Accelerating Evolutionary Design Exploration with Predictive and Generative Models

Adam Gaier

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Accelerating Evolutionary Design Exploration with Predictive and Generative Models

THÈSE

présentée et soutenue publiquement le 17 juillet 2020

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Adam Gaier

Composition du jury

Rapporteurs :	Emma HART	Professor Edinburgh Napier University, UK
	Sebastian RISI	Associate Professor IT University of Copenhagen, Denmark
Examinateurs :	Kenneth STANLEY	Professor University of Central Florida, USA
	Sylvain LEFEBVRE	Directeur de recherche Inria, CNRS, Université de Lorraine, France
Directeur :	Jean-Baptiste MOURET	Directeur de recherche Inria, CNRS, Université de Lorraine, France
	Alexander ASTEROTH	Professor Hochschule Bonn-Rhein-Sieg, Germany

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

European Research Council
Established by the European Commission

informatics mathematics

advancing the frontiers

resibots

Loria
Laboratoire lorrain de recherche en informatique et ses applications

ENGLISH ABSTRACT

Optimization plays an essential role in industrial design, but all too often it boils down to a simple function to be minimized, such as cost or strength. More difficult considerations such as ease of manufacture, aesthetics or environmental impact can be more difficult to assess. So when optimization tools are used in design, they are often used not at the end, but at the conceptual phase to explore different possibilities.

To truly support designers, it follows that we should focus on algorithms that empower exploration. Quality diversity (QD) algorithms, which place variety on equal footing with optimality could produce the variety needed for design exploration — but can require the evaluation of millions of designs. Engineering optimization often relies on expensive physical simulations, making applying QD impractical.

In this thesis we propose methods to radically improve the data-efficiency of QD in order to make it applicable to design problems. Machine learning and QD are integrated into algorithms that rapidly explore by active learning of predictive and generative models.

In our first contribution, we develop a method of modeling the performance of evolved neural networks used for control and design. The structures of these networks grow and change, making them difficult to model. We find that by tracking their heredity we can estimate their performance. Creating models which form predictions based on heredity allows integration of evolved neural networks into model-based optimization frameworks — in our experiments improving data-efficiency by several times.

In our second contribution we combine model-based optimization with MAP-Elites, a QD algorithm. A model of the objective function is created from known designs, and MAP-Elites creates a new set of optimal designs using this inexpensive approximation. A selection of these designs are chosen to be evaluated and added to the model, improving the predictions of the models in the next iteration. In benchmark aerodynamics problems we show that with the same evaluation budget for a black-box optimizer to find a single solution, the algorithm finds collections of hundreds of designs of equivalent optimality. Improving the efficiency of MAP-Elites by several orders of magnitude expands the reach of QD to expensive domains — demonstrated by the generation of hundreds of thousands of varied designs in an expensive 3D aerodynamics case.

Our third contribution integrates generative models into MAP-Elites to learn domain specific encodings. A variational autoencoder is trained on the high performing solutions produced by MAP-Elites, capturing the common "recipe" for high performance. The algorithm

can then use this autoencoder as a custom encoding, allowing the rapid generation of high performing solutions. We show this approach allows MAP-Elites to easily scale to problems with 1000 interacting dimensions, and demonstrate that the learned encoding can be reused by other algorithms for rapid optimization.

Throughout this thesis, though the focus of our vision is design, we examine applications in other fields, such as robotics. The contributions in this manuscript are not meant to be exclusive to design, but as foundational work on the integration of QD and machine learning.

FRENCH ABSTRACT

L'optimisation joue un rôle essentiel dans la conception industrielle, mais trop souvent elle se résume à une simple fonction à minimiser, comme le coût ou la résistance. Les considérations plus difficiles à évaluer comme la facilité de fabrication, l'esthétique ou l'impact environnemental sont difficiles à prendre compte. Ainsi, lorsque des outils d'optimisation sont utilisés dans un processus de conception, ils le sont souvent non pas à la fin, pour optimiser, mais plutôt lors de la phase conceptuelle pour explorer différentes possibilités.

Cette thèse part du principe que pour soutenir les concepteurs nous devons nous concentrer sur des algorithmes qui facilitent l'exploration et non uniquement l'optimisation. Les algorithmes de "quality diversity" (QD), qui mettent sur un pied d'égalité la diversité et l'optimalité, pourraient produire le type de variété nécessaire à l'exploration en conception. Néanmoins, ils exigent l'évaluation de millions de candidats, alors que les domaines industriels nécessitent souvent des simulations physiques coûteuses.

Dans cette thèse, nous proposons des méthodes pour améliorer radicalement l'efficacité en données de la QD afin de la rendre applicable aux problèmes de conception et d'optimisation dans l'industrie. Pour cela, nous intégrons des méthodes d'apprentissage automatique et des algorithmes de "quality diversity" pour accélérer les calculs.

Dans notre première contribution, nous développons une méthode de modélisation des performances des réseaux neuronaux évolués utilisés pour la commande (par exemple en robotique) et dans conception automatique. Les structures de ces réseaux se développent et changent, ce qui rend difficile d'apprendre un prédicteur des performances. Nous montrons que la création de modèles qui forment des prévisions basées sur l'hérédité permet d'intégrer les réseaux neuronaux évolués dans des cadres d'optimisation basés sur des modèles – dans nos expériences, l'efficacité des données a été améliorée significativement.

Dans notre deuxième contribution, nous combinons l'optimisation basée sur des modèles avec MAP-Elites, un algorithme de QD. Un modèle de la fonction objectif est créé à partir de modèles connus, et MAP-Elites crée un nouvel ensemble de modèles optimaux en utilisant cette approximation peu coûteuse. Une sélection de ces conceptions est choisie pour être évaluée et ajoutée au modèle, améliorant les prédictions des modèles lors de la prochaine itération. Dans les problèmes d'aérodynamique de la littérature, nous montrons qu'avec le même budget d'évaluations que celui utilisé par optimiseur de l'état de l'art pour trouver une solution unique, l'algorithme trouve des collections de centaines de solutions d'optimalité équivalente.

L'amélioration de l'efficacité des MAP-Elites de plusieurs ordres de grandeur étend la portée de la QD à des domaines coûteux - ce qui est démontré par la génération de centaines de milliers de solutions diverses et performantes dans un cas coûteux d'aérodynamique en trois dimensions.

Notre troisième contribution intègre des modèles génératifs dans MAP-Elites pour apprendre des codages spécifiques à un domaine. Un auto-codeur variationnel est formé sur les solutions à haute performance produites par MAP-Elites, capturant la "recette" commune. L'algorithme peut ensuite utiliser cet auto-codeur comme un codage personnalisé, permettant la génération rapide de solutions à haute performance. Nous montrons que cette approche permet à MAP-Elites de s'adapter facilement à des problèmes de 1000 dimensions, et que le codage appris peut être réutilisé par d'autres algorithmes pour une optimisation rapide.

Tout au long de cette thèse, bien que notre vision se concentre sur la conception, nous examinons des applications dans d'autres domaines, comme la robotique. Les contributions de ce manuscrit ne sont pas destinées exclusivement à la conception, mais forment un travail fondamental sur l'intégration de la QD et de l'apprentissage automatique.

Pour entrevoir l'avenir de l'optimisation dans le domaine de la conception, on peut se pencher sur l'histoire des échecs, un domaine déjà "conquis" par les ordinateurs. En 1997, Garry Kasparov, le champion du monde d'échecs, a perdu contre Deep Blue, le supercalculateur d'IBM. Le tournoi est devenu une parabole maintes fois répétée de l'inévitable domination de la force technologique sur le talent humain. À partir de ce moment, l'histoire conclut qu'aucun humain ne pourra plus jamais espérer battre une machine aux échecs.

Sept ans plus tard, les meilleures machines de jeu d'échecs du monde ont été battues — sévèrement.

Aucun nouveau prodige du jeu d'échecs n'est apparu comme le sauveur de l'humanité. La machine d'échecs la plus avancée jamais créée, un mélange d'algorithmes de pointe et de matériel sur mesure, a été complètement surclassée par un groupe d'adversaires. Mais ce n'est pas la ruse humaine qui a fait la différence, mais une mise à niveau du terrain de jeu. Dans ce qu'on appelait un tournoi d'échecs "libre", chaque concurrent avait accès à tous les coéquipiers humains ou aux machines qu'il pouvait assembler. Ces "centaures", équipes mixtes d'ordinateurs et d'humains, ont dominé le tournoi.

Plus surprenant encore, le vainqueur du premier tournoi d'échecs de style libre à grande échelle n'était pas une équipe de grands maîtres et de superordinateurs, mais deux amateurs avec des ordinateurs portables. La composante tactique des échecs maîtrisée par des professionnels, une vie entière passée à remplir des livres d'ouverture et à s'entraîner avec des mentors pour gérer au mieux chaque attaque, a été rendue superflue par le format d'équipe mixte — chaque équipe pouvait compter sur une tactique sans faille.

Ce n'est que sur le plan de la stratégie que les équipes ont dû s'affronter. L'avantage tactique des grands maîtres ayant été anéanti, ce sont les amateurs qui l'ont emporté. En consultant quatre programmes d'échecs différents pour comparer en profondeur diverses options, ils ont élaboré des plans qui ont surmonté l'intuition des grands maîtres. La possibilité d'utiliser les ordinateurs non seulement pour les empêcher de faire des erreurs, mais aussi pour amplifier leur propre réflexion stratégique, a donné à l'équipe un avantage irrésistible. Kasparov a observé que "un processus faible humain + meilleure machine est supérieur à un ordinateur fort seul et, plus remarquable encore, supérieur à un processus fort humain + machine inférieure".

En tant que premier champion d'échecs en style libre, la défaite de Kasparov n'a pas été la fin du mythe, mais seulement le deuxième acte. Dans le troisième acte, la machine est vaincue — non par la force de la volonté, mais par la coopération.

