EasyChair Preprint

№ 3291

Designing Air Flow with Surrogate-assisted Phenotypic Niching

Alexander Hagg, Dominik Wilde, Alexander Asteroth and Thomas Bäck

April 28, 2020

Designing Air Flow with Surrogate-assisted Phenotypic Niching

Alexander Hagg[1,3(✉)][0000−0002−8668−1796], Dominik Wilde[2,1][0000−0003−3263−7287], Alexander Asteroth[1][0000−0003−1133−9424], and Thomas Bäck[3][0000−0001−6768−1478]

[1] Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany
{alexander.hagg,dominik.wilde,alexander.asteroth}@h-brs.de
[2] Chair of Fluid Mechanics, University of Siegen, Siegen, Germany
[3] Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands
t.h.w.baeck@liacs.leidenuniv.nl

Abstract. In complex, expensive optimization domains we often narrowly focus on finding high performing solutions, instead of expanding our understanding of the domain itself. But what if we could quickly understand the complex behaviors that can emerge in said domains instead? We introduce surrogate-assisted phenotypic niching, a quality diversity algorithm which allows to discover a large, diverse set of behaviors by using computationally expensive phenotypic features. In this work we discover the types of air flow in a 2D fluid dynamics optimization problem. A fast GPU-based fluid dynamics solver is used in conjunction with surrogate models to accurately predict fluid characteristics from the shapes that produce the air flow. We show that these features can be modeled in a data-driven way while sampling to improve performance, rather than explicitly sampling to improve feature models. Our method can reduce the need to run an infeasibly large set of simulations while still being able to design a large diversity of air flows and the shapes that cause them. Discovering diversity of behaviors helps engineers to better understand expensive domains and their solutions.

Keywords: evolutionary computation · quality diversity · phenotypic niching · computational fluid dynamics · surrogate models · Bayesian optimization.

1 Introduction

We design objects with the expectation that they will exhibit a certain behavior. In fluid dynamics optimization, we want an airplane wing to experience low drag forces, but also have a particular lift profile, depending on angle of attack and air speed. We want to understand how the design of our public transportation hubs, dealing with large influxes of travelers, can cause congestion at maximal flow rates. We want our buildings to cause as little wind nuisance as possible and understand how their shape and the wind turbulence they cause are linked.

In all these cases, it is not easy to design without prior experience and we often require long iterative design processes or trial-and-error methods.

What if we could quickly understand the possible types of behavior in expensive engineering problems? How could we get an early intuition about how shape and behavior are related? In this work, we try to answer these questions, and in particular, whether we can discover different high performing behaviors of shapes, designing air flow simultaneously to the shapes that cause it.

An overview of related work is given in Section 2, where we explain quality diversity (QD) algorithms and the use of surrogate assistance, describe turbulence effects in air flow and the fluid dynamics solver that allows us to quickly calculate air flow time sequences. In Section 3 we introduce a new QD algorithm that performs surrogate-assisted phenotypic niching, which allows us to collect different high performing solutions. Two problem domains are introduced (Section 4): one inexpensive domain that optimizes the symmetry of polygons, allowing us to perform an in depth evaluation of various QD methods, and an air flow domain, where we apply our method to an expensive domain (Section 5).

2 Quality Diversity

QD algorithms combine the idea of "blind" novelty search [14], which searches for novel solutions without taking into account their performance, and performance-based search. QD finds a diverse set of high performing optimizers [3, 15] by only allowing solutions belonging to the same niche to compete locally. Niching is based on features that describe some aspects of the phenotype, like shape, structure or behavior. It keeps track of a growing archive of niches, where solutions are added if their phenotype fills a new niche or their quality is higher than that of the solution that was previously placed inside.

