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

Hagg, A.

CHAPTER 4

Efficiency

QD is able to produce many solutions with diverse behaviors. However, the algorithms in this class perform many evaluations, making them unsuitable for design problems that need computationally expensive or real world evaluation.

In this chapter, two of the problems QD algorithms have in computationally expensive domains are addressed. Although QD has been reformulated to use efficient surrogate models for the objective function by Gaier et al. (2017), expensive features are not calculated in an efficient way. The second issue arises in e.g. neural encodings that, due to structural dissimilarities and a high degree of sensitivity, cannot be modeled well.

In order to fulfill the efficiency requirements C1 ("Search should efficiently generate a diverse set of novel, functional solutions") and C2 ("Search should efficiently sample complex design spaces") from Table 2.3, the following research questions are answered in this chapter:

Research Questions

V Can we model behavioral features in a surrogate-assisted way by sampling based on optimality alone? (requirements C1 and C2)
VI Can we model neural encodings' behavior ex situ by sampling their outputs and using a behavior-kernel? (requirement C2)

The main insight highlighted in this chapter is that we can predict behavioral features from two domains, fluid dynamics and robotics, using statistical models. Using the models, the efficiency of QD can be increased.

What now follows is an extension of previous work on increasing the efficiency of QD applied to shape optimization in an expensive fluid dynamics domain.

4.1 Surrogate Modeling of Phenotypic Features

In order to create diverse solution sets in expensive domains, the feature dimensions along which QD's archive is defined have to be calculated in an efficient manner. Research question V ("Can we model behavioral features in a surrogate-assisted way by sampling based on optimality alone?") is answered in this section for a fluid dynamics optimization domain. Airflow features serve as behavioral features in this context, showing whether we can fulfill requirements C1 and C2, which were mentioned in the introduction to this chapter.

To decrease the number of expensive objective evaluations, approximative surrogate models can replace the evaluation of the objective function close to optimal solutions (see Jin (2011) for a review of this group of methods). To sample the design space effectively and efficiently for training samples, Bayesian optimization (BO) is used, which is explained in the next subsection. This section is based on earlier work by Hagg et al. (2020).

Surrogate models, oftentimes used in a Bayesian context (see Section 4.1.1) are based on similarity, the distance of candidate solutions to known examples. Similarity-based surrogate models have been used in such varied domains as: shape optimization in fluid dynamics (see Ong et al. (2003) and Daniels et al. (2018)), the discovery of new drugs (see De Grave et al. (2008)), the placement of hospital trauma centers (see Wang et al. (2016)), and even for the optimization of other machine learning methods (see Snoek et al. (2012) and Stork et al. (2017)).

4.1.1 Bayesian Optimization

BO is a strategy that is used to efficiently find optima of an expensive objective function (see Fig. 4.1). Given a prior[6] over the objective function, evidence from known samples is used to select the next best observation. The assumptions are encoded in a so-called covariance function. This decision is based on an acquisition function that balances exploration of the design space, sampling from unknown regions, and exploitation, choosing samples that are likely to perform well. The example shows an upper confidence bound (UCB) acquisition function. UCB was

[6] Priors usually consist of general assumptions on the mathematical description of the objective function. One such assumption is that a function is *smooth*: if points are close together, their function value is similar.

introduced by Auer (2002) and uses the uncertainty information from a random process to make decisions with an exploitation-versus-exploration trade-off. It is described by the function

$$\mathrm{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}),$$

which is a balance between exploitation ($\mu(\mathbf{x})$, the predicted (mean) objective value of the model), and exploration ($\sigma(\mathbf{x})$, the model's uncertainty), weighted by κ. The surrogate model becomes more accurate in optimal regions when more samples are evaluated and added.

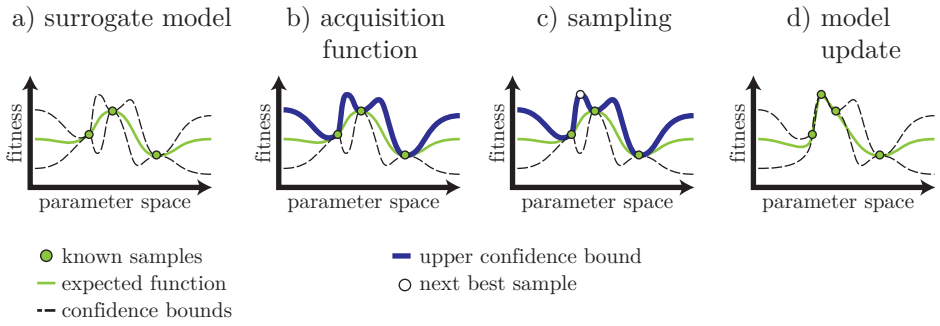

Figure 4.1: In Bayesian sampling, a statistical model serves as a surrogate for an expensive evaluation function. The model predicts function values as well as a confidence value on its prediction (a). Confidence is low depending on the distance to the next known sample in the model. The model's confidence value is included in the sampling process. An acquisition function (b), which takes into account both values, presents an 'optimistic' prediction about possible optimal locations. The sampling decision is based on a maximization of the acquisition function (c). After sampling, the model is updated, taking into account the new sampled location (d).

The most common surrogate models used are Gaussian process (GP) regression models, introduced by Rasmussen (2004). More information on GP regression is given in Appendix C.

4.1.2 Surrogate-Assisted Quality Diversity

QD became applicable to expensive optimization problems after the introduction of surrogate-assisted illumination (SAIL) by Gaier et al. (2017). In this Bayesian interpretation of QD, an archive is created based on UCB sampling. Surrogate

models, like GP models, cannot easily be based on phenotypes or behavior due to the high dimensionality of the artifacts. Furthermore, the models have to be able to deal with a large diversity of solutions, for example using a hierarchical decomposition of the parameter space (see Hagg (2017) and Stein et al. (2015)).

In SAIL, a GP regression model predicts the performance of new solutions based on the genetic distance to previous examples, serving as a surrogate (replacement) for expensive evaluation. The distance is modeled using the squared exponential (covariance) function, which has two hyperparameters: the length scale (or sphere of influence) and the signal variance, which are found by minimizing the negative log-likelihood of the process.

Algorithm 7 SAIL and its extension SPHEN

1: **procedure** SAIL/SPHEN($f()$,$p()$)
2: Set $budget, \sigma$
3: Set $\mathcal{X}, \mathbf{p}, \mathcal{F} \leftarrow \emptyset$
4: $\mathcal{X}' \leftarrow$ SOBOL($dim(\mathcal{X})$)
5: **while** $|\mathcal{X}| < budget$ **do**
6: $(\mathcal{F}', \mathbf{p}') \leftarrow$ EVALUATE($\mathcal{X}', f(), p()$)
7: $(\mathcal{X}, \mathbf{p}, \mathcal{F}) \leftarrow (\mathcal{X} \cup \mathcal{X}', \mathbf{p} \cup \mathbf{p}', \mathcal{F} \cup \mathcal{F}')$
8: $(\mathbf{M}_p, \mathbf{M}_f) \leftarrow$ TRAIN($\mathcal{X}, \mathbf{p}, \mathcal{F}$)
9: $f_s() \leftarrow$ PREDICT($\mathcal{X}, \mathbf{M}_f$)
10: $p_s() \leftarrow$ UCB($\mathcal{X}, 20, \mathbf{M}_p$)
11: $\mathcal{A}_a \leftarrow$ MAP-ELITES($\mathcal{X}, f()/f_s(), p_s(), \sigma$)
12: $\mathcal{X}' \leftarrow$ SELECT($\mathcal{A}_a$)
13: **end while**
14: $p_s() \leftarrow$ UCB($\mathcal{X}, 0, \mathbf{M}_p$)
15: $\mathcal{A} \leftarrow$ MAP-ELITES($\mathcal{X}, f()/f_s(), p_s(), \sigma$)
16: **end procedure**

SAIL is explained in more detail in Alg. 7, as it is both used and extended in this work. The extension (red elements) is explained in the next section.

First, a space-filling sequence is used to fill the initial population (line 4). Then, as long as the evaluation budget is not fully consumed, the population is evaluated to extract its performance and feature values (line 6). The features are calculated based on the expressed phenotype, or behavior, of a solution. A surrogate model, usually a GP regression model, is trained to predict new solutions' performance

values (line 8). The surrogate-assisted performance function $p_s()$ consists of a UCB acquisition function (see Section 4.1.1) with a large exploration factor $\kappa = 20$. MAP-Elites (see Alg. 2) generates an acquisition archive based on the populations' genomes (the search space), the feature function $f()$, and the surrogate-predicted performance function $p_s()$ in line 10. After MAP-Elites fills the acquisition archive which contains 'optimistic' solution candidates, a random selection of those candidates (line 12) is analyzed in the expensive evaluation function to form additional training samples for the GP model.