Mais les échecs n'ont jamais été une question d'échecs. Les échecs ont toujours été une métaphore de l'intellect et de la pensée humaine. La défaite de Kasparov a été aussi importante que décevante ; malgré la conquête, on a peu appris sur la nature de l'intelligence. Et bien que l'on ait peu appris de Deep Blue, nous pouvons peut-être apprendre quelque chose sur le rôle de l'optimisation grâce aux centaures qui ont tué ses descendants.

L'optimisation est la recherche de solutions qui minimisent certains coûts. Le coût peut être défini au sens large afin d'englober toutes sortes de problèmes : des itinéraires qui minimisent la distance parcourue par les flottes de véhicules de livraison, aux valeurs de poids des réseaux neuronaux qui minimisent l'erreur de classification des chiffres manuscrits, en passant par la forme des ailes des avions.

Historiquement, la communauté de l'optimisation a abordé la conception avec le même rêve que ceux qui se sont attaqués aux échecs : remplacer les concepteurs par des outils qui peuvent résoudre comme par magie tout problème d'ingénierie mieux qu'un humain ne pourrait le faire. Mais malgré les années qui ont suivi ce rêve, les vrais succès sont rares. L'optimisation est capable de trouver de meilleurs paramètres ou d'affiner les conceptions existantes — mais sont loin de remplacer tout l'éventail des talents requis d'un concepteur.

Étonnamment, même lorsque les outils d'optimisation *sont* utilisés dans la conception, ils ne sont souvent pas utilisés de la manière prévue par leurs créateurs. L'optimisation est plus souvent utilisée au *début* du processus de conception qu'à la fin. Lorsque les équipes de centaures utilisent des programmes d'échecs pour explorer des ensembles de coups optimaux qui informent une grande stratégie, les concepteurs explorent des ensembles de conceptions optimisées pour découvrir les meilleurs concepts de conception. Dans ce manuscrit, nous soutenons que les outils d'optimisation suivront le même chemin que les programmes d'échecs : des outils destinés à remplacer les humains aux outils qui les rendent autonomes.

En design et en architecture, les expériences de ce type de collaboration homme-machine sont courantes. La diversité des contraintes, des parties prenantes, des préférences et des objectifs contradictoires dans les projets d'ingénierie, même mineurs, fait de leur "résolution" une tâche mal définie et impossible.

Plutôt que d'utiliser l'optimisation pour trouver une solution unique, les concepteurs utilisent l'optimisation pour produire une série de concepts. Ces concepts, qui satisfont à des contraintes générales et à des objectifs mal définis, aident les concepteurs à se faire une idée de la nature du problème.

Dans ce paradigme, connu sous le nom de conception générative, les projets sont générés par des algorithmes d'optimisation, et ces projets agissent comme des points de passage concrets dans un espace de conception complexe. Chaque conception représente des concepts et

des compromis différents et sert donc de vocabulaire aux concepteurs pour réfléchir et communiquer sur le problème. Des contraintes et des préférences supplémentaires peuvent être progressivement introduites et davantage de conceptions peuvent être générées de manière algorithmique dans le cadre d'une exploration de l'espace de conception dirigée par l'homme. En naviguant dans l'espace de conception à travers ce processus générateur, les concepteurs découvrent des concepts qu'ils affinent ensuite à la main pour produire un produit final.

Les concepteurs utilisent la conception générative pour explorer l'espace de conception à travers la lentille de divers concepts de conception performants. La variété des options est aussi importante que leur qualité — sans options diverses, il n'y a pas de place pour la "stratégie" de la pensée, de l'exploration et de l'intuition à long terme. Il est donc essentiel que les solutions produites par les algorithmes soient à la fois *haute qualité* et *diverses*.

Pour produire une variété de coups, les joueurs d'échecs en style libre utilisaient plusieurs programmes d'échecs ; la méthode algorithmique la plus fréquente pour produire des conceptions multiples est l'optimisation multi-objectifs (MOO). La MOO produit un ensemble de solutions optimales de Pareto qui représentent les compromis entre des objectifs opposés. Par exemple, l'ensemble de chassis pour quadricoptères qui sont à la fois solides et légers comprend un fil fragile qui relie à peine les composants à une extrémité, un morceau de matériau solide qui alourdit l'aéronef à l'autre extrémité, et les conceptions plus intéressantes, solides mais légères, entre les deux. Outre les solutions très performantes, ce processus d'optimisation permet de connaître l'espace de conception. L'analyse d'un ensemble de solutions hautement performantes peut donner un aperçu des principes de conception qui sous-tendent toutes les bonnes conceptions, et de la relation entre les objectifs.

Les algorithmes multi-objectifs sont utilisés pour l'analyse exploratoire, mais à bien des égards, ils ne sont pas bien adaptés. Une approche MOO nécessite de réduire au minimum la définition des objectifs, même lorsqu'une gamme serait plus utile : par exemple, pour trouver des comportements pour les avions à ailes battantes, nous nous intéressons aux mouvements qui produisent une gamme de vitesses, du vol stationnaire en place au vol à pleine vitesse.

La variété des solutions dans un front de Pareto n'apparaît que s'il y a un compromis entre les objectifs. Le fait d'exiger des objectifs contradictoires restreint considérablement les qualités que nous pouvons explorer. Dans de nombreux cas, les qualités que nous aimerions découvrir ne sont pas directement en conflit ou en corrélation avec d'autres objectifs, comme la taille des ailes de notre robot battant des ailes ou la ressemblance de son vol avec celui d'un oiseau.

La diversité de qualité (QD), une classe récente d'algorithmes évolutifs, pourrait mieux remplir ce rôle exploratoire. Les algorithmes de

QD produisent de multiples bonnes solutions, chacune étant qualitativement différente de la manière décrite par l'utilisateur. Pour un robot marcheur, cette variété pourrait prendre la forme de la fréquence d'utilisation de chaque jambe, pour un parcours de véhicule, des émissions produites, ou pour des propriétés esthétiques du contenu généré par la procédure, comme la symétrie. Les définitions de variation définies par l'utilisateur permettent plus qu'une sélection humaine de solutions variées, mais une *direction* pour cette variété. En couplant l'exploration et l'optimisation, QD trouve le meilleur exemple de chaque type de solution. QD produit une gamme de solutions plutôt qu'un front, créant ainsi un ensemble de solutions plus vaste et plus diversifié que MOO. Le potentiel de performance des différentes combinaisons de caractéristiques définies par l'utilisateur est révélé par la QD, et c'est pourquoi on l'appelle parfois *illumination* plutôt qu'optimisation.

Les techniques de recherche telles que la QD, qui recherchent la diversité, présentent des avantages qui vont au-delà de la recherche de solutions multiples. Lors de la recherche de solutions complexes, une recherche étroite et directe est souvent une mauvaise heuristique. Le chemin qui mène à la résolution de problèmes compliqués exige souvent de passer par des "étapes" intermédiaires qui ne ressemblent guère au but final. En recherchant de nombreuses façons d'aborder un problème, indépendamment des performances actuelles de l'approche, on peut souvent trouver une meilleure solution finale.

Si la QD a été utile pour les projets de création "humain dans la boucle", comme la génération de contenu procédural pour les jeux, les capacités de production de diversité se sont avérées particulièrement précieuses en robotique. L'illumination peut être utilisée pour créer des répertoires comportementaux, des catalogues de commandes motrices avec divers résultats qu'un autre algorithme sélectionne. Dans ce cas, les "statistiques" du mouvement sont précalculées par un algorithme de QD, et la "stratégie" du mouvement à utiliser est sélectionnée par un autre. Ces répertoires comportementaux peuvent contenir plusieurs façons d'exécuter la même action, permettant une adaptation rapide si un ou plusieurs composants échouent.

L'illumination d'un espace de recherche est un objectif plus ambitieux que l'optimisation d'une solution unique, et qui peut nécessiter beaucoup plus de calculs. Dans des travaux précédents, un répertoire comportemental a été créé pour un robot hexapode. Des comportements précalculés lui permettant d'avancer en utilisant n'importe quelle combinaison de jambes ont été trouvés avec QD, lui assurant une récupération rapide si l'une de ses jambes était endommagée. La production de ce répertoire a nécessité vingt millions d'évaluations dans un simulateur. Cette quantité de calculs est raisonnable lorsqu'un tel répertoire ne doit être créé qu'une seule fois, mais elle est bien trop lente pour un processus de conception interactif. Même si elle n'est effectuée qu'une seule fois pour obtenir une vue d'ensemble de

l'espace de conception, si l'évaluation d'une conception nécessite des simulations coûteuses, comme en aérodynamique où les simulations peuvent prendre des heures ou des jours, même un seul éclairage approfondi est irréaliste. Ce doctorat explorera plusieurs pistes pour faire de la QD un meilleur outil pour les concepteurs.

Dans les cas où même une seule évaluation est coûteuse, des modèles de substitution, c'est-à-dire des modèles prédictifs construits à l'aide de techniques d'apprentissage machine guidé par les données (ML – machine learning) sont couramment utilisés. Pour accélérer l'optimisation, un modèle de la fonction objectif est formé sur la base des solutions qui ont déjà été évaluées, et l'optimisation est effectuée sur la base de ce modèle. L'optima trouvé par cette optimisation basée sur le modèle est ensuite évalué sur la fonction objectif coûteuse, ce qui fournit de nouvelles données au modèle. L'optimisation se répète ensuite avec ce modèle amélioré, améliorant itérativement le pouvoir prédictif du modèle autour de l'optima. Les algorithmes d'optimisation actuels basés sur un modèle sont conçus pour trouver efficacement l'optima global, par opposition à la diversité des solutions que nous voudrions créer.