QD only became applicable to expensive optimization problems after the introduction of surrogate-assisted illumination (SAIL) [7]. In this Bayesian interpretation of QD, a multi-dimensional archive of phenotypic elites (MAP-Elites) [3] is used to fill an acquisition map based on *upper confidence bound* (UCB) [1] sampling, which takes an optimistic view at surrogate-assisted optimization. A Gaussian Process (GP) regression [18] model is used to predict the performance of new solutions, based on the distance to previous examples, which is modeled using a covariance function. A common covariance function is the squared exponential, which has two hyperparameters: the length scale (or sphere of influence) and the signal variance, which are found by minimizing the negative loglikelihood of the process. For any location, the GP model predicts a distribution of values, including confidence intervals of the prediction. This confidence interval is added to the prediction mean with the idea that a location where the model has low confidence also has the promise of holding a better performing solution: $UCB(x) = \mu(x) + \kappa \cdot \sigma(x)$. The parameter κ allows us to tune the UCB function between exploitation ($\kappa = 0$) and exploration ($\kappa >> 0$).

In SAIL, after MAP-Elites fills the acquisition map, which now contains "optimistic" solution candidates, a random selection of those candidates is analyzed

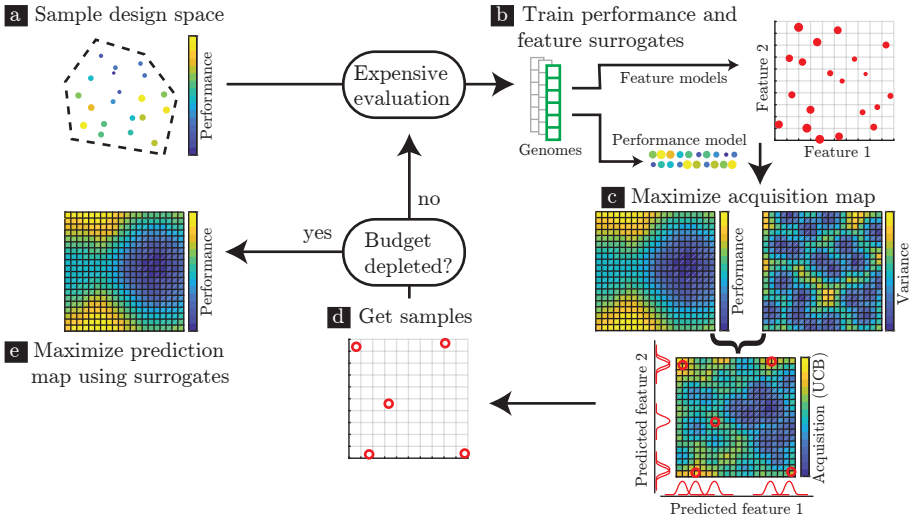

Fig. 1. Surrogate-assisted Phenotypic Niching. An initial sample set is evaluated (a), then models are trained to predict performance and feature coordinates (b), MAP-Elites is used to produce an acquisition map, balancing exploitation and exploration with the UCB of the performance model. Feature models predict the niche of new individuals (c). New samples are selected from the acquisition map (d). After the evaluation budget is depleted, the surrogate models are used to generate the prediction map with MAP-Elites, ignoring model confidence (e).

in the expensive evaluation function to form additional training samples for the GP model. This loop continues until the evaluation budget is exhausted. Then κ is set to 0 in a final MAP-Elites run to create a feature map that now contains a diverse set of solutions that is predicted to be high-performing. 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 [7] that can be determined without running the expensive evaluation.

With SAIL it became possible to use performance functions of typical optimization domains like fluid dynamics. But the strength of QD, to perform niching based on behavior, cannot be applied when measuring those behaviors themselves is expensive. In this work we evaluate whether we can include surrogate models for expensive features, specifically in a fluid dynamics domain.

3 Surrogate-assisted Phenotypic Niching

To be able to handle expensive behavioral features, we introduce surrogate-assisted phenotypic niching (SPHEN) (Fig. 1 and Alg. 1). By building on the insight that replacing the performance function with a surrogate model decreases the necessary evaluation budget, we replace the exact feature descriptors as well.