This loop continues until the evaluation budget is exhausted. Then, the UCB exploration factor κ is set to 0 in a final MAP-Elites run to create an archive that now contains a diverse set of solutions that is predicted to be high-performing (line 15). SAIL needs a budget orders of magnitudes smaller than MAP-Elites because it can exploit the surrogate model without 'wasting' samples. SAIL, however, is constrained to features that are cheap to calculate, like shape features that can be determined without running the expensive evaluation.

With SAIL it became possible to use performance functions of expensive optimization domains. But the strength of QD, to perform niching based on behavior, cannot be applied when determining those *behaviors* is expensive. Phenotypic features describe phenomena that can be related to complex domains, like behavioral robotics, mechanical systems, or computational fluid dynamics (CFD). Only when we can predict those expensive features efficiently, the road to productive phenotypic niching is opened up. Only then can we generate phenotypically diverse solution sets in engineering.

4.1.3 Surrogate-Assisted Phenotypic Niching

To be able to handle expensive features, surrogate-assisted phenotypic niching (SPHEN) is introduced (see Fig. 4.2, Alg. 7 including red elements). By building on the insight that replacing the performance function with a surrogate model decreases the necessary evaluation budget, the exact features are replaced with surrogate models as well.

The initial training sample set $\mathcal{X}'$, used to form the first seeds of the acquisition map, is produced by a Sobol sequence in the design space (see Fig. 4.2a and line 4 in Alg. 7). Due to the lack of prior knowledge in black-box optimization, using space-filling sequences has become a standard method to ensure a good coverage

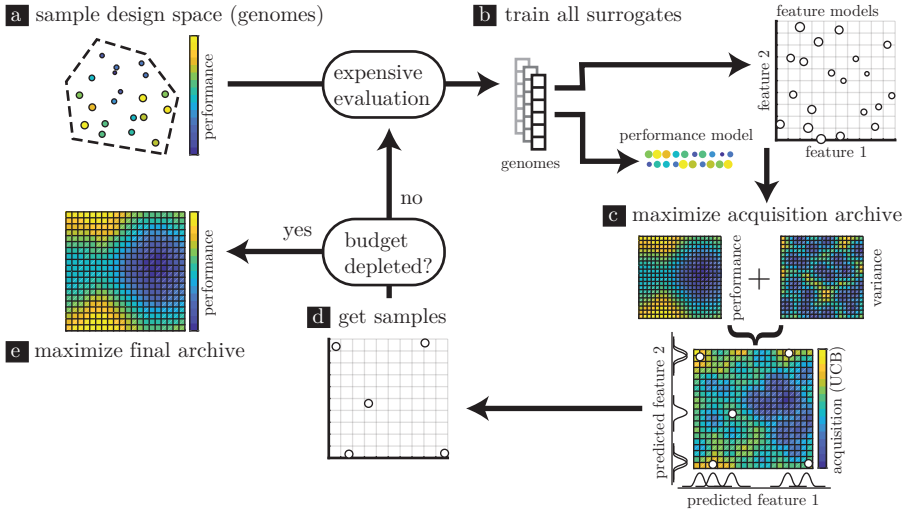

Figure 4.2: Surrogate-assisted Phenotypic Niching. An initial sample set (a) is evaluated. Surrogates are trained to predict performance and features (b). Surrogate-assisted MAP-Elites is used to produce an acquisition map, balancing exploitation and exploration with the UCB of the performance model. Feature models predict the niche of new individuals (c). New samples are selected from the acquisition archive (d). After the evaluation budget is depleted, the surrogate models are used to generate the final archive, ignoring model confidence (e).

of the search domain. The initial set is evaluated, for example in a computational fluid dynamics simulator (line 6).

Performance values ($\mathbf{p}$) and phenotypic features ($\mathcal{F}$) of those samples are derived from the results, or, in the case of simpler non-behavioral features, from the solutions' expression or shape themselves. The key issue here is to check the range of the initial set's features. Since it is unknown what part of the phenotypic space will be discovered in the process, the initial set's feature values only give us a first hint of the reachable space. A space-filling sampling technique used in the design space is not necessarily space-filling in feature space.

After collecting performance and feature values, the surrogate models are trained (see Fig. 4.2b and line 8). The GP models limit the number of samples to around 1000, as the training and prediction becomes quite expensive. A squared exponential covariance function is used and the (constant) mean function is set to the training samples' mean value. The covariance function's hyperparameters, length scale and

signal variance, are deduced using the GP toolbox GPML's (see Rasmussen and Nickisch (2010)) conjugate gradients based minimization method, which is run for 1000 iterations.

Importantly, MAP-Elites (Alg. 2) does not receive feature and performance functions directly, but instead, the feature model $\mathbf{M}_f$ predicts feature locations through the predicted feature function $f_s()$ and the performance model $\mathbf{M}_p$ is used as part of a predicted performance function $p_s()$ using UCB with $\kappa = 20$ (lines 9 and 10). MAP-Elites creates the acquisition map, a version of the archive that contains an 'optimistic' set of proposed samples. It does so by optimizing the UCB of the performance model only. Feature models assign samples to their niches (see Fig. 4.2c).

Notably, in the surrogate-assisted version of MAP-Elites the confidence of feature models is not taken into account. The reasoning behind this is that, although the search takes place in a high-dimensional space, QD only has to find the *elite hypervolume* (see Vassiliades and Mouret (2018)), or *prototypes* (see Hagg et al. (2018)), the regions consisting of high-performing solutions. Only the performance function can guide the search towards the hypervolume. Taking into account the feature models' confidence intervals adds unnecessary complexity to the modeling problem. SPHEN's goal is to be able to only predict features for high-performing solutions, so we let feature learning 'piggyback' on this search.

After having received back the acquisition archive, a selection of samples from the acquisition archive is taken using a space-filling algorithm like Sobol (see Fig. 4.2d). The samples are then evaluated to continue training the surrogate models. This process iterates as long as the evaluation budget is not depleted. Finally, MAP-Elites is used to create a prediction map, ignoring the models' confidence altogether (see Fig. 4.2e) by setting $\kappa = 0$, which is filled with diverse, high-performing solutions (Alg. 7, line 15).

SPHEN extends SAIL by replacing the direct calculation of phenotypic features with the predictions of a surrogate model. Before SPHEN is applied to an expensive CFD domain, its performance is compared to MAP-Elites and SAIL in a simpler, inexpensive domain.

4.1.4 Quantitative Comparison

SPHEN is compared to SAIL and MAP-Elites in terms of how many precise function and feature values have to be calculated. It is important to evaluate how accurate the feature models are when trained with a performance-based acquisition function. Only then is SPHEN applied to an expensive domain. To be able to calculate all performance and feature values, we optimize the same free-form deformed, eight-sided polygons as were shown in Fig. 3.11.

QD optimization is run without (MAP-Elites) and with surrogate model(s) (SAIL, SPHEN) on the polygon domain. This way all ground truth performance and feature values can be calculated in a feasible amount of time. The shape features are expected to be easier to learn than the flow features of the airflow domain. SPHEN is expected to perform somewhere between SAIL and MAP-Elites, as it has the advantage of using a surrogate model but also has to predict two phenotypic features. However, since the ultimate goal is to be able to use QD on expensive features, SPHEN will be the only possibility to achieve this.

Budgets Comparing MAP-Elites, SAIL and SPHEN in a fair manner is not straightforward, although the surrogate-assisted algorithms have a very similar structure. The budget, the number of precise function evaluations, can be defined in two manners. Because SAIL was introduced to reduce the number of function evaluations for expensive performance functions, it was compared with MAP-Elites based on the number of precise performance evaluations (PE) by Gaier et al. (2017). However, due to the introduction of expensive feature evaluations, the number of precise phenotypic feature evaluations (PFE) has to be taken into account. Here, SAIL will quickly consume the budget.

The (maximum) PE and PFE budgets of all algorithms used in the following comparison are listed in Table 4.1. The PE budget is the same for all algorithms, except MAP-Elites, as it was already shown that it needs many more PE than SAIL (see Gaier et al. (2017)). MAP-Elites receives a budget of 65,536 generations multiplied with the number of children per generation. The choice to restrict the PE instead of the PFE budget makes it easier to compare results to previous work.