L'apprentissage automatique peut également être utilisé pour modéliser un ensemble de données existant et produire de nouvelles données appartenant à la même distribution. Les modèles générateurs ont rapidement gagné en intérêt en raison de leur capacité à créer des artefacts impressionnants, tels que des voix humaines et des images photoréalistes de personnes inexistantes, et de leur promesse d'aider et d'améliorer la créativité humaine.

Les modèles générateurs produisent de nouveaux artefacts par interpolation et échantillonnage au sein de la distribution des données qui ont été formées — liant la variété des artefacts qui peuvent être créés à la gamme des données sources. Un modèle génératif formé sur un ensemble de visages d'anime ne produira que des visages féminins si la fraction de visages masculins dans l'ensemble de données est trop faible. La production de modèles générateurs de haute qualité nécessite la conservation d'ensembles de données de qualité équivalente ; et contrairement à la diversité produite par la QD, la variété des solutions créées par les modèles générateurs ne présente aucun biais de performance ni d'exploration explicitement dirigée.

Dans ce manuscrit, nous présentons des travaux fondamentaux sur l'intégration de la QD et du ML. Les approches de ML et de QD sont à la fois contradictoires et complémentaires : trouver efficacement l'optimum global vs. explorer la diversité ; interpoler dans des ensembles de données connus selon des dimensions dérivées vs. générer des ensembles de données qui couvrent des dimensions définies par l'utilisateur ; apprendre à partir d'exemples vs. développer de nouvelles solutions.

Nous commençons dans la section 2.1 par examiner les méthodes existantes de production et de maintien de la diversité dans l'optimisation. Nous concentrons notre examen sur les méthodes évolutionnistes, une classe d'algorithmes basés sur la population pour laquelle le maintien de la diversité est un sujet de recherche de longue date. Notre résumé se termine par une discussion sur les algorithmes de QD, qui créent des exemples très performants basés sur des notions de diversité définies par l'utilisateur. Nous nous concentrons en particulier sur l'algorithme MAP-Elites, un algorithme d'illumination qui délimite l'espace de recherche en niches phénotypiques discrètes et trouve une solution performante qui remplit chacune d'entre elles.

Nous exploitons les capacités de l'apprentissage machine pour découvrir, distiller et exploiter les modèles sous-jacents dans la variété de solutions performantes produites par la diversité de la qualité. Deux lignes d'attaque sont principalement explorées : (1) améliorer l'efficacité des données de la QD avec des modèles prédictifs et (2) apprendre des représentations à partir des solutions produites par la QD avec des modèles génératifs.

Dans la section 2.2, nous donnons un bref aperçu d'une classe de modèles prédictifs utilisés pour accélérer l'optimisation, appelés modèles de substitution. Nous nous concentrons sur les techniques d'optimisation bayésiennes, qui construisent des modèles de la fonction objectif à partir de données, tout en fournissant des conseils sur les meilleurs points à échantillonner pour améliorer le modèle. Bien que ce processus d'apprentissage actif soit conçu pour trouver un optima global, nous cooptons ce paradigme pour trouver efficacement une diversité de solutions.

Dans la section section 2.3, nous présentons des techniques de modélisation générative basées sur les récents progrès de l'apprentissage machine. Ces modèles compriment de grands ensembles de données en un ensemble de dimensions latentes, semblables aux composants principaux. En recherchant et en échantillonnant dans cet espace latent, de nouvelles solutions peuvent être créées qui ressemblent à celles de l'ensemble de données original. En général, ces méthodes sont utilisées pour modéliser un ensemble de données existant - nous utiliserons des algorithmes de QD pour créer ces ensembles de données.

L'intégration des approches évolutionnistes et de ML exige que nous trouvions des moyens d'interfacer des paradigmes parfois éloignés. La technique d'évolution des réseaux de neurones très utilisée, la neuroévolution des topologies d'augmentation (NEAT), optimise simultanément la structure et les poids d'une population de réseaux de neurones. Ces réseaux ont été utilisés non seulement pour créer des contrôleurs et des classificateurs, mais aussi pour faire évoluer les conceptions. Pour effectuer une recherche efficace dans l'espace des topologies, une maintenance de diversité étendue est effectuée : les différentes topologies de réseau sont protégées pour garantir que la

recherche s'effectue dans de multiples directions et que les réseaux ont la possibilité de croître en complexité et d'atteindre leur plein potentiel.

Les topologies variables des réseaux entraînent des difficultés pour les modèles prédictifs, qui font généralement des prédictions basées sur une entrée de taille constante, comme un vecteur de valeurs. Dans le chapitre 3, nous présentons une méthode d'intégration des modèles prédictifs avec l'algorithme NEAT, qui fait évoluer la topologie de réseaux de neurones. Nous montrons que les informations généalogiques des réseaux, enregistrées à l'origine pour gérer la diversité, peuvent également être un guide efficace pour prédire les performances. Nous présentons un nouvel algorithme, le Surrogate-Assisted NEAT (SA-NEAT), construit autour d'un noyau qui tire parti de ces informations. Dans les tâches de contrôle de référence ; nous montrons que cette nouvelle variante est plusieurs fois plus efficace en termes de données que le NEAT.

Bien que NEAT protège la diversité dans l'espace génotypique (le codage des solutions), c'est la diversité dans l'espace phénotypique (la forme ou le comportement des solutions) qui est vraiment importante — surtout pour résoudre des problèmes complexes et "trompeurs". La recherche explicite de solutions diverses ne se limite pas à fournir des options à la fin du processus d'optimisation, elle améliore la qualité du résultat final. Le chemin vers des solutions complexes passe souvent par des "tremplins" intermédiaires qui donnent de mauvais résultats sur la tâche cible, mais qui sont néanmoins des étapes nécessaires vers un résultat performant. Sans aucun moyen de savoir quelles solutions peuvent conduire à des performances élevées, les diverses populations offrent de multiples voies d'optimisation à suivre.

Dans le chapitre 4, nous explorons les capacités des algorithmes de QD à résoudre des problèmes complexes en accumulant les étapes intermédiaires. Nous revisitons le défi de recréer les images produites par l'évolution pilotée par l'utilisateur, une illustration classique des limites de la recherche pilotée uniquement par l'optimisation d'un objectif. Nous montrons que la QD surpasse de loin la recherche basée sur un objectif en tentant de retrouver les images produites par des utilisateurs d'un algorithme d'évolution interactive, démontrant ainsi les avantages de la diversité dans l'optimisation. Au cours de cette étude exploratoire, les premières observations sont faites sur l'utilité des modèles générateurs de QD pour augmenter la diversité et la performance.

La création d'un ensemble diversifié de solutions à l'aide de la QD exige beaucoup de calculs, car nous résolvons un problème plusieurs fois de différentes manières. Mais alors que nous recherchons de nombreuses solutions différentes, chacune d'entre elles résout le même problème. Nous observons qu'un modèle unique de la fonction objectif peut être partagé tout en recherchant des solutions diverses.

Au chapitre 5, nous nous appuyons sur cette observation pour intégrer des techniques de modélisation prédictive avec MAP-Elites afin de créer l'algorithme SAIL (Surrogate-Assisted Illumination). Dans SAIL, les solutions évaluées sont utilisées pour construire un modèle de la fonction objectif, qui peut ensuite être utilisé pour approximer les performances des solutions non testées. En combinant l'optimisation basée sur un modèle avec la QD, nous montrons que les modèles de substitution peuvent compenser des évaluations qui sont non seulement coûteuses, mais aussi nombreuses. Dans deux domaines de la conception aérodynamique, SAIL améliore l'efficacité des données de la QD par des ordres de grandeur - élargissant radicalement la gamme des domaines où la QD peut être appliquée.

Au chapitre 6, nous nous intéressons à la modélisation générative et à la QD. La gamme de solutions produites par les modèles génératifs est limitée par les ensembles de données sur lesquels ils sont formés. Non seulement cela rend difficile la création de tout ce qui est vraiment nouveau, mais cela limite leur utilisation à des domaines où des ensembles de données bien conçus sont facilement disponibles. De plus, dans des domaines créatifs comme le design industriel et l'architecture, où la forme et la fonction sont importantes, les modèles générateurs souffrent de n'avoir aucun moyen d'assurer une haute performance, voire la faisabilité.

La variété des solutions de haute qualité créées par QD constitue une riche source de données pour les modèles générateurs - que nous exploitons pour produire de puissants codages spécifiques à un domaine. QD peut produire des ensembles de données sur mesure pour la formation de modèles générateurs, modèles qui peuvent ensuite être utilisés comme représentations pour l'optimisation. Les modèles générateurs reflètent les données utilisés dans leur formation - donc en produisant les données avec QD nous insérons dans ces représentations apprises un biais vers la haute performance.

Nous utilisons MAP-Elites pour créer un ensemble de données diversifiées et un autoencodeur variationnel pour créer un codage piloté par les données (DDE) qui capture l'essence des solutions à haute performance tout en codant un large éventail de solutions. Notre approche apprend ce DDE au cours de l'optimisation en trouvant un équilibre entre l'exploitation du DDE pour généraliser les connaissances contenues dans les archives actuelles des élites et l'exploration de nouvelles représentations qui ne sont pas encore capturées par le DDE. L'apprentissage de la représentation pendant l'optimisation permet à l'algorithme de résoudre des problèmes à haute dimension, et fournit une représentation à basse dimension qui peut ensuite être réutilisée. Dans des problèmes de robotique de référence, nous montrons que cette approche DDE nous permet de résoudre des problèmes à haute dimension (1000D) avec des évaluations inférieures d'un ordre de grandeur à l'état actuel de la technique. En plus d'une optimi-

sation améliorée, notre approche produit simultanément un codage réutilisable qui capture à la fois la haute performance et la gamme d'expression de la carte éclairée.

