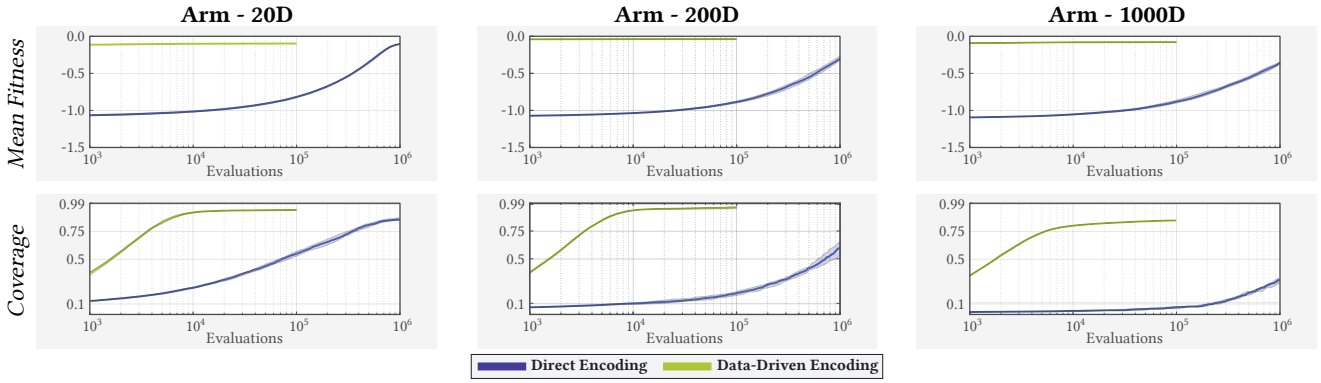

Figure 4: *Archive Recreation with Data-Driven Encoding*
Performance of MAP-Elites algorithm when run with direct or data-driven encoding. When using the direct encoding, MAP-Elites was given one order of magnitude more evaluations (note logarithmic scale of evaluations). Fitness is measured as the mean fitness of all solutions in the archive, coverage as the fraction of behavior space bins which contain solutions. Results over 50 replicates with dotted lines indicating medians and quartile bounds shaded.

fitness solutions that characterized its performance in lower dimensions.

The line mutation operator is clearly able to leverage the similarities in high performing solutions across the archive — in every case performing far better than the isometric mutation operator. The mechanism for doing this, adjusting the range of parameter mutations, does not appear to scale well enough to handle very high dimensional problems. The reconstructive crossover operator is able to rapidly find high-performing solutions even in high-dimensional spaces, but is poor at exploring. Search with reconstructive crossover is confined to the distribution of genes that already exist in the archive, if used exclusively that distribution of genes is limited to the initial population. By combining these operators — expanding the range of genes in the archive with mutation, and spreading high performing genes with reconstructive crossover — DDE-Elites is able to create high-performing archives even in high-dimensional problems.

4.2 Archive Recreation

DDE-Elites is as much a method of optimizing representations as solutions. By learning a representation from the archive, we create an encoding that is biased towards high performance and has a range of expression matching the defined behavior space. In these experiments, our DDE encodes smooth joint configurations which place an arm's end effector anywhere in its reach. To demonstrate that DDE-Elites does more than guide search, but learns a representation, we search the space again, using the found DDE in place of the direct encoding.

We run the standard MAP-Elites algorithm, with isometric mutation only, using a learned DDE[4] acting as our genome. In the 20D arm this DDE has 10 parameters, in the 200D and 1000D arms the DDE has 32 parameters. No previous solutions are maintained, *only the trained DDE*. For reference we compare to the MAP-Elites algorithm using the direct encoding. An order of magnitude fewer evaluations were budgeted when using the DDE.

In every case the DDE far outperforms the direct encoding, reaching the same levels of fitness and coverage with several orders of magnitude fewer evaluations (Figure 4). The DDE can express the same range of solutions as were found in the original archive, and finds them rapidly. Archives were recreated after only 10,000 evaluations — a rate of about 5 evaluations per bin.[5] The found solutions are also high performing.

Such improvement cannot be explained away by the decrease in dimensionality of the search. In both low and high dimensional cases the bias toward high performance is also apparent: the mean fitness curve is nearly flat at the optima, indicating that when new solutions are added to the map they are already near optimal. The contrast with the direct encoding is stark, with the direct encoding considerable effort is taken to search for good solutions, the DDE finds little else. DDE-Elites not only produces solutions, but learns domain-specific representation.