Table 4.1: Maximum budget for MAP-Elites, SAIL, budget-reduced SAIL^{r} and SPHEN in experiments.

Parameter	MAP-Elites	SAIL	SAIL^{r}	SPHEN
# MAP-Elites generations	4096	1024	64	1024
# MAP-Elites children	16	32	16	32
# acquisition iterations	-	64	64	64
budget per iteration	-	16	16	16
total PE	65,536	1024	1024	1024
total PFE	65,536	2,098,192	65,536	1024

The fixed maximum PE budgets yield the PFE budgets shown in the last row of Table 4.1. It is easy to see how SAIL is intractible when using expensive features, as it needs 2,098,192 PFE to reach the GP model's maximum number of 1024 samples. By reducing the budget (SAIL^{r}) to at most the number of PFE used by MAP-Elites, we can both reestablish a more feasible number of PFE as well as maintain some kind of comparability. This is accomplished by reducing the number of generations and children in the internal (MAP-Elites) loop in SAIL^{r}. It is not useful to reduce the budgets even more because for a 'fair' comparison, the number of PFE within the inner loop would have to be reduced to $1024/64 = 16$, which would amount to a single MAP-Elites generation.

A comparison is now possible by showing the results for all PFE budgets (16, 1024, 65,536, and 2,098,192), as far as they are practically attainable.

Parameterization The initial sample set of 16 examples as well as the selection of new samples (16 in every iteration) is created using a pseudo-random Sobol sequence. In QD, the phenotypic archive allows us to find many diverse solutions but in order to fill the archive, it needs to be ensured that we can sweep through the searched genetic space multiple times. The σ value is therefore quite high. The mutation operator adds a value drawn from a Gaussian distribution with $\sigma = 10\%$ of the parameter range.

Misclassification in SPHEN Due to the expected inaccuracy of the feature models, solutions will be misclassified. Misclassification will decrease the accuracy of the niching mechanism. Fig. 4.3 shows a niching archive at a resolution of 32x32 and the true performance and feature archive. Holes appear due to misclassification,

which is why SPHEN is trained on a higher resolution map. The archive is a reduction to a resolution of 16x16. Most niches are now filled. In this experiment all archives have a resolution of 16x16 solutions.

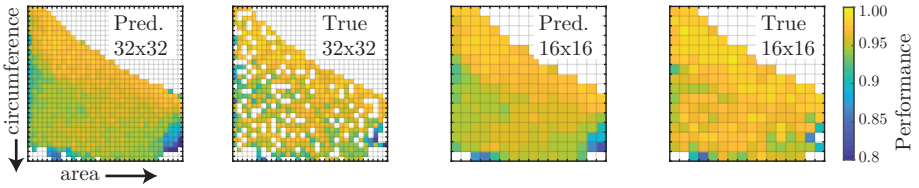

Figure 4.3: Predicted and true SPHEN maps on symmetry domain, trained in 32x32 resolution (left), then reduced to 16x16 resolution to remove holes (right).

Results The mean number of filled archive niches and performance values for five replicates are shown in Fig. 4.4. SAIL and SPHEN find about the same number of solutions using the same number of PE. Notably, the mean performance of SPHEN's solutions is higher than that of SAIL.

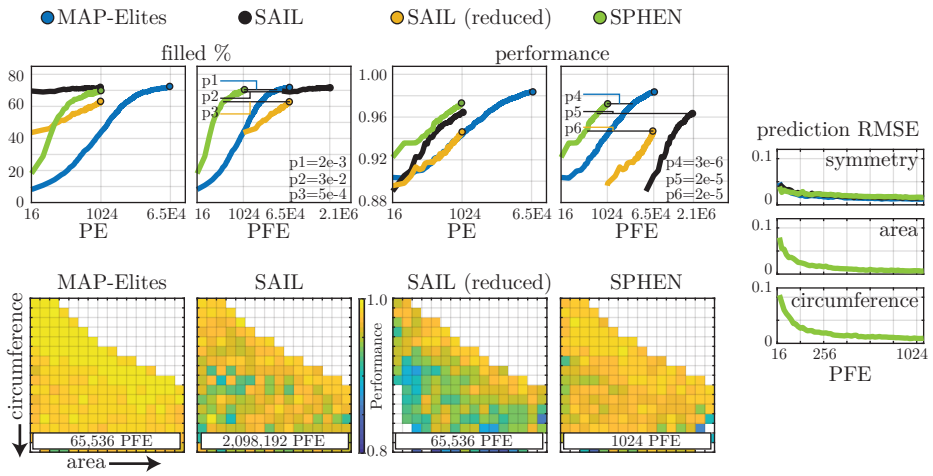

Figure 4.4: Comparison of MAP-Elites, SAIL and SPHEN based on performance evaluations (PE) and performance/feature evaluations (PFE). Experiments were repeated five times to produce the mean percentage of the archive's filled and mean performance values. Prediction errors are included on the right and example archives at the bottom. The experiments include SAIL with a budget reduced to the number of PFE used in MAP-Elites.

In domains with expensive feature evaluations, the performance and PFE need to be taken into account. Due to the number of necessary feature evaluations, SAIL needs more than two million PFE to perform almost as well as SPHEN, which only needs 1024 – over three orders of magnitude less and still more than an order of magnitude less than MAP-Elites. Since in expensive real world optimization problems we cannot expect to run more than about 1000 function evaluations, due to the infeasibly large computational investment, the efficiency gain of SPHEN is substantial. If the number of PFE of SAIL is lowered to the same budget as MAP-Elites and given more time to search the iteratively improving surrogate model before running out of the budget of 65,536 PFE (see Table 4.1), SAIL still takes a big hit. It is not able to balance out quality and diversity. The example prediction maps are labeled with the number of PFE necessary to achieve those maps. Although new training examples are not sampled to improve the feature models specifically, their root mean square error (RMSE) ended up at 0.012 and 0.016 respectively. Finally, SPHEN is compared to the three alternative configurations on the null hypothesis that they need the same number of PFE to reach an equally filled archive or equal performance. Significance levels, calculated using a two-sample t-test, are shown in Fig. 4.4. In all cases, the null hypothesis is improbable ($p < 0.05$), although for the comparison of filled levels to SAIL it is rejected with less certainty.

The acquisition function of SAIL needs no adjustments. SPHEN and SAIL search for the same elite hypervolume, which is only determined by the performance function.

4.1.5 Use Case: Wind Nuisance in Architecture

The previous section showed that both performance as well as feature models can be predicted using an optimality-based acquisition function. The implementation of an efficient divergent search method, SPHEN, produces a diverse and high-performing set of solutions with the help of simple statistical models. To put SPHEN to a real test, however, at least one of the features should describe the behavior of a solution in an expensive domain. Fluid dynamics is commonly used as a use case in optimization studies. The problem of wind nuisance in the built environment combines this expensive domain with a typical setting for human-computer co-creation, where not all preferences can be formalized. Co-creating

diverse solutions in this domain can uncover a large potential of solutions that, in real world design processes, lead to better alternatives to current design.

The polygon domain that was described in Section 3.3.3 is used again (see Fig. 3.11). One of the features is replaced by wind nuisance. It is defined in building norms in NEN 8100 (2006), described by Janssen et al. (2013), and uses the wind amplification factor measured in standardized environments, with respect to the hourly mean wind speed. In a simplified two-dimensional setup, we translate this problem to that of minimizing the maximum airflow speed (u_{max}). The performance metric is defined as the inverse over the normalized maximum velocity in the flow: $p(x) = \frac{2}{(1+u_{max}(x))} - 1$ and is calculated with a fixed flow input speed in the simulation. The value for u_{max} only needs to be kept within a *nuisance threshold*, which is set to $u_{max} \leq 0.12$.

The closed form encoding from the polygon domain is used to produce two-dimensional shapes that are then placed into a CFD simulation. To put emphasis on the architectural nature of the domain, two features are used: area and airflow turbulence. The chaotic behavior of turbulence provokes oscillations around a mean flow velocity, which influences the maximum flow velocity.

Both features are not optimization goals. Rather, it is to be analyzed how the size of the area and turbulence are related to each other, under the condition of keeping the flow velocity low. Shapes should be produced that are combinations between their appearance (small to large) and their effect on the flow (low to high turbulence). Concretely, at the lowest and highest values of area and turbulence, regular intuitive shapes should be generated by the algorithm such as slim arrow-like shapes for low turbulence and area, or regular polygons for high turbulence and area. However, for area/turbulence combinations in between, the design of the shape is not unique and will possibly differ from intuition.