Algorithm 1 Surrogate-assisted Phenotypic Niching

Set $budget$, $maxGens$, $numInitSamples$ ▷ Configure
$\mathcal{X}' \leftarrow \mathcal{X} \leftarrow Sobol(numInitSamples)$ ▷ Initial samples
while $|\mathcal{X}| < budget$ **do**
  $(\mathbf{f}', \mathbf{p}') \leftarrow Sim(\mathcal{X}')$ ▷ Precisely evaluate performance and features
  $(\mathcal{X}, \mathbf{f}, \mathbf{p}) \leftarrow (\mathcal{X} \cup \mathcal{X}', \mathbf{f} \cup \mathbf{f}', \mathbf{p} \cup \mathbf{p}')$
  $(\mathbf{M}_p, \mathbf{M}_f) \leftarrow Train(\mathcal{X}, \mathbf{f}, \mathbf{p})$ ▷ Train surrogate models
  $\mathbf{A}_{map} \leftarrow \text{MAP}(20, \mathcal{X}, Predict(\mathcal{X}, \mathbf{M}_f), Predict(\mathcal{X}, \mathbf{M}_p), \mathbf{M}_p, \mathbf{M}_f)$ ▷ Produce acquisition map based on predicted sample performance and features
  $\mathcal{X}' \leftarrow Sobol(\mathbf{A}_{map})$ ▷ Select new samples from acquisition map
end while
$\mathbf{P}_{map} \leftarrow \text{MAP}(acq(), 0, feat(), \mathcal{X}, \mathbf{f}, \mathbf{p}, \mathbf{M}_p, \mathbf{M}_f)$ ▷ Produce prediction map

procedure $\text{MAP}(\sigma_{ucb}, \mathcal{X}, \mathbf{f}, \mathbf{p}, \mathbf{M}_p, \mathbf{M}_f)$
  $\mathbf{I}_{map} \leftarrow (\mathcal{X}, \mathbf{f}, \mathbf{p})$ ▷ Create initial map
  while $gens < maxGens$ **do**
    $\mathbf{P} \leftarrow Sobol(\mathbf{I}_{map})$ ▷ Evenly, pseudo-randomly select parents from map
    $\mathbf{C} \leftarrow Perturb(\mathbf{P})$ ▷ Perturb parents to get children
    $\mathbf{f} \leftarrow Predict(\mathbf{C}, \mathbf{M}_f)$ ▷ Predict features
    $\mathbf{p} \leftarrow UCB(\mathbf{C}, \sigma_{ucb}, \mathbf{M}_p)$ ▷ Predict performance (Upper Confidence Bound)
    $\mathbf{I}_{map} \leftarrow Replace(\mathbf{I}_{map}, \mathbf{C}, \mathbf{f}, \mathbf{p})$ ▷ Replace bins if empty or better
  end while
end procedure

The initial training sample set, used to form the first seeds of the acquisition map, is produced by a space-filling, quasi-random low-discrepancy Sobol [21] sequence in the design space (Fig. 1a). 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 of the search domain. The initial set is evaluated, for example in a computational fluid dynamics simulator. Performance and phenotypic features of those samples are derived from the simulation 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 we do not know what part of the phenotypic feature space will be discovered in the process, the initial set's feature coordinates only give us a hint of the reachable feature space. Just because we used a space-filling sampling technique in the design space, does not mean the samples are space-filling in feature space.

After collecting performance and feature values, the surrogate models are trained (Fig. 1b). We use standard GP models, which limit the number of samples to around 1,000, as the training and prediction becomes quite expensive. We use a squared exponential covariance function 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 [19] conjugate gradients based minimization method, running up to 1,000 iterations.

MAP-Elites is used to create the acquisition map by optimizing the UCB (with a large exploration factor $\kappa = 20$) of the performance model, using feature

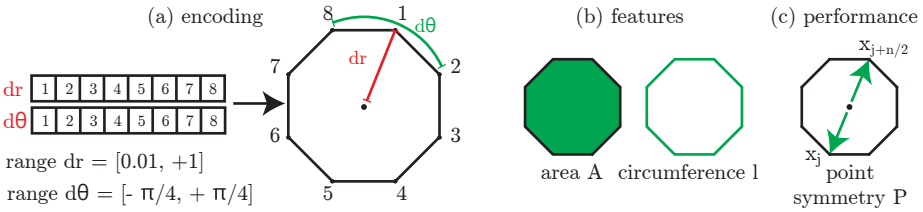

Fig. 2. The genome (a), consisting of 16 parameters that define axial and radial deformations, shape features (b) and performance (c) of polygons in the domain.

models to assign niches for the samples and new solutions (Fig. 1c). Notably, we do not take into account the variance of those feature models. Surrogate assisted QD works, because, although the search takes place in a high-dimensional space, QD only has to find the *elite hypervolume* [23], or *prototypes* [9], the small volumes 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.