4.3 Optimization with Learned Encodings

Beyond its place in the DDE-Elites optimization loop, the produced DDE is a powerful representation with high expressivity and built in biases. Though created by MAP-Elites, the DDE is not tied to it. Once discovered, a DDE can be used as a representation for any black box optimization algorithm.

We illustrate this generality by using again solving the arm inverse kinematics problem with the black-box optimizer CMA-ES [26]. A set of target positions for the end effector is defined (Figure 5, left), and CMA-ES used to find a joint configuration which reaches each target. In one case optimization is performed using the DDE; in the other the direct encoding is used.

When optimizing with the DDE, CMA-ES quickly finds solutions to the target hitting problems with a precision never matched with the direct encoding (Figure 5, top). Moreover, a bias for *how* the problem is solved is built into the representation (Figure 5, bottom). As the DDE was trained only on solutions with high fitness, that is low joint variance, this same property is found in the solutions

[4] The decoder network of the VAE found in the highest coverage replicate of DDE-Elites.

[5] 10,000 individuals/1950 bins ≈ 5 evaluations/bin discovered.

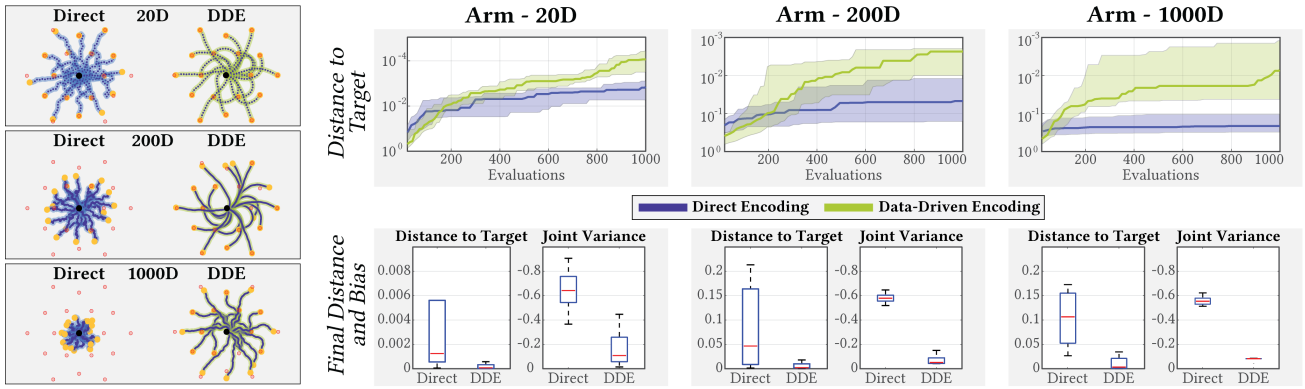

Figure 5: *Optimization with Direct and Data-Driven Encodings*
CMA-ES is given a set budget to find a solution with a target behavior, and searches with either a direct encoding or a DDE. ***Left:*** **Example solutions for target matching with the direct and data driven encodings. End effectors in yellow, targets in red.** ***Top:*** **Optimization over time of median distance (dotted line) to the 18 targets over 50 replicates (quartiles shaded).** ***Bottom:*** **The final distance to the targets, and a characteristic of the solution. These characteristics were not optimized by CMA-ES, but optimized during the creation of the DDE, biasing the solutions produced.**

found by CMA-ES with the DDE — even without searching for them. With the DDE CMA-ES not only finds solutions to the IK problem, with the built-in priors of the DDE it finds the solutions we want.

5 DISCUSSION

Learning representations by combining quality diversity (here, MAP-Elites) and generative models (here, a VAE) opens promising research avenues for domains in which optimizations of the same cost function are launched continuously. This is, for example, the case of Model Predictive Control [34], in which the sequence of actions for the next seconds is optimized at every time-step of the control loop, or the case of shape optimization in interactive design tools [4, 29], in which each modification by the user requires a novel optimization.

In preliminary experiments, we searched for an encoding to describe action sequences for a walking robot. The results show that using MAP-Elites to generate a diversity of sequences, then using a VAE to learn a representation leads to an encoding that can accelerate future optimizations by several orders of magnitude. Nevertheless, using the representation during optimization, as described in this paper, did not accelerate the quality diversity optimization as much as in the high-dimensional arm used here. One hypothesis is that the regularities in action sequences are harder to recognize than in the arm experiments, especially at the beginning of the process. For instance, it might help to use an auto-encoder that is especially designed for sequences [10, 52].