Lattice Boltzmann Method Viscous fluid dynamics systems are described by the Navier-Stokes equations, which is a set of partial differential equations. The Lattice Boltzmann method (LBM) is an established tool for the simulation of fluid dynamics (see Krüger et al. (2017)). Instead of directly solving the Navier-Stokes equations, the method operates a stream and collide algorithm of particle distributions derived from the Boltzmann equation. In this contribution, LBM is used on a two-dimensional grid with the usual lattice of nine discrete particle velocities. At the inlets and outlets, the distribution function values are set to

equilibrium according to the flow velocity. The full bounce-back boundary condition is used at the solid grid points corresponding to the polygon. Although there are more sophisticated approaches for the boundaries, this configuration is stable throughout all simulations. In addition, the bounce-back boundary condition is flexible, as the boundary algorithm is purely local with respect to the grid points.

As an extension of the Bhatnagar-Gross-Krook (BGK) collision model by Bhatnagar et al. (1954), a Smagorinsky subgrid model (see Gaedtke et al. (2018)) is used to account for the under-resolved flow in the present configuration. A more detailed description of the underlying mechanisms can be found in Krüger et al. (2017). The results of the two-dimensional domain do not entirely coincide with results that will be found in three dimensions, caused by the difference in turbulent energy transport, which was shown by Tennekes (1978).

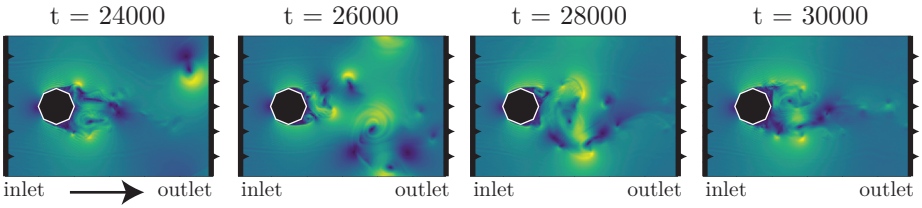

Figure 4.5: Airflow around a circular polygon shape at four different time steps.

The simulation domain consists of 300×200 grid points. A bitmap representation of the polygon is placed into this domain, occupying up to 64×64 grid points. As the Lattice Boltzmann method is a solver of weakly compressible flows, it is necessary to specify a Mach number (0.075), a compromise between computation time and accuracy. The Reynolds number is $Re = 10,000$ with respect to the largest possible extent of the polygon. For the actual computation, the software package *Lettuce* by Krämer et al. (2020) is used, which is based on the PyTorch framework (see Paszke et al. (2019)), allowing easy access to graphics processing unit (GPU) functionality. The fluid dynamics experiment was run on a cluster with four GPU nodes, each simulation taking ten minutes. Fig. 4.5 shows the airflow around a circular polygon at four different, consecutive time steps. Brighter colors represent higher magnitudes of airflow velocity. Throughout the 100,000 time steps of the simulation, maximum velocity and enstrophy are measured. The enstrophy, a measure for the turbulent energy dissipation in the system with respect to the

resolved flow quantities (see Gassner and Beck (2013) and Krämer et al. (2019)), increases as turbulence intensity increases in the regarded volume.

Validation and Prediction of Flow Features The maximum velocity u_{max} and enstrophy E are measured every 50 steps. A running average is employed over the last 50,000 time steps. To test whether the simulations indeed converge to a stable value, they are tested with different shapes (nine varied-size circles and nine deformed star shapes) and calculate the moving average of the enstrophy values, which is plotted in Fig. 4.6. The value converges to the final feature value (red).

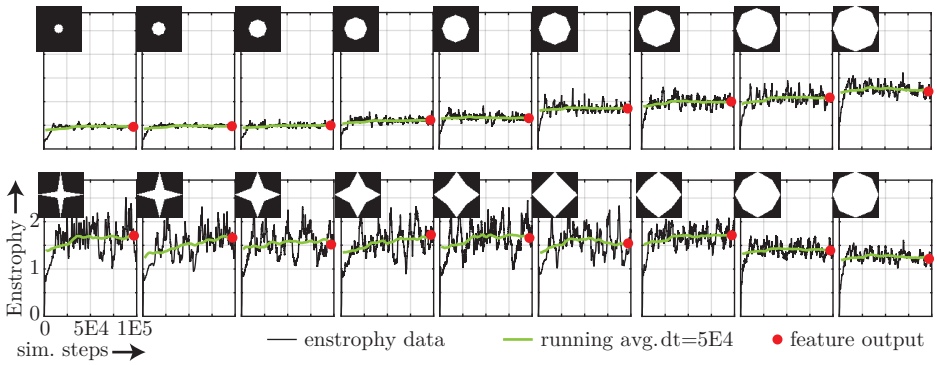

Figure 4.6: Enstrophy values during simulation of circles and stars. The running average of the last 50,000 time steps converges to the final feature output.

The two small shape sets are used to validate the two measures. Increasing the radius of the circles set should lead to higher u_{max} and E, as more air is displaced by the larger shapes. The stars set is expected to have larger u_{max} and E for the more irregular shapes. This is confirmed in Fig. 4.7.

Next, the prediction accuracy for the flow feature values is evaluated using a GP model. Although GP models are often called 'parameter free', this is not entirely accurate. The initial guess for the hyperparameter's values, before minimization of the negative log-likelihood of the model takes place, can have large effects on the accuracy of the model. The log-likelihood landscape can contain local optima. A grid search is performed on the initial guesses for length scale and signal variance. Using leave-one-out cross validation, GP models are trained on all but one shape, after which the accuracy is measured using the mean absolute percentage error

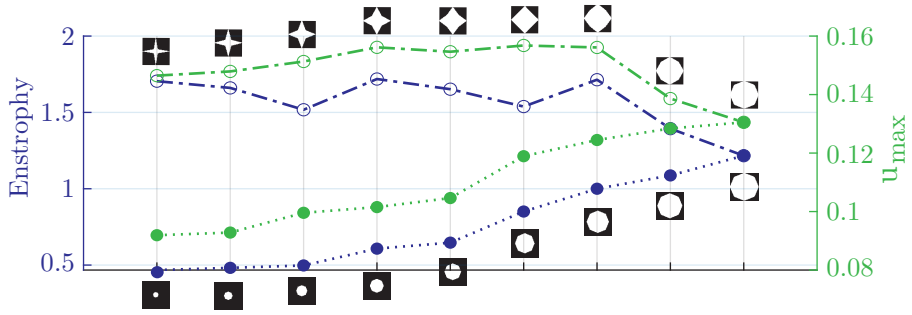

Figure 4.7: Enstrophy and maximum velocity of circles and stars.

(MAPE), giving a good idea about the magnitude of the prediction error. The process is repeated until all examples were part of the test set once. The MAPE on u_{max} was 2.4% for both sets. The enstrophy was harder to model, at 4.9% and 10.3% for the respective sets, but still giving us confidence that these two small hypervolumes can be modeled.

Resulting Shapes The objective is to find a diverse set of airflows using a behavioral feature, turbulence, and one shape feature, the surface area of the polygon. This should answer the question how the size of the area and turbulence are related to each other and which shapes do not pass the wind nuisance threshold. The same budget for SPHEN is used as listed in Table 4.1, with the exception of allowing 4096 generations in the prediction phase. The enstrophy and velocity are normalized between 0 and 1 using a predetermined value range of $E \in [0.15, 1.1]$ and $u_{max} \in [0.05, 0.20]$.

The resulting archive in Fig. 4.8 shows that turbulence and surface area tend to increase the mean maximum airflow velocity, as expected. A small selection of airflows is shown in detail. Due to the chaotic evolution of turbulent and transient flows, a static snapshot of the velocity field provides only limited information about the flow structures. Therefore, dynamic mode decomposition (DMD) is used to extract and visualize coherent structures and patterns over time from the flow field (see Demo et al. (2018) and Schmid (2010)).