Dans le chapitre 7, nous concluons en discutant des limites des approches que nous proposons, des pistes prometteuses pour surmonter ces contraintes et des nouvelles pistes de recherche actuelles. Nous décrivons brièvement les points communs entre les algorithmes de QD basés sur des modèles que nous avons introduits, et nous décrivons les lignes d'attaque afin de réaliser plus complètement la vision de leur application à l'exploration de conception.

Dans ce manuscrit, nous présentons des algorithmes de diversité de qualité qui apprennent des modèles de l'objectif, permettant à MAP-Elites de trouver des milliers de modèles très performants après seulement l'évaluation de centaines, et qui apprennent des modèles de génotypes très performants, permettant l'optimisation de centaines de paramètres tout en ne recherchant que des dizaines. L'extension de l'applicabilité de MAP-Elites permet aux domaines de l'ingénierie du monde réel d'atteindre une diversité de qualité. Mais plus important que les succès de ces techniques spécifiques est la validation d'une intuition sous-jacente sur l'apprentissage machine et la diversité de qualité : QD et ML sont loin d'être des paradigmes opposés, mais des approches complémentaires dont l'intégration plus profonde peut débloquer de puissantes capacités dans les deux domaines.

PUBLICATIONS

Some figures, text, and ideas in this section are based on the following works which have either been previously published or are in review:

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2017a). "Aerodynamic design exploration through surrogate-assisted illumination." In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, p. 3330.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2017b). "Data-efficient exploration, optimization, and modeling of diverse designs through surrogate-assisted illumination." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 99–106.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2018a). "Data-efficient design exploration through surrogate-assisted illumination." In: *Evolutionary computation* 26.3, pp. 381–410.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2018b). "Data-efficient neuroevolution with kernel-based surrogate models." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 85–92.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2019). "Are quality diversity algorithms better at generating stepping stones than objective-based search?" In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 115–116.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2020). "Discovering Representations for Black-box Optimization." In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. Vol. 11.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

To glimpse the future of optimization in design, we might look back at the history of chess, a domain already "conquered" by computers. In 1997 Garry Kasparov, the world chess champion of the world, lost to Deep Blue, the IBM super computer, in an event that has been retold often enough to take on the aura of myth *(Campbell et al., 2002)*. The tournament has become a well worn parable of the inevitable dominance of technological force over human talent. From that point on, the story concludes, no human could ever again hope to defeat a machine in chess.

Seven years later, the best chess playing machines in world were beaten — badly.

No new chess wunderkind appeared as the savior of humanity. The most advanced chess machine ever created *(Donninger and Lorenz, 2004; ChessBase, 2005b)*, a blend of top algorithms and custom built hardware, was completely outclassed by a field of opponents *(ChessBase, 2005c)*. But it was not human cunning which made the difference, but a leveling of the playing field. In what was dubbed a "freestyle" chess tournament every competitor had access to whatever human teammates or machines they could assemble. These "centaurs", mixed teams of computers and humans, dominated the tournament.

More surprising still, the winner of the first large scale freestyle chess tournament was not a team of grandmasters and supercomputers, but a pair of amateurs with laptops *(ChessBase, 2005a)*. The tactical component of chess mastered by professionals, a lifetime pouring over books of opening moves and training with mentors to best handle every skirmish, was made redundant by the mixed team format — every team could count on flawless tactics.

It was only in strategy that the teams had to compete. With the grandmasters' tactical advantage wiped away it was amateurs who carried the day. By consulting four different chess programs to deeply compare a variety of options they developed plans which overcame the grandmasters' intuition. The ability to use the computers not only to prevent them from making mistakes, but to amplify their own strategic thinking, gave the team an irresistible edge. Kasparov observed that, "weak human + machine + better process was superior to a strong computer alone and, more remarkably, superior to a strong human + machine + inferior process."*(Kasparov, 2010)*

As an early champion of freestyle chess, Kasparov's defeat was not the end of the myth, only the second act. In the third act the machine *is* overcome — not through force of will, but cooperation.

But chess has never been about chess. Chess has always been a totem standing in for human intellect and thought *(Berliner, 1974; Brooks, 1990; McCarthy, 1990)*. The defeat of Kasparov was as momentous as it was anticlimactic; despite the conquest little was learned about the nature of intelligence. And though little may have been learned from Deep Blue, perhaps we can learn something about the role of optimization from the centaurs who slew its descendants.

Optimization is the search for solutions that minimize some cost. Cost can be broadly defined in order to encompass all manner of problems: from routes which minimize the distance traveled by fleets of delivery vehicles *(Bell and McMullen, 2004)*, to neural network weight values that minimize classification error of handwritten digits *(LeCun et al., 1998)*, to the shape of low drag airplane wings *(Hicks et al., 1974)*.

Historically the optimization community has approached design with the same dream as those that tackled chess: to replace designers with tools that can magically solve any engineering problem better than a human possibly could. But despite years following that dream, true successes are rare. Optimization is capable of finding better parameters *(Yang and Deb, 2013)* or fine-tuning existing designs *(*Generative Design at Airbus: Customer Stories *2020)* — but are far from replacing the full spectrum of talents required of a designer.

Surprisingly, even when optimization tools *are* used in design, they are often not used in the way their creators intend. Optimization is more commonly used at the *beginning* of the design process than the end *(Bradner et al., 2014b)*. Where centaur teams use chess programs to explore sets of optimal moves that inform a grand strategy, designers explore sets of optimized designs to discover the best design concepts *(Bradner et al., 2014b; Matejka et al., 2018)*. In this manuscript we contend that optimization tools will follow the same path as chess programs: from tools to replace humans to tools that empower them.

In design and architecture, experiments with this kind of human-machine collaboration are common *(Holzer et al., 2007; Turrin et al., 2011; Gerber et al., 2012; Arieff, 2013; Nagy et al., 2017; Matejka et al., 2018)*. The variety of conflicting constraints, stake-holders, preferences, and objectives in even minor engineering projects makes "solving" them an ill-defined and impossible task.

Rather than using optimization to find a single solution, designers instead use optimization to produce a range of designs concepts. These designs, which satisfy broad constraints and loosely defined objectives, help designers build an intuition about the nature of the problem *(Holzer et al., 2007; Tsigkari et al., 2013; Bradner et al., 2014b)*. In this paradigm, known as generative design, draft designs are generated by optimization algorithms, and these designs act as concrete way points in a complex design space. Each design represents different concepts and trade-offs and so act as a vocabulary for designers to think and communicate about the problem *(Flager et al., 2009)*. Additional

constraints and preferences can be gradually introduced and more designs algorithmically generated in a human-led exploration of the design space. By navigating the design space through this generative process designers discover concepts which they then further refine by hand to produce a final product.

Designers use generative design to explore the design space through the lens of varied high performing design concepts. The variety of options is as important as their quality — without diverse options there is no room for the "strategy" of long range thinking, exploration, and intuition. It is critical then that the solutions produced by algorithms are of both *high-quality* and *diverse*.

To produce a variety of moves the freestyle chess players used multiple chess programs; the most frequency applied algorithmic method to produce multiple designs is multi-objective optimization (MOO) *(Deb et al., 2002; Hamdaoui et al., 2010; Gaspar-Cunha et al., 2011; Gerber et al., 2012; Druot et al., 2013; Deb, 2014; Matejka et al., 2018)*. MOO produces a set of Pareto optimal solutions which represent the trade-offs between opposing objectives. For example the Pareto set of quadcopter frames that are both strong and light includes a flimsy wire that barely connects the components at one extreme, a solid piece of material which weighs down the aircraft at the other, and the more interesting strong but lightweight designs between the two. In addition to high-performing solutions, knowledge of the design space can be gathered through this optimization process. Analyzing a set of high performing solutions can yield insights into design principles that underlie all good designs, and the relationship between objectives *(Deb and Srinivasan, 2006; Doncieux et al., 2015b)*.

Multi-objective algorithms are used for exploratory analysis, but in many ways are not well suited for it. A MOO approach requires the definition of objectives to be minimized, even when a range would be more useful: to find behaviors for flapping wing aircraft we are interested in movements which produce a range of speeds, from hovering in place to full speed flight *(Doncieux et al., 2015b)*. The variety of solutions in a Pareto front only appear if there is a trade-off between objectives *(Deb, 2003)*. Requiring conflicting objectives sharply constricts the qualities we can explore. In many cases the qualities we would like to discover are not directly in conflict or correlated with other objectives, such the wing size of our flapping robot or how closely its flight resembles that of a bird.

Quality diversity (QD) *(Lehman and Stanley, 2011b; Cully et al., 2015; Mouret and Clune, 2015; Pugh et al., 2016; Cully and Demiris, 2017)*, a recent class of evolutionary algorithms, could better fulfill this exploratory role. QD algorithms produce multiple good solutions, each qualitatively different in ways described by the user. For a walking robot this variety could take the form of how often each leg is used *(Cully et al., 2015)*, for a vehicle route the produced emis-

sions *(Urquhart and Hart, 2018)*, or for procedurally generated content aesthetic properties like symmetry *(Alvarez et al., 2019)*. User-defined definitions of variation allow more than human selection of varied solutions, but user *direction* of that variety. By coupling exploration and optimization QD finds the best example of each type of solution. QD produces a range of solutions rather than a front, creating a larger and more diverse set of solutions than MOO. The performance potential of different combinations of user-defined features is revealed by QD, and so it is sometimes referred to as *illumination* rather than optimization *(Mouret and Clune, 2015)*.

Search techniques such as QD which look for diversity have benefits beyond finding multiple solutions. When looking for complex solutions a narrow and direct search is often a poor heuristic *(Woolley and Stanley, 2011)*. The path to solving complicated problems often requires traveling along intermediate "stepping stones" which have little resemblance to the final goal *(Stanley and Lehman, 2015)*. By searching for many ways to approach a problem, regardless of the approach's current performance, a better final solution can often be found *(Lehman and Stanley, 2011a)*.