We use a Sobol sequence on the bins of the acquisition map to select new solutions (Fig. 1d) that 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 (Fig. 1e), which is filled with diverse, high-performing solutions.

4 Domains

Phenotypic features can describe phenomena that are related to complex domains, like behavioral robotics, mechanical systems, or computational fluid dynamics (CFD). Before we apply SPHEN to an expensive CFD domain, we compare its performance to MAP-Elites and SAIL in a simpler, inexpensive domain.

4.1 Polygons

To be able to calculate all ground truth performance and feature values, in this domain we optimize free form deformed, eight-sided polygons. The polygons are encoded by 16 parameters controlling the polar coordinate deviation of the polygon control points (Fig. 2a). The first half of the genome determines the corner points' radial deviation ($dr \in [0.01, 1]$). The second half of the genome determines their angular deviation ($d\theta \in [-\pi/4, \pi/4]$). The phenotypic features are the area of the polygon A and its circumference l (Fig. 2b). These values are normalized between 0 and 1 by using predetermined feature ranges ($A \in [0.01, 0.6]$ and $l \in [1, 4]$). The performance function (Fig. 2c) is defined as the point symmetry P. The polygon is sampled at $n = 100$ equidistant locations

on the polygon's circumference, after which the symmetry metric is calculated (Eq. 1). The metric is based on the symmetry error E_s, the sum of Euclidean distances of all $n/2$ opposing sampling locations to the center:

$$f_P(x_i) = \frac{1}{1 + E_s(x_i)}, E_s(x) = \sum_{j=1}^{n/2} \left\| \boldsymbol{x}_j - \boldsymbol{x}_{j+n/2} \right\| \tag{1}$$

4.2 Air Flow

The air flow domain is inspired by the problem of wind nuisance in the built environment. Wind nuisance is defined in building norms [10, 16] and usually uses the wind amplification factor measured in standardized environments, based on the potential wind speed, the hourly mean wind speed. In a simplified 2D setup, we translate this problem to that of minimizing maximum air flow speed (u_{Max}) based on a fixed flow input speed. The performance is determined as the inverse over the normalized maximum velocity: $p(x) = \frac{2}{(1+u_{Max}(x))} - 1$. However, we only need to keep u_{Max} within a set *nuisance threshold*, which we set to $u_{Max} = 0.12$.

The encoding from the polygon domain is used to produce 2D shapes that are then placed into a CFD simulation. To put emphasis on the architectural nature of the domain, we use two features, area and air flow 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, we want to analyze, under the condition of keeping the flow velocity low, how the size of the area and turbulence are related to each other. We want to produce polygons that are combinations between their appearance (small to large) and their effect on the flow (low turbulence and 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 intensities and areas. However, for area/turbulence combinations in between, the design of the shape is not unique and will possibly differ from intuition.

Lattice Boltzmann Method The lattice Boltzmann method (LBM) is an established tool for the simulation of CFD [13]. 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 2D grid with the usual "D2Q9" 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 [2], a Smagorinsky subgrid model [6] is used

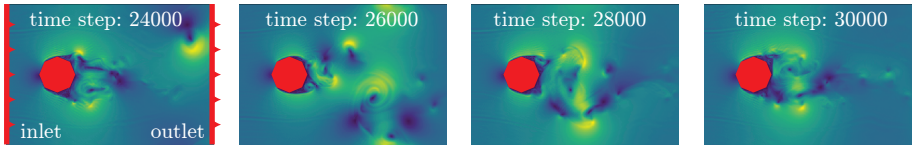

Fig. 3. Air flow around a circular polygon shape at four different time steps.