Especially those shapes at the extrema of area and turbulence align with the aerodynamic expectations. At low turbulence intensity, the shapes tend to be slim and long with respect to the flow direction (shapes A and B). High turbulence

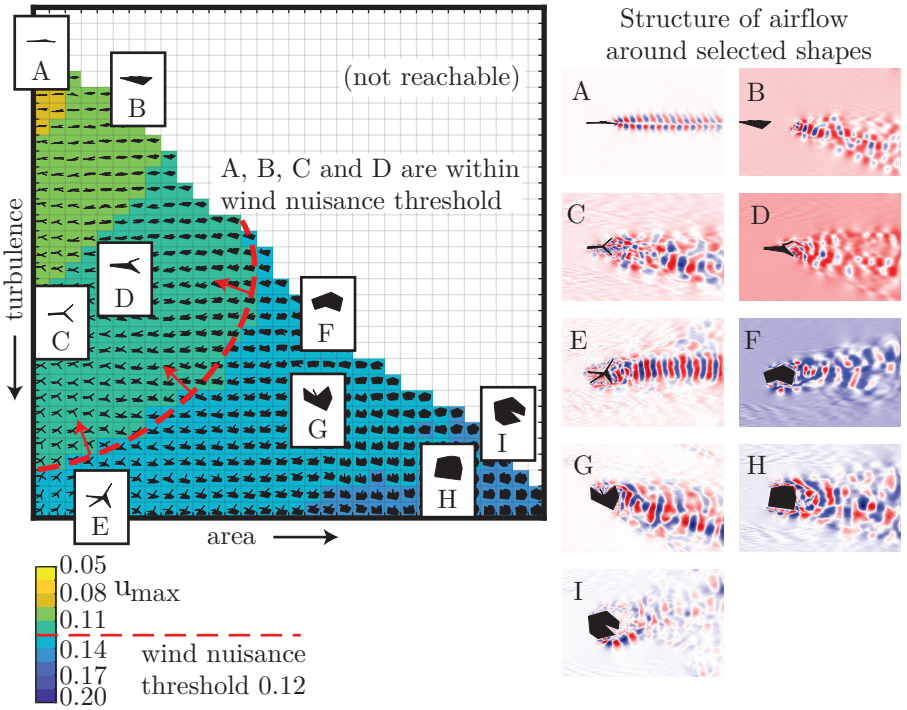

Figure 4.8: A diversity of shapes and airflows that shows which designs conform to the wind nuisance threshold. The dominant DMD mode shows the structure of the airflow around nine selected shapes. A, B, C and D are within the wind nuisance threshold.

levels at small shape areas are achieved if the shapes are oriented perpendicularly to the flow (shape E). Pentagons or hexagons evoke high turbulence levels at large areas (shapes H and I). However, impressively, there is an enormous variety of nuances in between these extrema with non-intuitive shapes, enabling the designer to determine a shape for given flow parameters down to a lower turbulence bound for each area value. Furthermore, the algorithm also suggests known tricks to manipulate the flow. Side arms are an appropriate measure to vary the turbulence intensity in the wake (shapes C, D, E, and G). Indentations or curved shapes redirect the flow and extract kinetic energy similar to turbine blades (see Dorschner et al. (2017)), which can be observed in shape D. Conclusively, for the highest and lowest area and turbulence values, SPHEN matches the expectations, while for the shapes in between, SPHEN exceeds expectations by introducing unusual shape nuances, which encourage further investigation.

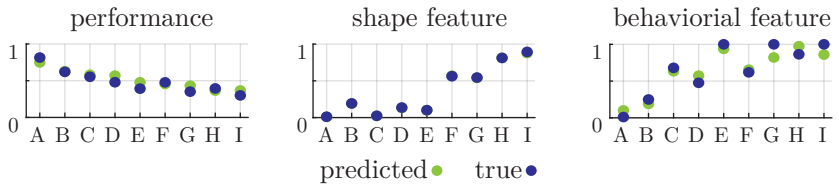

Figure 4.9: RMSE of performance and feature models.

The accuracy of the surrogate models is shown in Fig. 4.9. The RMSE of the models is 0.06, 0.01 and 0.10, respectively. A non-parametric hypothesis test is performed to determine the rank correlation between pair-wise comparisons using either the predicted or true values. In evolutionary computation, only the rank correlation is important, because values are compared only pair-wise within the evolutionary process.

The rank correlation coefficient by Kendall and Gibbons (1990) is used (τ). In contrast to Pearson correlation, this measure only considers ordinal correlation, i.e., the ranks of two compared sets of samples. Kendall correlation is therefore a good measure to estimate the accuracy of a model when used in rank-based evolutionary optimization methods. The coefficient τ amounts to one when all comparisons lead to the same outcome and to minus one when all comparisons lead to the exact opposite outcome. If in this experiment, τ equals one, the surrogate models produce the same outcome as using the true values. In this case, τ amounts to 0.78, 1.00 and 0.73. The models are therefore accurate enough for rank-based selection.

In this real world optimization case, SPHEN is able to produce a diverse set of solutions. The surrogate models are able to both predict performance as well as diversity.

4.1.6 Conclusions

This section gave evidence for a positive answer to research question V (“Can we model behavioral features in a surrogate-assisted way by sampling based on optimality alone”), fulfilling requirements C1 (“Search should efficiently generate a diverse set of novel, functional solutions”) and C2 (“Search should efficiently sample complex design spaces”). Behavioral surrogate models can be trained by

piggybacking upon an acquisition function that selects samples based on optimality alone.

In the polygon domain, both surrogate-assisted algorithms are able to find a large variety of solutions. When features do not have to be modeled, they show similar performance, although SAIL converges much sooner. However, when taking into account the number of feature evaluations, SPHEN clearly outperforms SAIL as well as MAP-Elites. Modeling features does not lower the performance of a prediction map. In terms of solution performance, both surrogate-assisted algorithms are outperformed by MAP-Elites in the simple domain, but SPHEN clearly beats MAP-Elites by requiring less evaluations. The feature models become more accurate even when sampling only to improve the performance model.

When designing diverse airflows, one SPHEN run took 23 hours, producing 494 different flow profiles. With SAIL, obtaining the same result would have taken over five years, which is not feasible in practice. Although MAP-Elites outperformed SAIL in the simple polygon domain, and might have outperformed it in the airflow domain as well, it still would have taken two months to calculate with uncertain results. Fig. 4.8 shows the structure in the airflows that can appear in this problem domain. Variations (area) of the object we want to design as well as their effect on the environment (turbulence) can be efficiently created. Even when only using two phenotypic features, the nuances between the variations give us an idea which shapes do not pass the wind nuisance threshold and which ones do. With this we could continue the design process based on our new intuition.

Expensive features can be efficiently predicted. As such, phenotypically diverse solution sets can be efficiently generated, even in expensive domains such as CFD.

4.2 Surrogate Modeling of Neural Behaviors

Evolutionary encodings can have a non-closed form or even be indirect, including e.g. neural representations. Research question VI ("Can we model neural encodings' behavior ex situ by sampling their outputs and using a behavior-kernel?") is answered to fulfill requirement C2 ("Search should efficiently sample complex design spaces") for neural representations.

Besides having to deal with expensive numerical fitness evaluations, neural representations, commonly used in the robotics domain, have the need to run physics-enabled simulations or real world experiments. Iterative optimization requires many of these evaluations to reach a satisfactory solution.

A prerequisite for similarity-based surrogate models is that a distance metric is defined for the encoding of a solution. Surrogate models are therefore usually applied to solution representations that encode a fixed number of parameters. Recently, more complex non-closed form encodings have been developed that do not have a constant input space. A prime example of such encodings are compositional pattern producing network (CPPN) (see Stanley (2006)), that encodes complex shapes or behaviors indirectly. In neuroevolution (see Stanley and Miikkulainen (2002)), the topology of neural networks can be evolved. Genetic programming (see Koza (1994)), evolves the topology of graphs, trees representing computer code, or mathematical equations. The non-uniform input space of these encodings frustrates typical ways of measuring distance as the dimensionality and even the meaning of these dimensions varies from one individual to the next (see Figure 4.10).

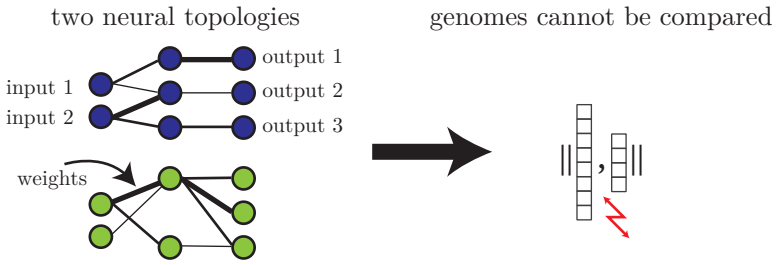

Figure 4.10: Two networks with different topologies cannot be compared based on their genomes.