While QD has been useful for human-in-the-loop creative endeavors like procedural content generation for games *(Alvarez et al., 2019)*, the diversity producing capabilities have proven especially valuable in robotics. Illumination can be used to create behavioral repertoires, catalogs of motor commands with various outcomes which another algorithm selects from *(Cully et al., 2015; Ecarlat et al., 2015; Engebråten et al., 2018; Nordmoen et al., 2018a)*. In this case, the "tactics" of movement are precomputed by a QD algorithm, and the "strategy" of which movement to use is selected by another. These behavioral repertoires can contain many ways of performing the same action, allowing rapid adaption if one or more component fails *(Cully et al., 2015; Chatzilygeroudis et al., 2018; Kaushik et al., 2020a)*.

Illumination of a search space is a more ambitious goal than optimization of a single solution, and one that can require much more computation. In *(Cully et al., 2015)* a behavioral repertoire was created for a hexapod robot. Precomputed behaviors that allowed it to move forward using any combination of legs were found with QD, ensuring that it could rapidly recover if any of its legs were damaged. Producing that repertoire required twenty million evaluations in a simulator. That amount of computation is reasonable when such a repertoire only has to be created once, but is far too slow for an interactive design process. Even if only performed once to get an overview of the design space, if evaluation of a design requires expensive simulations, such as in aerodynamics where simulations can take hours or days, even a single thorough illumination is unrealistic. This PhD will explore several avenues of making QD a better tool for designers.

In cases where even a single evaluation is expensive, surrogate models *(Forrester and Keane, 2009; Brochu et al., 2010; Shahriari et al., 2015; Stork et al., 2018)*, predictive models build using data-driven machine learning (ML) techniques are commonly used *(Hasenjäger and Sendhoff, 2005; Giannakoglou et al., 2006; Zhou et al., 2007; Dumas, 2008; Lian et al., 2010)*. To accelerate optimization a model of the objective function is trained based on the solutions which have already been evaluated, and optimization is performed based on this model. The optima found by this model-based optimization is then evaluated on the expensive objective function, providing new data to the model. Optimization then repeats with this improved model, iteratively improving the predictive power of the model around optima. Current model-based optimization algorithms are geared to efficiently find the global optima, as opposed to the diversity of solutions we would like to create.

ML can also be used to model an existing data set and produce new data that belongs to the same distribution. Generative models *(Kingma and Welling, 2013; Goodfellow et al., 2014; Oord et al., 2016; Kingma and Dhariwal, 2018)* have rapidly gained interest for their ability to create impressive artifacts, such as human sounding voices *(Oord et al., 2016)* and photorealistic images of non-existent people *(Karras et al., 2019b; Wang, 2020)*, and their promise to aid and enhance human creativity *(Carter and Nielsen, 2017)*.

Generative models produce novel artifacts by interpolating and sampling within the data distribution that were trained — binding the variety of artifacts that can be created to the range of the source data. A generative model trained on a set of anime faces will produce only female faces if the fraction of male faces in the data set is too small *(Jin et al., 2017)*. Producing high quality generative models requires curation of equally high quality datasets; and unlike the diversity produced by QD, the variety of solutions created by generative models has none of the performance bias or explicitly directed exploration.

In this manuscript we present foundational work on the integration of QD and ML. ML and QD approaches are at once contradictory and complementary: finding global optima efficiently vs. exploring diversity; interpolating within known datasets along derived dimensions vs. generating datasets that span user-defined dimensions; learning from examples vs. evolving new solutions.

We begin in Section 2.1 by examining existing methods of producing and maintaining diversity in optimization. We focus our review on evolutionary methods, a class of population-based algorithms for which diversity maintenance has been a long researched topic. Our summary culminates in a discussion of QD algorithms, which create high performing examples based on user-defined notions of diversity. We focus in particular on Multidimensional Archive of Phenotypic Elites (MAP-Elites) *(Mouret and Clune, 2015)*, an illumination algorithm

which demarcates the search space into discrete phenotypic niches and finds a high performing solutions that fills each.

We leverage the capabilities of machine learning to discover, distill, and exploit the underlying patterns in the variety of high performing solutions produced by quality diversity. Two lines of attack are chiefly explored: (1) improving the data-efficiency of QD with predictive models and (2) learning representations from the solutions produced by QD with generative models.

In Section 2.2 we provide a brief overview of a class of predictive models used to accelerate optimization known as *surrogate models*. We focus on Bayesian optimization techniques, which build models of the objective function based on data, while also providing guidance as to the next best points to sample to improve the model. Though this active learning process is designed to find a global optima, we will co-opt this paradigm to efficiently find a diversity of solutions.

In Section 2.3 we introduce generative modeling techniques based on recent machine learning advances. These models compress large datasets to a set of latent dimensions, akin to principal components. By searching and sampling within this latent space new solutions can be created which resemble those in the original dataset. Typically these methods are used to model an existing dataset — we will use quality diversity algorithms to *create* these datasets.

Integration of evolutionary and ML approaches requires that we find ways of interfacing what are sometimes distant paradigms. The widely-used neural network evolution technique Neuroevolution of Augmenting Topologies (NEAT) *(Stanley and Miikkulainen, 2002)* simultaneously optimizes both the structure and weights of a population of neural networks. These networks have been used not only to create controller and classifiers, but also to evolve designs *(Clune and Lipson, 2011; Clune et al., 2013; Richards and Amos, 2014)*. To effectively search the space of topologies extensive diversity maintenance is performed: differing network topologies are protected to ensure that the search proceeds in multiple directions, and that networks have the opportunity to grow in complexity and reach their full potential.

The variable topologies of the networks produces difficulties for predictive models, which typically make predictions based on a input of constant size, such as a vector of values. In Chapter 3 we present a method of integrating predictive models with NEAT. We show that the genealogical information of the networks, originally recorded to manage diversity, can also be an effective guide to predict performance. We introduce a new algorithm, Surrogate-Assisted NEAT (SA-NEAT), built around a custom kernel that leverages this insight. In benchmark control tasks we show this new variant is several times more data-efficient than NEAT.

Though NEAT protects diversity in the genotype space, the encoding of the solutions, it is diversity in phenotype space, the form

or behavior of the solutions, which is truly important — especially for solving complex and deceptive problems. Explicitly searching for diverse solutions does more than provide options at the end of the optimization process, it improves the quality of the end result. The path to complex solutions often passes through intermediate "stepping stones" which perform poorly on the target task, but are nevertheless necessary steps toward the high performing result. With no way to know which solutions may lead to high performance, diverse populations provide multiple paths for optimization to follow.

In Chapter 4 we explore the capabilities of QD algorithms to solve complex problems by accumulating stepping stones. We revisit the challenge of recreating images produced by user-driven evolution, a classic illustration of the limits of search driven only by objective optimization. We show that QD far outperforms objective-based search in matching user-evolved images, demonstrating the advantages of diversity in optimization. In the course of this explorative study first observations are made about the usefulness of generative models in QD to increase diversity and performance.

Creating a diverse collection of solutions with QD is computationally intensive because we solve a problem many times in many different ways. But while we are looking for many different solutions, each are solving the same problem. We observe that a single model of the objective function can be shared while searching for diverse solutions.

In Chapter 5 we leverage this insight to integrate predictive modeling techniques with MAP-Elites to create the Surrogate-Assisted Illumination (SAIL) algorithm. In SAIL evaluated solutions are used to build a model of the objective function, which can then be used to approximate the performance of untested solutions. By combining model-based optimization with QD we show that surrogate models can compensate for evaluations that are not only expensive, but also numerous. In two aerodynamics design domains SAIL improves the data-efficiency of QD by orders of magnitude — radically expanding the range of domains where QD can be applied.

In Chapter 6 we turn our attention to generative modeling and QD. The range of solutions produced by generative models are limited by the data sets they are trained on. Not only does this make it difficult for the creation of anything truly novel, but restricts their use to domains where well-curated datasets are readily available. Moreover, in creative fields like industrial design and architecture, where both form and function are important, generative models suffer from having no way of ensuring high performance, or even feasibility.

The variety of high quality solutions created by QD are a rich data source for generative models — which we exploit to produce powerful domain specific encodings. QD can produce bespoke datasets for training generative models, models which can then in turn be used as representations for optimization. Generative models mirror the data

used in their training — so by producing the data with QD we insert into these learned representations a bias toward high performance.

We use MAP-Elites to create a diverse data set and a Variational Autoencoder to create a Data-Driven Encoding (DDE) which captures the essence of high-performing solutions while still encoding a wide array of solutions. Our approach learns this DDE during optimization by balancing between exploiting the DDE to generalize the knowledge contained in the current archive of elites and exploring new representations that are not yet captured by the DDE. Learning representation during optimization allows the algorithm to solve high-dimensional problems, and provides a low-dimensional representation which can be then be re-used. In benchmark robotics problems we show that this DDE approach allows us to solve high dimensional (1000D) problems with orders of magnitude fewer evaluations than the current state-of-the-art. In addition to improved optimization, our approach simultaneously produces a reusable encoding that captures both the high performance and range of expression of the illuminated map.

In Chapter 7 we conclude by discussing the limitations of our proposed approaches, promising avenues for overcoming these constraints, and the new avenues for research the present. We briefly describe commonalities across the model-based QD algorithms we introduced, and outline lines of attack to more fully realize the vision of their application to design exploration.

2 BACKGROUND

Some figures, text, and ideas in this section are based on the following works which have either been previously published or are in review:

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2017a). "Aerodynamic design exploration through surrogate-assisted illumination." In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, p. 3330.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2017b). "Data-efficient exploration, optimization, and modeling of diverse designs through surrogate-assisted illumination." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 99–106.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2018a). "Data-efficient design exploration through surrogate-assisted illumination." In: *Evolutionary computation* 26.3, pp. 381–410.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2018b). "Data-efficient neuroevolution with kernel-based surrogate models." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 85–92.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2019). "Are quality diversity algorithms better at generating stepping stones than objective-based search?" In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 115–116.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2020). "Discovering Representations for Black-box Optimization." In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. Vol. 11.