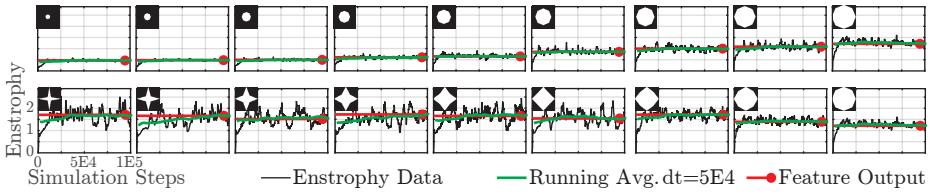

Fig. 4. Enstrophy values during simulation of circles and stars. The running average of the last 50,000 time steps converges to the final feature output.

to account for the under-resolved flow in the present configuration. For a more detailed description of the underlying mechanisms, we refer to [13]. Note that the results of the 2D domain do not entirely coincide with results that will be found in 3D, caused by the difference in turbulent energy transport [22].

The simulation domain consists of $300 \cdot 200$ grid points. A bitmap representation of the polygon is placed into this domain, occupying up to $64 \cdot 64$ 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* is used [11], which is based on the PyTorch framework [17], allowing easy access to GPU functionality. The fluid dynamics experiment was run on a GPU-cluster with four GPU nodes, each simulation taking ten minutes. Fig. 3 shows the air flow around a circular polygon at four different, consecutive time steps. Brighter colors represent higher magnitudes of air flow 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 [8, 12], 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. We employ a running average over the last 50,000 time steps. To test whether we indeed converge to a stable value, we run simulations 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. The value converges to the final feature value (red).

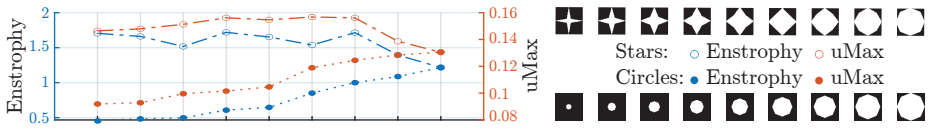

Fig. 5. Enstrophy and maximum velocity of circles and stars.

To validate the two measures, we simulate two small shape sets of circles and stars. 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. 5.

Next, we investigate whether we can predict the simple shapes' flow feature values correctly. 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. We perform a grid search 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 we measure the accuracy using the mean absolute percentage error (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.

5 Evaluation

We evaluate how well SPHEN performs in comparison to SAIL and MAP-Elites when we include the cost of calculating the features, how accurate the feature models are when trained with a performance based acquisition function, and whether we can apply SPHEN to an expensive domain.

5.1 Quality Diversity Comparison

We run QD optimization without (MAP-Elites) and with surrogate model(s) (SAIL, SPHEN) on the polygon domain (Section 4.1). This allows us to check all ground truth performance and feature values in a feasible amount of time. The shape features should be easier to learn than the flow features of the air flow domain. The working hypothesis is that we expect SPHEN to perform somewhere between SAIL and MAP-Elites, as it has the advantage of using a surrogate model but also has to learn two phenotypic features. But in the end, since the ultimate goal is to be able to use QD on expensive features, SPHEN will be our only choice. The parameterization of all algorithms is listed in Table 1. The initial sample set of 16 examples as well as the selection of new samples (16

Table 1. Parameter settings for MAP-Elites, SAIL[A], restricted SAIL[B] and SPHEN.

Parameter	MAP-Elites	SAIL[A]	SAIL[B]	SPHEN
Generations	4,096	1,024	63	1,024
Descendants	16	32	16	32
Budget (per iteration)	-	1,024 (16)	1,024 (16)	1,024 (16)
Resolution (acquisition)	-	16x16	16x16	32x32

[A] Due to the number of feature evaluations, MAP-Elites uses $4{,}096 \cdot 16 = 65{,}536$ and SAIL uses $16 + 1{,}024 \cdot 32 \cdot (\frac{1{,}024}{16}) + 1{,}024 = 2{,}098{,}192$ evaluations.
[B] Here, SAIL is restricted to the number of evaluations that was used in MAP-Elites. Number of generations ($\frac{4{,}096 \cdot 16 - 1{,}024 - 16}{1{,}024} \approx 63$).