For other tasks, appropriate generative models could be explored, for example convolutional models for tasks with spatial correlations [42]. In addition, though the latent spaces created by VAEs are easier to navigate than those created by normal autoencoders, even better models offer the opportunities for further improvements. Much work has been done to create VAEs which have even better organized latent spaces [8, 9, 27, 30], ideally with each dimension responsible for a single phenotypic feature such as the lighting or color of an image.

A second research avenue is to improve the bandit algorithm that is here used to balance between operators. In theory, it should ensure that adding new operators can only help the process, since useless or detrimental operators would be never selected. However, we observed that it is not always effective: in some cases, using only the line mutation outperformed DDE-Elites, whereas DDE-Elites could revert to using only line mutation with a perfect bandit. Our hypothesis is that this is a sign that "successes" — child solutions which discover new bins or improve on existing solutions — is not the perfect measure of utility for a QD algorithm. In the case of our experiments, it may be that reconstructive crossover can consistently improve solutions, but may only do so slightly. According to the "success" measure, a tiny improvement is worth the same as a large one, or for discovering a new bin. To best utilize the bandit, other methods of judging performance in QD algorithms should be explored.

Beyond performance advantages, for both the current and future optimizations, these "disentangled" representations offer even more interesting opportunities. Reducing the dimensionality of the search space into meaningful components would allow rapid model-based optimization of single solutions [45], or entire archives [21]. Engineers could interactively explore and understand such encodings, laying bare the underlying properties responsible for performance and variation — and so from encodings receive, rather than provide, insight and domain knowledge.

ACKNOWLEDGEMENTS

This work received funding from the European Research Council (ERC) under the EU Horizon 2020 research and innovation programme (grant agreement number 637972, project "ResiBots") and the German Federal Ministry of Education and Research (BMBF) under the Forschung an Fachhochschulen mit Unternehmen programme (grant agreement number 03FH012PX5, project "Aeromat").

SOURCE CODE

The source code used to produce the results in this paper is available at https://github.com/resibots/2020_gaier_gecco

REFERENCES

[1] Mark D Adams, Susan E Celniker, Robert A Holt, Cheryl A Evans, Jeannine D Gocayne, Peter G Amanatides, Steven E Scherer, Peter W Li, Roger A Hoskins, Richard F Galle, et al. 2000. The genome sequence of Drosophila melanogaster. *Science*.

[2] Lee Altenberg. 1994. Evolving better representations through selective genome growth. In *First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. IEEE.

[3] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*.

[4] Martin P Bendsøe and Ole Sigmund. 1995. *Optimization of structural topology, shape, and material*. Vol. 414. Springer.

[5] Josh C Bongard and Rolf Pfeifer. 2003. Evolving complete agents using artificial ontogeny. In *Morpho-functional Machines: The new species*. Springer, 237–258.

[6] Philip Bontrager, Wending Lin, Julian Togelius, and Sebastian Risi. 2018. Deep interactive evolution. In *International Conference on Computational Intelligence in Music, Sound, Art and Design*. Springer.

[7] Philip Bontrager, Aditi Roy, Julian Togelius, Nasir Memon, and Arun Ross. 2018. Deepmasterprints: Generating masterprints for dictionary attacks via latent variable evolution. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*. IEEE, 1–9.

[8] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. 2018. Understanding disentangling in beta-VAE. *arXiv preprint arXiv:1804.03599*.

[9] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. 2018. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*.

[10] John D Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. 2018. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[11] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015. Robots that can adapt like animals. *Nature*.

[12] Antoine Cully and Yiannis Demiris. 2018. Quality and diversity optimization: A unifying modular framework. *IEEE Trans. on Evolutionary Computation*.

[13] Luis DaCosta, Alvaro Fialho, Marc Schoenauer, and Michèle Sebag. 2008. Adaptive operator selection with dynamic multi-armed bandits. In *10th annual conference on Genetic and evolutionary computation*.

[14] Kenneth De Jong. 2007. Parameter setting in EAs: a 30 year perspective. In *Parameter setting in evolutionary algorithms*. Springer.

[15] Stephane Doncieux and Jean-Arcady Meyer. 2004. Evolving modular neural networks to solve challenging control problems. In *Fourth International ICSC Symposium on engineering of intelligent systems (EIS 2004)*. ICSC Academic Press Canada.

[16] Peter Durr, Dario Floreano, and Claudio Mattiussi. 2010. Genetic representation and evolvability of modular neural controllers. *IEEE Computational Intelligence Magazine*.

[17] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural Architecture Search: A Survey. *Journal of Machine Learning Research*.