A second problem arises when the quality of a solution depends on interaction with its environment. This behavior might vary greatly even if the parameterization of the encoding is only minimally changed. If a similarity-based model would be trained to predict the quality of such an encoding, a parameterization that is close to a training example would be assigned a similar fitness, although its actual fitness might be very different.

To enable surrogate-assisted optimization of neural encodings, the idea of measuring distances not of the encoding, the genome, but rather of their expression, the

phenotype, is investigated in this section (based on Hagg et al. (2019)). The phenotype may include morphological as well as behavioral aspects and can therefore give us more information about how similar two individual solutions are than the genome alone (see Stork et al. (2019)). The main insights are that (1) regardless of a network's internal composition, the size of the output in relation to the input is constant, and (2) the relation between input and output describes the behavior, and is thus a useful proxy for similarity between networks. To measure the difference in the behavior of two networks, the same input sequence is used on the networks. The difference in the output sequence can then be compared using a standard metric, like Euclidean distance.

By using randomly selected, but fixed input sequences, actual simulations are not necessary to get the output sequence. Instead, the input/output relation is sampled and uses the ad hoc difference in the output sequences of two individuals to measure their distance. This distance measure can now be used to build a similarity-based surrogate model.

4.2.1 Related Distance Kernels

Similar to phenotypic distances, semantic distances are used in genetic programming by Moraglio et al. (2012). These semantic distances can be defined as a distance of the outputs of population members, determined with the same measure that is used in the fitness function. Semantic distances are applicable where the fitness function can be computed as a distance between the optimal target vector and the candidate outputs, such as in supervised classification or symbolic regression. In these cases, the semantic distance has a fitness distance correlation of exactly one and can be utilized to construct specific mutation and crossover operators, rendering the problem uni-modal.

Phenotypic distances have also been employed in a surrogate modeling context. Hildebrandt and Branke (2015) suggested a phenotypic distance for dynamic job shop scheduling problems. Their definition of phenotypic distance compares individual solutions according to the results of evolved dispatching rules on a small set of test situations. Unlike semantic distances, their phenotypic distance is not identical to the measure used in the actual fitness function. This is necessary in the context of surrogate modeling for expensive fitness functions: if they are expensive to compute, it would also be expensive to use the same evaluation to compute a

distance between candidates. Such an approach would render the construction of the surrogate model itself expensive, which defeats its very purpose.

Zaefferer et al. (2018) compared different genetic and phenotypic distances for surrogate models in symbolic regression. Here, the underlying measure is not identical to that used in the fitness function. Specifically, the fitness function considers fixed coefficients in the symbolic expression. These coefficients are otherwise optimized during an actual fitness evaluation, which may become costly. In both of these cases, the phenotypic distance was reported to yield better results than genetic distances (see Hildebrandt and Branke (2015); Zaefferer et al. (2018)).

Doncieux and Mouret (2010) discussed the use of behavioral similarity in evolutionary robotics to employ a diversity measure for a multiobjective optimization approach. They compared different distances based on the states, outputs and trajectories given concrete robot tasks. The authors outlined that using these behavioral distances as a second objective in multi-objective optimization is able to enhance the overall performance.

A first approach utilizing a surrogate model for evolving neural networks given complex control tasks was discussed by Gaier et al. (2018). An evolutionary algorithm was combined with a surrogate model based on a hereditary distance, which is defined in the context of neuroevolution of augmenting topologies (NEAT) as *compatibility distance*. The approach is able to significantly improve the evaluation efficiency. Stork et al. (2019) also investigated surrogate models for neuroevolution. They examined simple classification tasks and compared a phenotypic distance measure to genetic distances in surrogate-assisted Cartesian genetic programming (CGP) (see Miller et al. (1997)). The use of a phenotypic distance was shown to be very promising in terms of evaluation efficiency.

4.2.2 Sampled Phenotypic Distance

The networks that are investigated here are results of optimization runs with fixed network topologies. This allows a comparison of the efficiency of models based on both genetic and phenotypic distance measures. To define a genetic distance the vector of weights of the neural networks is considered. Let $\mathbf{w} = [w_1, w_2, ..., w_j]$ be a weight vector of length j, then the genetic distance is calculated by the

related weights of two samples: $d(\mathbf{w}, \mathbf{w}')$, with d being an appropriate distance metric.

The disadvantage of genetic distance measures is their lack of applicability when changing topologies are considered. If in these cases no clear concept to compare genetic changes exists (as applied by Gaier et al. (2018)), the genetic distance comparison is difficult, misleading and even destructive (see Stork et al. (2019); Doncieux and Mouret (2010)). The ability to compare non-uniform topologies makes phenotypic distances a valuable technique, especially in cases when typical distances are not a viable option.

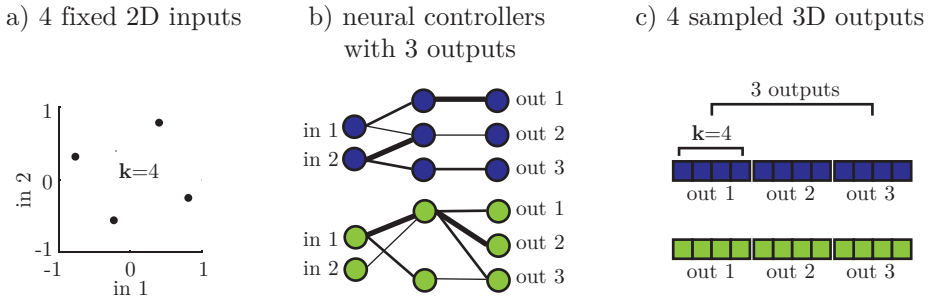

Figure 4.11: Sampling the phenotype with fixed inputs (a) to compare two individual networks (b). The fixed input strings are inserted into the networks to calculate the phd outputs (c).

The phenotype displays the behavior of a neural network given a certain set of inputs. For example, in the case of neural networks used as controllers for robots the phenotype can be defined as the control commands that are issued in response to different sensor inputs. Phenotypic distance is defined as follows: Let $\mathbf{s} = [s_1, s_2, ..., s_k]$ be the vector of inputs with length k, then $\mathbf{o} = [o_1, o_2, ..., o_{k \times z}]$ is the associated processed output vector, or phenotype, for a neural network with length $k \times z$, where z is the number of neural network output neurons. The phenotypic distance is employed by calculating the difference in the outputs of two samples: $d(\mathbf{o}, \mathbf{o}')$. Fig. 4.11 illustrates the sampling of phenotypes and Fig. 4.12 shows a comparison of both distances.

The phenotypic distance is always task sensitive, i.e., a comparison of two samples requires the definition of an adequate input vector $\mathbf{s}$. It needs to fulfill two requirements in the context of model-based optimization:

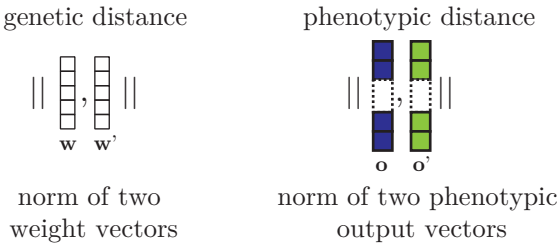

Figure 4.12: Weight models are based on weight vectors for fixed-topology networks. Phenotypic distance models are based on fixed-length sampled phenotypic output vectors for any-topology networks. The L1 norm (Manhattan distance) is used here for interpolative modeling.

a) The input should be representative for the underlying task, i.e., in case of robot control it should follow the given sensor ranges and/or depict a trajectory of states present in the task.

b) The dimensionality of the phenotype needs to be considered, the length of the input vector for generating the phenotypes might significantly affect the modeling performance as well as the computation time for querying the networks.

Given a carefully selected input vector, the phenotypic distance should be able to provide an impression of how the behaviors of two candidate networks compare to each other. A possible disadvantage of this definition of a phenotypic distance is that depending on the underlying task, the real behavior cannot be defined by the output of the neural network controller alone. This is in line with the discussion in Section 3.1.2. A full description of a behavior can only be obtained through its ecologic expression, by inserting a solution into its environment and measuring its entire influence and interaction with the environment. For example, a robot is influenced by the structure of the environment and its own body. Two robots with different controllers and phenotypes, one that uses four legs for movement and another that uses three legs, might behave the same if the fourth leg is disabled due to damage. This use case was introduced by Doncieux and Mouret (2010), utilizing an effect of neutrality.