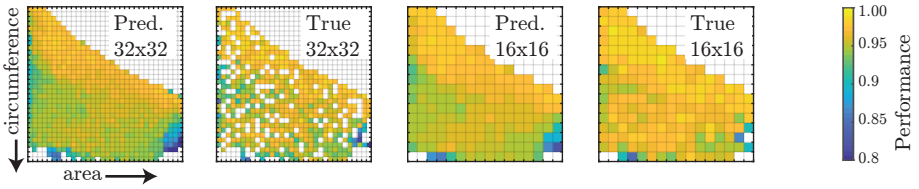

Fig. 6. Predicted and true SPHEN maps on symmetry domain, trained in 32x32 resolution (left), then reduced to 16x16 resolution to remove holes (right).

in every iteration) is handled by a pseudo-random Sobol sequence. The mutation operator adds a value drawn from a Gaussian distribution with $\sigma = 10\%$.

Due to the expected inaccuracy of the feature models, misclassifications will decrease the accuracy of the maps. Fig. 6 shows a prediction map at a resolution of 32x32 and the true performance and feature map. Holes appear due to misclassifications, which is why we train SPHEN on a higher resolution map and then reduce the prediction map to a resolution of 16x16. Most bins are now filled. In this experiment all prediction maps have a resolution of 16x16 solutions.

The mean amount of filled map bins and performance values for five replicates are shown in Fig. 7. SAIL and SPHEN find about the same number of solutions using the same number of performance evaluations (PE). Notably, the mean performance of SPHEN's solutions is higher than that of SAIL. However, in domains with expensive feature evaluations we need to take into account the performance or feature evaluations (PFE). SAIL now needs more than two million PFE to perform almost as well as SPHEN, which only needs 1,024, which is 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 1,000 function evaluations, due to the infeasibly large computational investment, the efficiency gain of SPHEN is substantial. If we lower the number of PFE of SAIL to the same budget as MAP-Elites and give it more time to search the iteratively improving surrogate model before running out of the budget of 65,536 PFE (see Table 1), SAIL still takes a big hit, not being able to balance out quality and diversity. The example

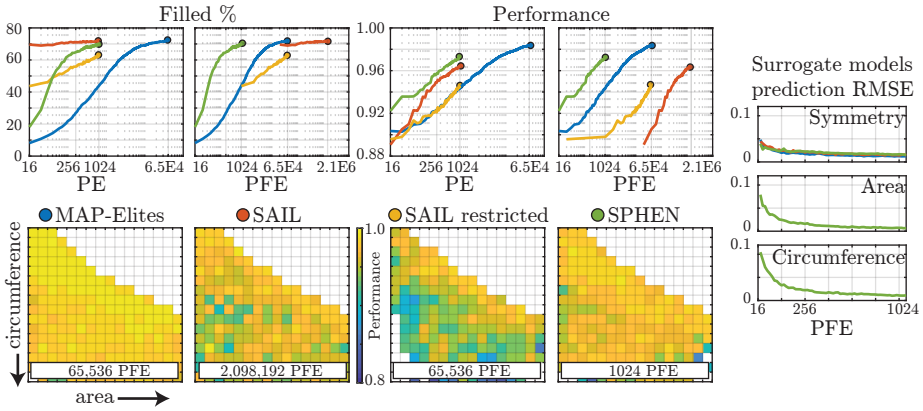

Fig. 7. Comparison of MAP-Elites, SAIL and SPHEN based on performance evaluations (PE) and performance/feature evaluations (PFE). Prediction errors are included on the right and example prediction maps at the bottom. The experiment includes a version of SAIL that is restricted to the number of PFE used in MAP-Elites.

prediction maps are labeled with the number of PFE necessary to achieve those maps. Although we do not sample new training examples to improve the feature models specifically, their root mean square error (RMSE) ended up at 0.012 and 0.016 respectively. We conclude that we do not need to adjust the acquisition function. SPHEN and SAIL search for the same elite hypervolume, which is only determined by the performance function.