2.1 QUALITY DIVERSITY

2.1.1 *Evolving Diversity*

While evolutionary approaches are often employed like any other optimization algorithm, to find a single high performing solution, their population-based nature makes them well-suited for producing a variety of solutions. Such diverse collections of diverse solutions are valuable beyond finding a global optimum.

In industrial design optimization algorithms are often used to better understand the constraints and possibilities inherent in a design space *(Bradner et al., 2014a)*. Producing a range of design alternatives

allows engineers to explore different concepts and trade-offs, and by analyzing a collection of diverse and high performing solutions the underlying properties responsible for high performance can be discovered *(Deb and Srinivasan, 2006)*.

Producing a set of good solutions rather than a solitary optima ensures robust optimization results. Multiple solutions act as insurance, a host of back up solutions in the case that the optimal solution does not perform as hoped. Solutions may optimize the objective function in unexpected and useless ways *(Lehman et al., 2018)*; solutions optimized in simulation may not transfer as expected to reality *(Koos et al., 2013)*; or the situation they were designed for may change during operation *(Cully et al., 2015)*.

In many cases the solution to a problem *is* a collection of solutions. This is especially true in the case of learning behavioral repertoires for robots *(Cully and Mouret, 2013)*. Such repertoires can define gaits *(Cully and Mouret, 2016)*, manipulator primitives *(Cully and Demiris, 2018)*, or produce a single controller that solves many tasks, such as tossing a ball at varied targets *(Kim and Doncieux, 2017)* or moving to designated location *(Duarte et al., 2016)*.

Even when only a single final solution is required from an algorithm, fostering diversity is beneficial. A diverse population is able to avoid converging on local minimal, avoid deception, and find high performing solutions which may only be found by adapting worse performing "stepping stones" *(Lehman and Stanley, 2011a)*.

DIVERSITY IN GENOTYPE SPACE Classic methods of diversity maintenance in evolutionary algorithms *(Mahfoud, 1995)* focus on maintaining and creating diversity in the *genotype*, that is in the vectors, strings, or other data structures which encode the solution and are acted on by recombination operators. These genotypic diversity maintenance strategies include niching and local competition tactics such as fitness sharing *(Goldberg and Richardson, 1987)*, clustering *(Yin and Germay, 1993)*, restricted tournament selection *(Harik, 1995)*, clearing *(Pétrowski, 1996)*, and speciation *(Li et al., 2002; Stanley and Miikkulainen, 2002)*. Through different mechanisms these methods bias search to produce populations which are both high performing and spread across the genotype space.

These multimodal optimization techniques excel at finding the local optima in a search space *(Das et al., 2011)*, but local optima do not necessarily translate to interesting diversity. Basing diversity on the encoding of solution may not translate into an interesting diversity of ways of solving the problem, similar solutions can be encoded in different ways. A tour of the traveling salesman problem and its reverse may be very distant genotypically but represent identical solutions. Basing diversity on performance is also no guarantee of interesting diversity. Interesting ways of solving the problem are not

necessarily local optima of the fitness landscape — variety is often only vaguely related to performance.

DIVERSITY IN OBJECTIVE SPACE In multi-objective optimization (MOO) individuals are optimized to satisfy two or more objectives simultaneously *(Deb, 2014)*. A solution is considered optimal, or non-dominated, if no other solution performs better on *all* objectives. When the objectives are in conflict, such as minimizing the weight of a bridge while maximizing its structural strength, the optimal results take the form of a Pareto set of solutions which each represent a different trade-off of the objectives *(Deb, 2003)*.

MOO algorithms not only search for non-dominated solutions, but for a set of non-dominated solutions which span the range of possible trade-offs: the lightest bridge, the strongest bridge, and the strongest bridges for their weight. In NSGA-II *(Deb et al., 2002)*, one of the mostly widely used MOO algorithms, this spread is accomplished by ranking solutions not only by dominance but by crowding distance *(Horn et al., 1994)*, a niching strategy. Critically, this crowding distance is not calculated in genotype space, but in the objective space. Defining diversity based on the phenotype of the solution rather than its encoding ensures an even spread of solutions in the space we actually care about — the space of trade-offs.

In order to foster the phenotypic diversity produced by MOO, single objective problems can be redefined as multi-objective problems. This *multiobjectivization (Knowles et al., 2001)* can be done in one of two ways. The first is decompose the original objective into multiple objectives, for instance by dividing a traveling salesman problem into several independent tours to be solved independently or by separating out the terms of a mathematical formulation and optimizing each *(Handl et al., 2008)*. The other is to add additional "helper" objectives to the original objective *(Jensen, 2004)*. These objectives optimize some other measure unrelated to performance such as the type of cross sections in a structural frame *(Greiner et al., 2007)*, the age of individuals in the population *(Schmidt and Lipson, 2011)*, the modularity of evolved networks *(Mouret and Doncieux, 2009)*, or the similarity to previously found solutions *(Mouret, 2011)*. Useful diversity is found in MOO approaches by finding a variety of phenotypes, and multiobjectivization expands on this capability by allowing the nature of this variety to be orthogonal to performance and to be freely defined.

Ensuring diversity in the objective space brings us much closer to finding an interesting variety of solutions, but the type of diversity is wholly determined by the trade-offs and interactions of the objectives. If there is little conflict in the objectives, then all solutions will be variations on a similar theme: optimizing for a bridge's strength and *maximizing* its weight will result only in a solid block of material.

DIVERSITY IN PHENOTYPE SPACE Helper objectives in multiobjectivization are typically descriptive rather than directly related to performance. In cases where they embody some intuition or heuristic: modular networks will be more successful, or choosing newer individuals over older will prevent stagnation. These phenotypic descriptors can also be characteristics we would like to explore, rather than minimize. If designing a flying robot with flapping wings we may not be interested only in the smallest wingspan or the largest, but in the whole range of wing sizes.

The motivation to explore a range is especially strong in cases where the goal is to find multiple solutions. If the phenotype of a robot arm is described by the position of its end effector it makes little sense to maximize the description — we are looking for is a set of solutions that can reach *any* point, not a maximal point.

Exploration of user-defined phenotypic descriptors is such a powerful paradigm it can be successful even without optimization of objectives. Novelty search (NS) *(Lehman and Stanley, 2011a)* eschews traditional objective optimization, and instead acts as a space-filling algorithm in a phenotype space defined by the user *(Doncieux et al., 2019)*. When an individual is evaluated, rather than calculating a fitness value, a phenotypic descriptor is returned. This descriptor is typically a vector of real-valued numbers which situates the solution in a phenotype space [1]. In a mobile robot this descriptor could be a vector the describes the robot's body plan *(Corucci et al., 2015)*, trajectory *(Pugh et al., 2016)*, or end position *(Lehman and Stanley, 2011a)*.

The most "novel" individuals, those whose location in phenotype space are farthest from those individuals already created, are selected for. While an algorithm employing objective-based fitness converges on local optima, NS spreads out across the phenotype space and covers it evenly. In this view NS can be understood as a space filling algorithm, and has been shown to tend toward a uniform distribution of the reachable phenotype space *(Doncieux et al., 2019)*. In this context space filling is interesting and useful because the phenotype space, just like the objective space, cannot be directly sampled.

In the classic NS experiment *(Lehman and Stanley, 2011a)* a robot is placed at the start of a 2D maze and must find its way to the end. The robot operates by detecting walls with range finders whose readings serve as input to a reactive neural network controller, with the phenotype defined as the end position in (x,y) coordinates after a number of time steps. Rather than selecting individuals who end closest to the goal, a doomed objective-minimization strategy because of the deceptive nature of the maze, individuals are selected for ending

1 Novelty Search and Quality Diversity methods were first developed in the context of Artificial Life and Evolutionary Robotics and so it is common in the literature to refer to the phenotype as "behavior" and to refer to "behavior space", "behavior characterization", or "behavioral descriptor". Having since been applied to a variety of use cases, we adopt the more general terminology of "phenotype" and "descriptor".

in a part of the maze which has not yet been explored. By disregarding the objective of finding the goal and instead rewarding exploration of new regions of the phenotype space NS finds controllers which reach the end of the maze.

Space filling is effective only because it is done in the low-dimensional space we are interested in exploring. Performing the same kind of space filling in genotype space would have far from the same result. In a simplified version of this classic maze problem — where a fixed topology network with discrete weights was used so every possible individual could be computed — of the 49 million possible genomes only 53 were able to reach the goal area *(Lehman and Stanley, 2018)*. It is only by leveraging the idea of a low-dimensional phenotype space that space filling becomes a useful search strategy.

Despite its non-intuitive nature novelty search has achieved impressive results in variety of domains ill-suited to objective based search *(Risi et al., 2010; Kistemaker and Whiteson, 2011; Lehman and Stanley, 2011a; Gomes and Christensen, 2013; Martínez et al., 2013; Naredo and Trujillo, 2013; Corucci et al., 2015; Liapis et al., 2015; Overbury and Berthouze, 2015; Conti et al., 2018)*, in many of the problems where NS can help us to find a range of diversity, we are *also* interested in finding optimal solutions.

DIVERSITY OF PHENOTYPIC ELITES Quality diversity (QD) *(Lehman and Stanley, 2011b; Cully et al., 2015; Mouret and Clune, 2015; Pugh et al., 2016; Cully and Demiris, 2017)* algorithms combine the phenotypic space filling approach of novelty search with fitness-based optimization. QD algorithms gather solutions which vary based on their distance in the descriptor space, as in NS, but require solutions to compete for a place in the population with their phenotypic neighbors base on fitness. In this way QD algorithms search not only for a variety of solutions, but the best performing example of each type. QD is discussed in more depth in Section 2.1.2.