The ultimate task is to obtain a representative set of samples of the input/output relationship which is descriptive enough to capture the behavioral differences and so allow the construction of surrogate models.

4.2.3 Evaluation

It needs to be determined, whether reasonable surrogate models can be trained based on a diverse set of phenotypic vectors, and whether the models have a comparable performance to genetic models. The main question answered in this evaluation is, whether we can correctly predict the pair-wise ranks of the performance of neural controllers. Due to the use of surrogates in an evolutionary context, it is only interesting to be able to correctly select the better-performing neural controller.

For this, model-free optimization algorithms are used that optimize the weights of fixed topology neural networks for robot control. Then, a QD archive of several hundred diverse neural networks is created based on these starting points. Different genetic and phenotypic surrogate models are then produced based on a subset of these networks, and their performance tested by predicting the performance of the remainder of the networks.

Robot controllers are designed for the multi-modal maze problem depicted in Figure 4.13. The environment consists of multiple rings and openings (Figure 4.13a). The robot begins in the center of the maze and needs to find a path to exit the maze. In the context of QD, it is not interesting to find the best solution to escape it. Instead, it needs to be established to what degree the behavior of neural network controllers can be sampled, and how accurate the derived performance model is. This problem is much more fundamental and difficult than predicting fitness alone. To produce this data set of as many different high-performing behaviors as possible, an archive consisting of robot controllers that reach every point in the maze in the shortest path possible (b) is created. To force a diversity of ending positions, a grid-like diversity measure is defined (c). At the end of the optimization, every niche should contain a robot that was able to reach it using a short path. This way the distance measures can be evaluated over a diverse set of optimal behaviors.

Simple feed forward controllers (see Figure 4.11) consisting of either two or five hidden neurons are sought that traverse the maze. Evaluation is performed using the simulation that was created by Mouret (2011a). The robot is equipped with three laser sensors that are able to detect the distance to the nearest walls, and are set at 45 degree angles around the front (d). In addition, each robot has a home beacon that detects the direction of its start position.

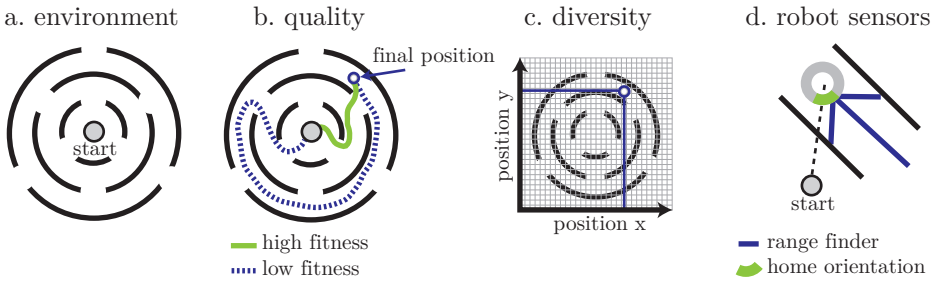

Figure 4.13: Evaluation takes place in a maze environment (a) with a robot starting in the center. The distance of the path of a robot to its final position defines its quality (b), whereby a diversity measure allows us to train robots to reach all niches in the archive (c). Robots can sense the orientation quadrant of the start position and use three range finders to perceive the distance to the nearest wall (d).

Data Generation Data sets are generated to test the quality of the surrogate models. To that end, the data of model-free optimization experiments is recorded. Here, optimization is performed with the MAP-Elites algorithm. It is not only used to find good solutions, but it is also intended to find as many diverse and high-performing solutions as possible. Here, niches are defined as bins in the grid shown in Figure 4.13.

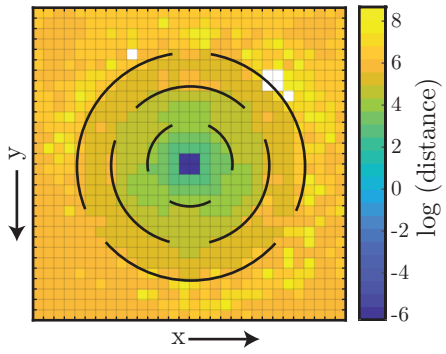

Figure 4.14: Distance archive generated by MAP-Elites (lower distance equals higher fitness). Each niche in the archive contains a robot controller that is optimized towards reaching that niche via the shortest path possible.

Figure 4.14 shows an example distance archive after 5000 generations, with almost every niche filled with a high-performing controller. The distance values naturally

grow the further they are from the center. A number of controllers end up driving around the maze in circles, which explains the high distance values in some niches.

For this experiment, 20 different MAP-Elites runs were produced for each experiment configuration, each with a different random number generator seed. This leads to 40 data sets (20 for each number of hidden neurons). Each data set is produced by roughly 900 neural network controllers. For each of those, nine different data sets were created: one with the weights, and eight with phenotypes of different sizes (4, 8, 16, 32, 64, 128, 256, 512). Note that the phenotypes are derived from the two outputs of the networks, that is, if the network is fed with four input samples, eight phenotype values are observed. Each of the 900 controllers can now be described either by its weights, or by a phenotype output vector.

The nine different data sets are now used to model the (ranked) performance of neural controllers. Again, it is only of interest to correctly predict the pair-wise ranks of the performance in this evolutionary context. During modeling, the data sets are split as follows: 400 controllers are used to train a model, the remainder is used to test the model quality. It must be considered that the observed values y will be log-scaled before modeling, as the data contains strong outliers which might deteriorate the models.

GP Model The GP model is created using the R-package CEGO (which can be found at Zaefferer (2019)) as follows. For MLE, the optimization of the likelihood is performed via the locally biased variant of the dividing rectangles (DIRECT) algorithm, which was proposed by Gablonsky and Kelley (2001). It is configured to stop after 2000 likelihood evaluations, or when the relative decrease in function values between iterations drops below 10^{-16}. Regularization of the model is turned on, to potentially account for noise in the data or ill-conditioned kernel matrices. The model uses the Manhattan distance.

Comparison Baseline: Linear Model A linear regression model is included as a benchmark for the GP model. Like the GP model, the linear model is trained with the genetic or phenotypic data. Since the generated data is potentially very high dimensional, some form of variable selection is needed to generate reasonable models. A forward selection approach via the Aikake information criterion (AIC) (see Venables and Ripley (2002)) fulfills this need, starting from a model that only

consists of an intercept. The most complex linear model may include main effects for all variables, but no interactions or higher order terms are considered.

Results and Discussion To judge the quality of the models, the rank correlation coefficient by Kendall and Gibbons (1990) is used. Figure 4.15 shows the Kendall correlation achieved by each of the models. Firstly, it can be observed that the GP model outperforms the linear model in most cases, as expected. Secondly, the variants based on phenotypic data are able to perform at least as well as the weight models, if the number of elements in the phenotype vector is at least 32 or more. The larger phenotype vectors do not seem to yield much further improvement.

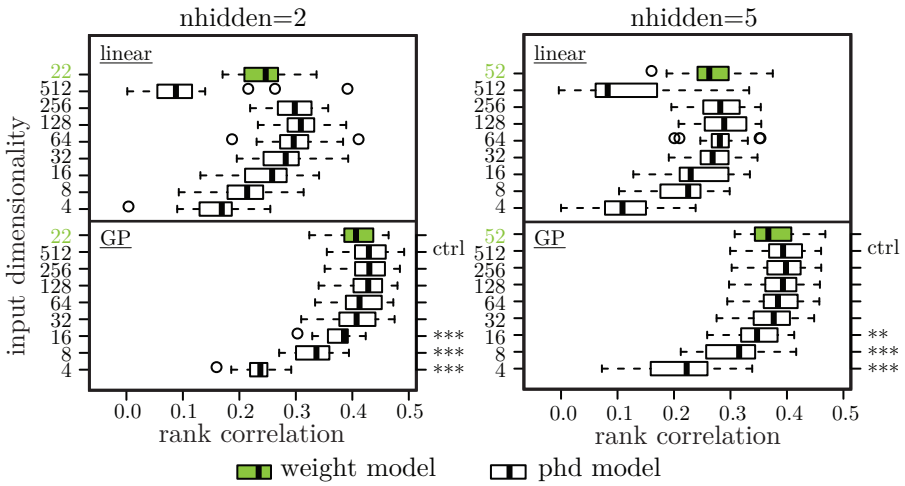

Figure 4.15: The model quality in terms of correlation (x-axis), for linear and GP models and different input spaces, and different numbers of hidden neurons (nhidden). Here, the numbers at the start of each y-axis label denote the dimensionality of the input vector for the corresponding model (number of weights or length of the phd sampling string). Green fill color indicates a model based on the weights or genome, the white fill color indicates phenotypic models. The y-axis labels on the right-hand side indicate p-values from a statistical test that compares each of the GP models against the model marked with ctrl (*: $p < 0.05$,**: $p < 0.01$,***: $p < 0.001$).