5.2 Designing Air Flow

After showing that SPHEN can learn both performance as well as feature models, we now run SPHEN in the air flow domain (Section 4.2). The objective is to find a diverse set of air flows using a behavioral feature, turbulence, and one shape feature, the surface area of the polygon. We want to find out how the size of the area and turbulence are related to each other and which shapes do not pass the wind nuisance threshold. We use the same parameters for SPHEN as were listed in Table 1, but allow 4,096 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 map of solutions in Fig. 8 shows that turbulence and surface area tend to increase the mean maximum air flow velocity, as expected. A small selection of air flows is shown in detail. RMSE of the models is 0.06, 0.01 and 0.10 respectively and Kendall's tau rank correlation to the ground truth amounts to 0.78, 1.00 and 0.73 (1.00 for A, B, C and D).

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 [4, 20].

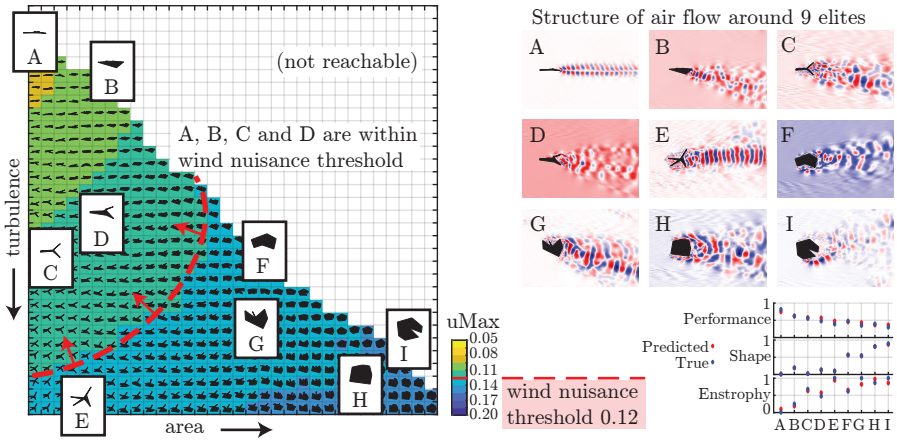

Fig. 8. A diversity of shapes and air flows that shows which designs conform to the wind nuisance threshold. The dominant DMD mode shows the structure of the air flow around nine selected shapes. A, B, C and D are within the wind nuisance threshold.

Especially those shapes at the extrema of area and turbulence align with the aerodynamic expectations as detailed in Section 4.2. At low turbulence intensity, the shapes tend to be slim and long with respect to the flow direction (shapes A and B). High turbulence 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 [5], 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.

5.3 Discussion

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 air flows, one SPHEN run took 23 hours, producing 494 different air flow profiles. With SAIL, obtaining the same result would have taken over five years. Although MAP-Elites outperformed SAIL in the simple polygon domain, and might have outperformed it in the air flow domain as well, it still would have taken two months to calculate with uncertain result. Fig. 8 shows that we can find structure in the air flows that can appear in this problem domain. We can efficiently combine variations (area) of the object we want to design as well as their effect on the environment (turbulence). 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, and could continue the design process based on our new intuition.

6 Conclusion

In this work we showed that expensive phenotypic features can be learned along with an expensive performance function, allowing SPHEN, an evolutionary QD algorithm, to find a large diversity of air flows. In an inexpensive domain we showed that, when we take into account the number of feature evaluations, SPHEN clearly outperforms state of the art algorithms like MAP-Elites and SAIL. The result clears the way for QD to find diverse phenotypes as wel as behaviors in engineering domains without the need for an infeasible number of expensive simulations. This is made possible because only the elite hypervolume needs to be modeled. Fluid dynamics domains count as some of the most complicated. Although often solved in ingenious ways by engineers relying on experience, QD can add *automated intuition* to the design process. Variations of the object we want to optimize as well as variations in the effects on the object's environment can be seen "at a glance", which is what intuition is all about.

The most urgent future work is to study whether we can make adjustments to the acquisition function, taking into account feature models' confidence intervals to improve SPHEN, or whether we can abandon semi-non-parameterized modeling techniques altogether to invest in more performant parameterized models. Furthermore, the solution diversity should be analyzed in higher-dimensional feature spaces and applied to 3D shapes.

We showed what expected and unexpected behavioral patterns can emerge in complicated problem domains using surrogate-assisted phenotypic niching. Our main contribution, automatic discovery of a broad intuition of the interaction between shape and behavior, allows engineers to think more out-of-the-box.