SUMMARY Methods of maintaining and inducing diversity in evolutionary algorithms by biasing selection toward less crowded regions are long established. The most ground-breaking innovations have come not from developing new techniques for crowding or niching, but in reinterpreting the space that these mechanisms operate in. Niching techniques promote diversity in the genotype space, while multiobjective approaches search for a well spaced Pareto front in the objective space. Multiobjectivization fabricates entirely new objectives to create diversity in the characteristics of the solutions, while novelty search overlays these new dimensions of variation over the search space in place of the objective and produces solutions across their entire range. Quality diversity methods produce solutions across the

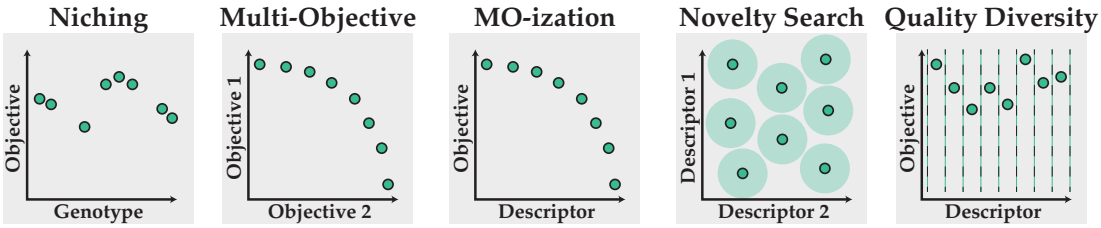

Figure 2.1: *Approaches to Discovering Diversity.*
Niching: Solutions which span the genotype space are maintained, with areas of higher fitness supporting more individuals.
Multiobjective: An even spread of individuals that span the Pareto front of optimized objectives is produced.
Multiobjectivization: Individuals are produced which optimize the problem objective and an orthogonal "helper objective" or user-defined descriptor.
Novelty Search: A set of individuals which span the range of values for one or more descriptors is found.
Quality Diversity: The best performing individuals are found which span the range of one or more descriptors.

entire range of user-defined dimensions, and find the best performing example of each. These approaches are illustrated in Figure 2.1.

2.1.2 *Quality Diversity Algorithms*

The goal of Quality Diversity (QD) algorithms is two-fold: to produce a collection of solutions that are high performing (*Quality*) and which cover a descriptor space as completely as possible (*Diversity*). QD is a marriage of novelty search and objective-based optimization. The same descriptor space filling strategy used by NS is employed, but with the acknowledgment that not every solution that occupies the same niche of descriptor space is equal. Once a region of descriptor space is discovered competition to occupy that niche ensues to find the highest fitness individual.

Several variants of QD have been proposed, but all are composed of a *container* which gathers and orders solutions into a collection, and all follow the same general procedure *(Cully and Demiris, 2017)*:

1. A selection operator selects the parents from the population
2. Children are created based on those parents
3. Children are evaluated to obtain a fitness score and descriptor
4. Children are added to the container, based on the solutions which already exist in the container

QD approaches can be broadly divided into two families exemplified by Novelty Search with Local Competition (NSLC) *(Lehman and Stanley,*

2011b) and Multidimensional Archive of Phenotypic Elites (MAP-Elites) *(Mouret and Clune, 2015)*.

The core difference between these two approaches is the form of their container, which defines how the descriptor space is organized. Both NSLC and MAP-Elites rely on localized competition of individuals within niches, and the container defines where these niches are located.

The container of MAP-Elites is very explicit: the n-dimensional descriptor space is divided into discrete bins, or niches, at the start of the run. In contrast the container in NSLC is continuous and unstructured. In NSLC when a new child is evaluated it is assigned a *novelty score*, calculated as the average euclidean distance to its k-nearest neighbors, and if that novelty score is above a threshold it is added to the archive. The descriptor space is organized in an emergent fashion around the collected individuals, and so expands as new individuals are added to the archive.

The ad hoc nature of the NSLC container means that the density of solutions across the descriptor space is likely to be uneven *(Cully and Mouret, 2013; Cully and Mouret, 2016)*, and to vary from run to run. In contrast, the grid structure of MAP-Elites allows a uniform coverage and distribution of solutions to be enforced. Such explicit definition allows for easier analysis of repeated results, and uniform density of solutions provides a more reliable "map" of the descriptor spaces' relationship to fitness. Still, the emergent nature of the NSLC archive is appealing, as it requires less prior knowledge of the structure of the domain or the bounds of the descriptor space.

NSLC and MAP-Elites interact with their archives differently. MAP-Elites is a steady state evolutionary algorithm which maintains a population, in the form of the archive, that is gradually improved. Parent solutions are selected from the archive at random and used to create new child solutions. When a new child is evaluated it is assigned a bin based on its descriptor. If the bin is empty, the individual is placed inside. If the bin is occupied the new individual and the current occupant are compared, and the fittest solutions individual is placed in the bin. Because of the elitist nature of joining the population, the MAP-Elites archive serves as both a source of parent solutions and as a record of the best solutions ever found in each niche.

NSLC uses a generational approach, recreating the population at each iteration. Parents are chosen based on their ranking in a Pareto front based on two criteria, *novelty*, euclidean distance to their k-nearest neighbors, and *local competition*, fitness compared to their k-nearest neighbors. In order to prevent the population from "cycling", or rediscovering the same regions repeatedly, the nearest neighbors taken into consideration in NSLC include both the current population and the archived solutions.

Interactions in NSLC between a changing population and archive, multiple non-stationary objectives whose value changes at each generation, and a host of hyperparameters, can create unstable and complex dynamics. With very large archives or populations performance can begin to be an issue as well: nearest neighbor calculations are $O(n \log(n))$ at each generation *(Friedman et al., 1977)*, while in the case of MAP-Elites looking up a bin address is $O(1)$.

Measuring the performance of quality diversity algorithms can be difficult, but two broad qualities are measured: the coverage of the descriptor space, and the performance of the found solutions. We can measure the coverage of the descriptor space in MAP-Elites by the number of filled bins, and in NSLC by the number of individuals in the archive.

Obvious single measures of performance, such as the highest performing solutions or the mean performance of all solutions, are problematic. Judging a QD algorithm by the single best solution makes little sense if the goal is to find many good solutions. QD algorithms solve what is in essence a multi-objective problem, and so defining a single metric to judge QD algorithms is a flawed exercise. Judging performance by the mean fitness is more in the spirit of QD, but can be gamed — algorithms which only explore high fitness regions will have a higher mean than algorithms which explore the same high fitness regions *and* low fitness regions, precisely the outcome we are trying to reward. QD Score *(Pugh et al., 2016)*, the mean fitness multiplied by coverage, attempts to address this, but does not avoid weighting high performing regions more than low performing regions. In MAP-Elites, because of the fixed descriptor space, this weighting problem can be remedied: the fitness in each bin can be normalized by the highest fitness ever found for that bin. This normalization treats performance in each bin equally, regardless of the fitness potential each bin, but still implicitly places a fitness value on filling bins.

Understood as a multi-objective problem QD algorithms can be compared based on dominance relations: if an algorithm has higher coverage and higher mean fitness we can qualitatively consider it better. If two algorithms both achieve full coverage we can compare the mean fitness to quantify how much better one is than another.

In *(Cully and Demiris, 2017)* an extensive comparison of QD variants was performed. To accomplish this comparison across disparate algorithm the authors introduced a modular framework which fit MAP-Elites, NSLC, and NS, and recombinations of the two. Three components were defined: a container, a selection process, and a score used for selection. In this framework MAP-Elites uses a *grid* container, *random* selection based on *no score*; NSLC an *archive* container, selection from *Pareto optimal* solutions based on *novelty and local quality*; and NS an *archive* container using *population-based* selection dependent on the *novelty* score. A standard EA would then be configured as an {*archive*,

population-based, *fitness*} algorithm, and unnamed variants are easily produced such as an NSLC with random selection configured as {*grid*, *random*, *no score*}.

The components were shuffled and variants tested in an effort to make generalizations and about their behavior. The authors found that both unstructured archives and grids were able to produce high quality collections of solutions, but that population-based approaches which optimized a single value failed to produce high performing and complete collections: when selecting for fitness the population converged on high performing regions, when selecting for novelty the population did not discover high performing solutions. Perhaps most surprisingly, despite the number of variants of selection operators tested, uniform random selection was found to be nearly the top performer across all variants.

As the number of solutions in an archives grows, the chance for an individual to be chosen as a parent by uniform random selection decreases. But the ever decreasing chance that the most fit individual will be selected as a parent is less problematic when the population is viewed through the lens of genes rather than of individuals. As the children of successful individuals fill more niches in the archive the chance of their genes being selected also increases. Blocks of related individuals take over regions of the map and uniform random selection begins to resemble proportionate selection — genes are selected in proportion to the number of niches they are present in. With this understanding uniform random selection does not seem as uninformed a strategy as it first appears.

The only selection operator which outperformed random selection with a grid container was the "Curiosity score" introduced in the same study *(Cully and Demiris, 2017)*. This metric favors successful parent solutions, increasing their score when a child is added to the archive and decreasing when it is not. Selection is then proportionate based on curiosity score, favoring those "evolvable" individuals which are improving the archive until they no longer produce useful children. If the map is viewed as subpopulations of related individuals the curiosity score rewards those individual's at the edges, which can enter unoccupied niches, and the most promising individuals within subpopulations, which have recently taken over occupied niches.