The observations are confirmed by applying statistical tests for each number of hidden neurons. A first experiment is run to evaluate the global presence of significant differences via the non-parametric rank-sum test by Kruskal and Wallis

(1952), which yields p-values of less than 10^{-8} in both cases, indicating that differences are present. Afterwards, the non-parametric many-to-one comparison test by Conover and Iman (1979) is performed, comparing each of the GP models against a single model, the control group. The chosen control group is the most complex model with phenotype data of dimensionality 512. The implementations of the employed tests are taken from the `stats` and the `PMCMRplus` `R` packages (see R Core Team (2018); Pohlert (2018)): `kruskal.test` and `kwManyOneConoverTest`. The respective cases with indications for significant differences are marked on the right-hand side of each plot in Figure 4.15. The statistical test largely confirms the visual evaluation. No evidence for differences is found between the control group and the model with the genetic weight data. Only models with phenotypic data of a dimensionality of 16 or less is deemed to be different from the control group.

Importantly, the results suggest that phenotypic surrogate models can be used instead of those based on the genome. The phenotypic data is largely unaffected by the number of hidden neurons, and, hence, the number of weights. Where standard models would struggle to compare the weights of differently structure networks, a phenotypic comparison would still be possible.

The baseline linear model shows some peculiar behavior. The model's performance drops off for models with phenotype vectors of more than 256 elements. This behavior can be largely explained with the number of coefficients selected by the AIC forward selection procedure, as shown in Figure 4.16. Clearly, the selection procedure will not select more than n variables. The required number of variables seems to increase non-linearly with the dimensionality of the data.

Notably, the GP model does not show such a performance drop, and in fact performs quite well even for the very high dimensional phenotype vectors. This may be counter-intuitive at first: GP regression is usually not suggested for high-dimensional data. But there are two possible reasons that could explain this behavior. Firstly, an isotropic model was used, which avoids the complex optimization of fitting numerous kernel parameters (θ). Secondly, there may be a correlation in the observed phenotypes. Increasing the number of samples used to generate the phenotype vector will not only increase the dimension, but also the density in the sampled space. In that sense, a new phenotype observation is likely to be quite similar to the large set of existing observations it is added

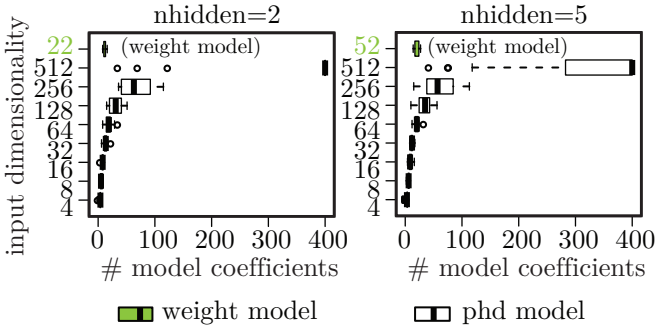

Figure 4.16: The number of linear model coefficients selected via forward selection based on AIC. Green fill color indicates a model based on the weights or genome, the remainder are based on phenotype data.

to. Essentially, it is assumed that the latent dimensionality of the data is much lower.

To verify this, a principal component analysis (PCA), introduced by Pearson (1901), of the input data is considered (that is, excluding the dependent variable). For each of the data sets, a PCA was performed on the weight data, as well as on the phenotype data. In each case, the number of principal components required to explain 90% of the variation in the data set was recorded. This number is shown in Figure 4.17.

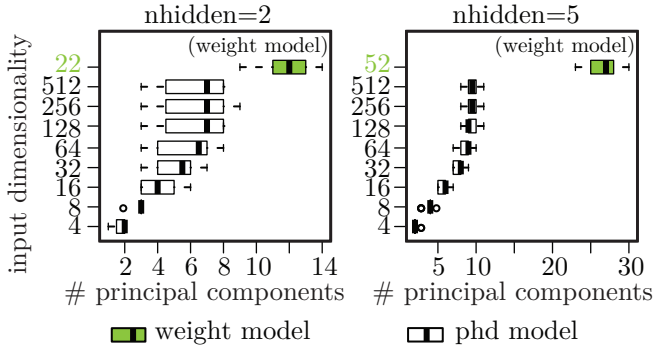

Figure 4.17: For each data set, the number of principal components required to explain 90% of the variation in the data. This only concerns the respective input data of the surrogate models, the observed output (i.e., quality of the controller) is not considered here. Green fill color indicates weight or genome data, the remainder is based on phenotypic data.

There are two interesting observations here. Firstly, the number of components levels off for the largest phenotype vectors. The median stays at seven (nhidden = 2) and nine (nhidden = 5), despite data sets with several hundreds of variables. It seems that this confirms the assumption that the additional columns due to higher-dimensional phenotype vectors actually describe a much lower-dimensional, latent space. Secondly, the number of principal components for the weights are much larger, yet, this does not coincide with better models based on the weight data.

4.2.4 Conclusions

This section gave evidence for an answer to research question VI ("Can we model neural encodings' behavior ex situ by sampling their outputs and using a behavior-kernel?"), fulfilling requirement C2 ("Search should efficiently sample complex design spaces") for neural encodings that are highly non-linear, sensitive and neutral. Phenotypic data, sampled from neural encodings, can serve as a kernel for surrogate modeling. Models based on phenotypic data can perform at least as well as those based on genetic data. This holds both for a baseline, linear model, and a non-linear GP model. The analysis further indicates that even high dimensional phenotypes with several hundreds of observations can yield sound models. A principal component analysis reveals that these high-dimensional data sets can be very well reproduced with only very few components. A much larger number of components is required for the genetic data.

This success of a phenotypic model is promising. A model based on genomes becomes infeasible if the compared networks have different structures or topologies. This can happen in the context of evolutionary algorithms that change the structure and size of the solution encoding, e.g. in surrogate-assisted neuroevolution. Measuring behavior of neural networks without using actual simulations not only seems to be possible, but also a practical way to compare networks.

Phenotypic distances can be used successfully as kernels to build surrogate models that predict the fitness of networks with varying sizes and topologies. Whereas previous approaches to construct surrogate models of neural networks with non-uniform structure rely on the peculiarities of the evolutionary algorithm (see Gaier et al. (2018)), the presented work is independent of the optimization approach. In fact, a phenotypic distance approach to modeling is independent even of encoding:

a neural network grown with NEAT, a fixed topology network optimized with particle swarm optimization, and a controller evolved with genetic programming could all share the same surrogate model.

As the PCA showed, as well as the diminishing returns for models with more phenotype samples, a lower-dimensional data set may suffice to produce good models. Creating better, more condensed phenotype samples with less redundant information is hence of interest for future work, to reduce the load of distance calculations.

Being able to successfully model the performance of a neural encoding by observing its behavior provides a computationally efficient and effective approach for surrogate modeling of varying-length representations. Modeling the behavior of networks avoids some complexities that are caused by genetic comparisons. Surrogate-assisted optimization of non-uniform representations allows a much more diverse set of solutions to be calculated with a limited number of real evaluations.

4.3 Chapter Summary

This chapter introduced a method that increase the efficiency of QD algorithms when using expensive phenotypic features for niching. It was shown that behavioral airflow features can be learned by piggybacking on Bayesian optimization, acquiring samples only according to uncertainties in the prediction of their performance. The resulting surrogate-assisted QD method, SPHEN, needs orders of magnitudes less exact evaluations than methods that do not or only partially use surrogate-assistance. This answers research question V ("Can we model behavioral features in a surrogate-assisted way by sampling based on optimality alone?"). Efficiently generating phenotypically diverse solution sets is necessary to the application of divergent search in problem domains common to engineering.

Furthermore, neural representations were modeled through their behavior using phenotypic sampling. This method allows building kernels for GP models based on behavioral distance. Although the application of phenotypic distance-based kernels has to be analyzed in future work, evidence was given for a positive answer to research question VI ("Can we model neural encodings' behavior ex situ by sampling their outputs and using a behavior-kernel?").

Both methods enable phenotype-based multi-solution optimization methods to be used as an efficient divergent component in co-creative processes.