[18] Matthew C Fontaine, Scott Lee, LB Soros, Fernando De Mesentier Silva, Julian Togelius, and Amy K Hoover. 2019. Mapping hearthstone deck spaces through MAP-elites with sliding boundaries. In *Proceedings of The Genetic and Evolutionary Computation Conference*.

[19] Adam Gaier and Alexander Asteroth. 2014. Evolution of optimal control for energy-efficient transport. In *IEEE Intelligent Vehicles Symposium Proceedings*.

[20] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. 2017. Aerodynamic design exploration through surrogate-assisted illumination. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*.

[21] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. 2018. Data-efficient design exploration through surrogate-assisted illumination. *Evolutionary computation*.

[22] Adam Gaier and David Ha. 2019. Weight agnostic neural networks. In *Advances in Neural Information Processing Systems*.

[23] Aurélien Garivier and Eric Moulines. 2011. On upper-confidence bound policies for switching bandit problems. In *International Conference on Algorithmic Learning Theory*. Springer.

[24] David E Goldberg, Bradley Korb, Kalyanmoy Deb, et al. 1989. Messy genetic algorithms: Motivation, analysis, and first results. *Complex systems*.

[25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*.

[26] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*.

[27] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *International Conference on Machine Learning*.

[28] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*.

[29] Stephan Hoyer, Jascha Sohl-Dickstein, and Sam Greydanus. 2019. Neural reparameterization improves structural optimization. *arXiv preprint arXiv:1909.04240*.

[30] Hyunjik Kim and Andriy Mnih. 2018. Disentangling by Factorising. In *International Conference on Machine Learning*.

[31] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representation (ICLR)*, Yoshua Bengio and Yann LeCun (Eds.).

[32] John R Koza. [n. d.]. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University, Department of Computer Science Stanford, CA.

[33] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*.

[34] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. 2000. Constrained model predictive control: Stability and optimality. *Automatica*.

[35] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. 2019. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier.

[36] Matthew Andres Moreno, Wolfgang Banzhaf, and Charles Ofria. 2018. Learning an evolvable genotype-phenotype mapping. In *Genetic and Evolutionary Computation Conference*. ACM.

[37] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.

[38] Jean-Baptiste Mouret and Stéphane Doncieux. 2008. MENNAG: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars. *Evolutionary Intelligence*.

[39] Markus Olhofer, Yaochu Jin, and Bernhard Sendhoff. 2001. Adaptive encoding for aerodynamic shape optimization using evolution strategies. In *2001 Congress on Evolutionary Computation*. IEEE.

[40] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*.

[41] Franz Rothlauf. 2006. Representations for genetic and evolutionary algorithms. In *Representations for Genetic and Evolutionary Algorithms*. Springer.

[42] Tim Salimans, Diederik Kingma, and Max Welling. 2015. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*. 1218–1226.

[43] Eric O Scott and Jeffrey K Bassett. 2015. Learning genetic representations for classes of real-valued optimization problems. In *Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM.

[44] Eric O Scott and Kenneth A De Jong. 2018. Toward learning neural network encodings for continuous optimization problems. In *Genetic and Evolutionary Computation Conference Companion*. ACM.

[45] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* (2015).

[46] Luís F Simões, Dario Izzo, Evert Haasdijk, and Agoston Endre Eiben. 2014. Self-adaptive genotype-phenotype maps: neural networks as a meta-representation. In *International Conference on Parallel Problem Solving from Nature*. Springer.

[47] Kenneth O Stanley. 2007. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*.

[48] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. 2009. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*.

[49] Kenneth O Stanley and R Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation*.

[50] Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. 2017. Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation* 22, 4, 623–630.

[51] Vassilis Vassiliades and Jean-Baptiste Mouret. 2018. Discovering the elite hypervolume by leveraging interspecies correlation. In *Genetic and Evolutionary Computation Conference*.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[53] Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M Lucas, Adam Smith, and Sebastian Risi. 2018. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Genetic and Evolutionary Computation Conference*. ACM.

[54] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*.