Acknowledgments. This work was funded by the Ministry for Culture and Science of the state of Northrhine-Westphalia (grant agreement no. 13FH156IN6) and the German Research Foundation (DFG) project FO 674/17-1. The authors thank Andreas Krämer for the discussions about the Lettuce solver.

References

1. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. Journal of Machine Learning Research **3**(Nov), 397–422 (2002)
2. Bhatnagar, P.L., Gross, E.P., Krook, M.: A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems. Physical review **94**(3), 511 (1954)
3. Cully, A., Clune, J., Tarapore, D., Mouret, J.B.: Robots that can adapt like animals. Nature **521**(7553), 503–507 (2015)
4. Demo, N., Tezzele, M., Rozza, G.: Pydmd: Python dynamic mode decomposition. Journal of Open Source Software **3**(22), 530 (2018)
5. Dorschner, B., Chikatamarla, S.S., Karlin, I.V.: Transitional flows with the entropic lattice Boltzmann method. J. Fluid Mech. **824**, 388–412 (2017)
6. Gaedtke, M., Wachter, S., Rädle, M., Nirschl, H., Krause, M.J.: Application of a lattice boltzmann method combined with a smagorinsky turbulence model to spatially resolved heat flux inside a refrigerated vehicle. Computers & Mathematics with Applications **76**(10), 2315–2329 (2018)
7. Gaier, A., Asteroth, A., Mouret, J.B.: Data-Efficient Exploration, Optimization, and Modeling of Diverse Designs through Surrogate-Assisted Illumination (2017)
8. Gassner, G.J., Beck, A.D.: On the accuracy of high-order discretizations for under-resolved turbulence simulations. Theoretical and Computational Fluid Dynamics **27**(3-4), 221–237 (2013)
9. Hagg, A., Asteroth, A., Bäck, T.: Prototype discovery using quality-diversity. In: International Conference on Parallel Problem Solving from Nature. pp. 500–511. Springer (2018)
10. Janssen, W., Blocken, B., van Hooff, T.: Pedestrian wind comfort around buildings: Comparison of wind comfort criteria based on whole-flow field data for a complex case study. Building and Environment **59**, 547–562 (2013)
11. Krämer, A., Wilde, D., Bedrunka, M.: Lettuce: PyTorch-based Lattice Boltzmann Solver (2020)
12. Krämer, A., Wilde, D., Küllmer, K., Reith, D., Foysi, H.: Pseudoentropic derivation of the regularized lattice Boltzmann method. Phys. Rev. E **100**(2), 1–16 (2019)
13. Krüger, T., Kusumaatmaja, H., Kuzmin, A., Shardt, O., Silva, G., Viggen, E.M.: The lattice Boltzmann method: Principles and practice (2017)
14. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search for novelty alone. Evolutionary computation **19**(2), 189–223 (2011)
15. Lehman, J., Stanley, K.O.: Evolving a diversity of virtual creatures through novelty search and local competition. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation. pp. 211–218 (2011)
16. Wind comfort and wind danger in the built environment (in dutch). Norm NEN 8100 (2006)
17. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach, H., Larochelle, H., Beygelzimer, A., D'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)
18. Rasmussen, C.E.: Evaluation of Gaussian processes and other methods for non-linear regression. Ph.D. thesis, University of Toronto Toronto, Canada (1997)

19. Rasmussen, C.E., Nickisch, H.: Gaussian processes for machine learning (GPML) toolbox. Journal of Machine Learning Research **11**, 3011–3015 (2010)
20. Schmid, P.J.: Dynamic mode decomposition of numerical and experimental data. Journal of fluid mechanics **656**, 5–28 (2010)
21. Sobol', I.M.: On the distribution of points in a cube and the approximate evaluation of integrals. Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki **7**(4), 784–802 (1967)
22. Tennekes, H.: Turbulent flow in two and three dimensions. Bulletin of the American Meteorological Society **59**(1), 22–28 (1978)
23. Vassiliades, V., Mouret, J.B.: Discovering the elite hypervolume by leveraging interspecies correlation. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 149–156 (2018)