SUPPLEMENTAL MATERIAL

A. Example Maps

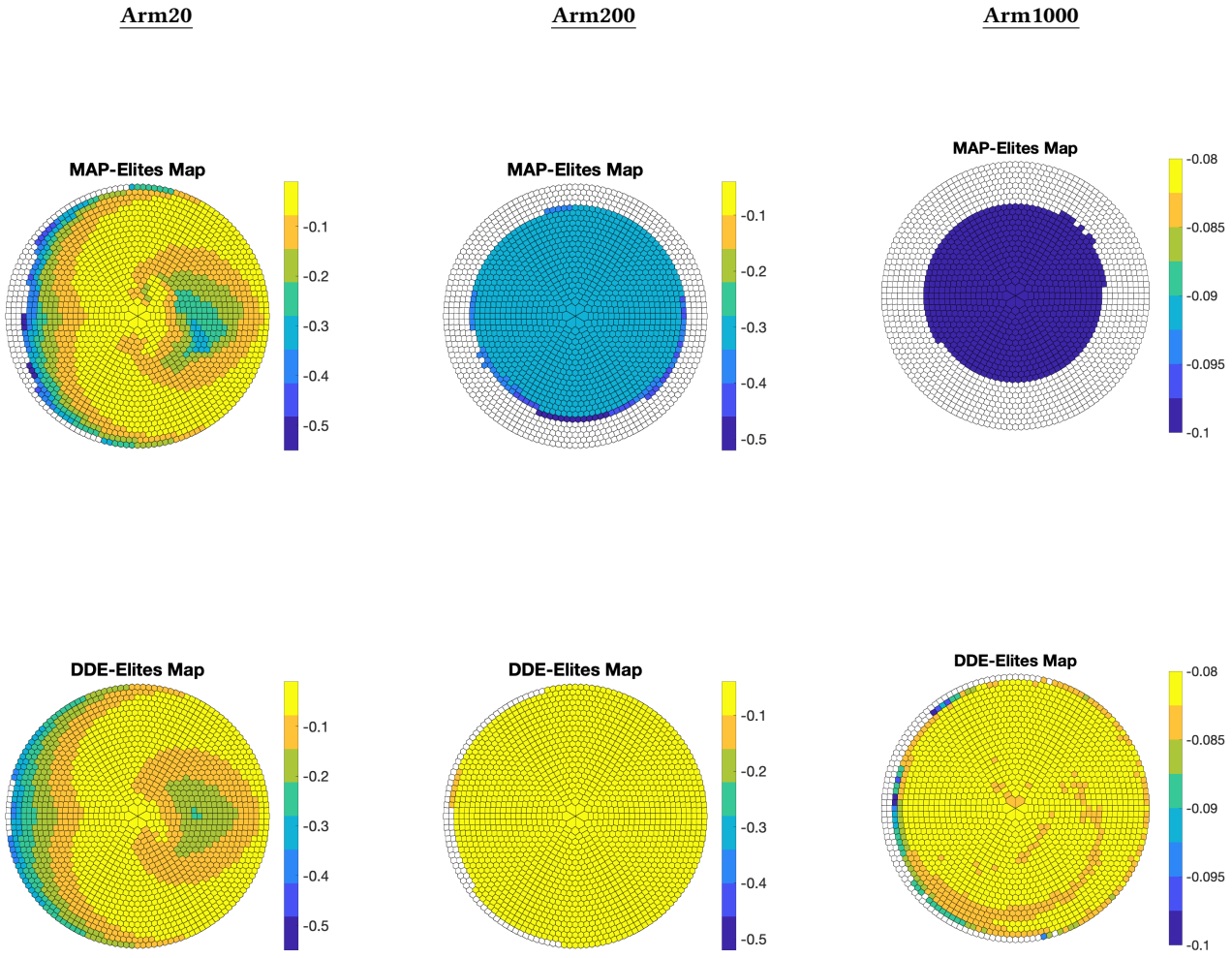

Table 2: *Example Maps*
Final archives colored by fitness value for each cell for each domain. In the 20D Arm both MAP-Elites and DDE-Elites converge on similar optimal solutions. In the 200D and 1000D Arm MAP-Elites is unable to reach the levels of performance of DDE-Elites in any region.

B. Hyperparameters of DDE Experiments

Hyperparameter	Value
Isometric Mutation Strength	0.003
Line Mutation Strength	0.1
Batch Size	100
Bandit Options,	[0.00:0.00:1.00], [0.25:0.00:0.75], [0.50:0.00:0.50], [0.75:0.00:0.25], [1.00:0.00:0.00], [0.00:0.25:0.75], [0.00:0.50:0.50], [0.00:0.75:0.25], [0.00:1.00:0.00]
Bandit Window Length	1000
Generations per VAE Training	1
Epochs per VAE Training	5
Mutation Strength when Searching DDE	0.15
Latent Vector Length [Arm20]	10
Latent Vector Length [Arm200]	32
Latent Vector Length [Arm1000]	32