2.1.3 *Multidimensional Archive of Phenotypic Elites (MAP-Elites)*

The QD algorithms analyzed and developed in this dissertation are based on the MAP-Elites algorithm. In its canonical form *(Mouret and Clune, 2015)* MAP-Elites uses an evenly space grid along predefined descriptor dimensions as an archive, known as a *map*, and creates new solutions by perturbing parent's real-valued genome vectors with isotropic Gaussian noise. During the course of this work we will at

times depart from this canonical form, using alternate methods of constructing and organizing the archive (Chapter 4), creating child solutions (Chapter 6), and evaluating individuals (Chapter 5).

Despite these modifications we do not depart from the main principles of MAP-Elites:

1. a descriptor (or behavior, or feature) space is divided into a large number of niches (or bins, or cells) that are organized spatially in an archive (also called a "map");
2. each solution is characterized by two values: fitness (or performance), and a set of coordinates in descriptor space;
3. each niche contains the best known solution (the elite) for the corresponding combination of descriptors;
4. to generate a new candidate (or child) solution, elites are selected from the archive and variation operators applied.

MAP-Elites is more precisely defined in Algorithm 1. A map is defined as a set of discretized bins defined over a descriptor space. Each of these bins can contain the genotype ($\mathcal{X}$) of a solution and its corresponding fitness value ($\mathcal{F}$). Initial solutions can be produced randomly or taken from another source. New candidate solutions ($\mathbf{x}'$) are produced by selecting and varying genotypes ($\mathcal{X}$) from the map. The new genotypes are evaluated by the fitness function to get two measures: a performance measure ($\mathbf{p}$) and coordinates in descriptor space ($\mathbf{d}$). These descriptor coordinates indicator the position of the solution in the map, which is used to assign a the candidate a bin. In the bin is empty, or the candidate solution has a higher fitness than the solutions already in the bin, than the candidate solution is placed inside — joining the archive as an elite. The process of variation, evaluation, and descriptor-localized competition repeats to "illuminate" the descriptor space with ever higher performing solutions.

MAP-Elites was originally developed in the context of learning behavioral repertoires for damage recovery in robotics *(Cully et al., 2015)*. This line of damage recovery research has continued fruitfully *(Chatzilygeroudis et al., 2016; Papaspyros et al., 2016; Chatzilygeroudis et al., 2018; Pautrat et al., 2018; Kaushik et al., 2020a)*, and outside of damage recovery the creation of behavioral repertoires has found use in robot manipulators *(Cully et al., 2015; Ecarlat et al., 2015; Cully and Demiris, 2017, 2018; Cully, 2019)*, legged robots *(Cully et al., 2015; Chatzilygeroudis et al., 2018; Nordmoen et al., 2018a,b)*, robot swarms *(Engebråten et al., 2018; Cazenille et al., 2019)*, and soft bodied robots *(Mouret and Clune, 2015; Rieffel and Mouret, 2018)*.

In an early sign of the power and versatility of the underlying technique, MAP-Elites was able to create adversarial images that could fool deep learning recognition systems *(Nguyen et al., 2015a)*, and generate evocative images that earned recognition at an art competition whose

Algorithm 1 Multi-dimensional Archive of Phenotypic Elites (MAP-Elites)

function MAP-ELITES($fitness()$, $variation()$, $\mathcal{X}_{initial}$)
  $\mathcal{X} \leftarrow \emptyset$, $\mathcal{F} \leftarrow \emptyset$ ▷ *Map of genomes $\mathcal{X}$, and fitnesses $\mathcal{F}$*
  $\mathcal{X} \leftarrow \mathcal{X}_{initial}$ ▷ *Place initial solutions in map*
  $\mathcal{F} \leftarrow fitness(\mathcal{X}_{initial})$
  for iter = $1 \rightarrow I$ **do**
    $\mathbf{x}' \leftarrow variation(\mathcal{X})$ ▷ *Create new solution from elites*
    $\mathbf{p}', \mathbf{b}' \leftarrow fitness(\mathbf{x}')$ ▷ *Get performance and behavior*
    if $\mathcal{F}(\mathbf{d}') = \emptyset$ or $\mathcal{F}(\mathbf{d}') < \mathbf{p}$ **then** ▷ *Replace if better*
      $\mathcal{F}(\mathbf{d}') \leftarrow \mathbf{p}'$
      $\mathcal{X}(\mathbf{d}') \leftarrow \mathbf{x}'$
    end if
  end for
  return $(\mathcal{X}, \mathcal{F})$ ▷ *Return illuminated map*
end function

judges were unaware of their algorithmic origin *(Nguyen et al., 2015b; Nguyen et al., 2016)*.

In games the diversity of solutions produced by MAP-Elites has proven especially useful to explore and refine the performance of various competitive strategies *(Fontaine et al., 2019a,b; Justesen et al., 2019; Mesentier Silva et al., 2019)*, to develop complementary strategies in cooperative games *(Canaan et al., 2019)*, and to solve single player games with sparse and deceptive reward signals *(Ecoffet et al., 2019)*.

Beyond playing games, the potential for MAP-Elites to *create* them is only beginning to be exploited *(Khalifa et al., 2018; Alvarez et al., 2019; Gravina et al., 2019; Khalifa et al., 2019; Charity et al., 2020)*. The ability to produce variety is invaluable for creating procedurally generated content, a domain where finding a single global optima has little use. Creative systems can take advantage of the variety produced to provide a menu of options to the user, who can then guide the direction of further optimization. Human-in-the-loop optimization with MAP-Elites has been applied to both video game content generation *(Alvarez et al., 2019)*, and the conceptual phase of industrial design *(Hagg et al., 2018)*.

The range of applications MAP-Elites finds use in continues to grow even into more traditional areas of optimization from scheduling and routing *(Nguyen et al., 2018; Urquhart and Hart, 2018; Urquhart et al., 2020)* to symbolic regression *(Bruneton et al., 2019; Dolson et al., 2019)*.

In tackling this range of domains a host of algorithmic innovations and tweaks have been introduced: extending and adjusting the geometry of the archive, the methods of selecting parents, new mutation operators to improve exploration and optimization, and methods to automatically learn and define descriptors.

With a large number of descriptor dimensions the number of cells in a uniform grid structure quickly grows unwieldy. By reimagining the grid as a high dimensional Voronoi tessellation *(Vassiliades et al., 2017)*, a set number of cells can be defined to evenly cover a descriptor space of any dimensionality — a technique that has been shown to be successful with two as well as one-thousand descriptor dimensions. In low-dimensional cases, to better fit the underlying distribution of the population and reachable space, archive boundaries can be shifted during optimization *(Fontaine et al., 2019b)* or defined as a quadtree instead of grid cells *(Smith et al., 2016)*.

Beyond fitting the boundaries of the descriptor space grid, design of the descriptor itself can be automated. Typically, like the objective, the descriptor is the result of a hand-crafted function which incorporates domain knowledge to the guide the algorithm toward the desired result. But this descriptor can also be learned. In *(Cully and Demiris, 2018)* an autoencoder (see Section 2.3.1) is used to learn a mapping from a high dimensional descriptor space to a two-dimensional space that can be easily explored. This autoencoder-based approach was extended to learn the descriptor during optimization *(Cully, 2019)* by compressing the high-dimensional sequence of actions of a robot into a low-dimensional descriptor space. As the robot explores more actions the descriptor space is expanded and refined — gradually fitting the descriptor to the space of possible actions. A descriptor space can also be learned from already existing examples. In *(Justesen et al., 2019)* a descriptor space was derived by compressing thousands of human game playing demonstrations, deriving a low-dimensional descriptor which captured the diversity of human strategies.

Though easily extended to domain specific recombination operators *(Mouret and Clune, 2015; Urquhart and Hart, 2018; Dolson et al., 2019)*, mutation in the canonical MAP-Elites algorithm is applied as isometric Gaussian noise to a real-valued parameter vector. This bare-bones operator has been adjusted to gradually lower perturbation variance with annealing schedules *(Nordmoen et al., 2018b)*, adjusting the covariance of the Gaussian based on correlations of individuals within the map *(Vassiliades and Mouret, 2018; Fontaine et al., 2019a)*, or performing multiple optimization steps based on an implicit gradient *(Colas et al., 2020)*.

2.2 PREDICTIVE MODELS FOR OPTIMIZATION

2.2.1 *Surrogate-Assisted and Bayesian Optimization*

Evolutionary approaches typically require a large number of evaluations before acceptable solutions are found. Fields such as antenna design *(Hornby et al., 2006)*, astrophysics *(Parkinson et al., 2006)* and biology *(Gonzalez et al., 2015)* require sophisticated and computationally

intensive simulations to evaluate solutions. The high cost of such simulations makes the overall computational cost of repeated evaluations prohibitively expensive for traditional optimization techniques.

In these cases approximate models of the fitness function, also known as metamodels or surrogate models, can be used in their place *(Emmerich et al., 2002; Jin, 2005; Stork et al., 2018)*. These models can take the form of data-driven models built through a combination of sparse sampling of the parameter space and active learning. Surrogate-assisted optimization techniques have been particularly important in domains which require complex fluid dynamics simulations to measure performance, in particular aerodynamic design *(Hasenjäger and Sendhoff, 2005; Giannakoglou et al., 2006; Zhou et al., 2007; Dumas, 2008; Forrester and Keane, 2009; Lian et al., 2010; Gaier et al., 2017a)*.

Modern surrogate-assisted optimization often takes place within the framework of Bayesian optimization (BO) *(Brochu et al., 2010; Cully et al., 2015; Calandra et al., 2016; Shahriari et al., 2016; Gaier et al., 2017b; Pautrat et al., 2018)*. BO approaches an optimization task not as one of finding the most optimal solution, but of modeling the underlying objective function in high-performing regions.

Bayesian optimization has two components. The first is a surrogate model of the objective function. These surrogates are probabilistic data-driven models based on a set of input/output pairs, or an *observation set*. The initial set of solutions can be taken randomly or by sampling the parameter space with design of experiments techniques such as Latin hypercube sampling*(McKay et al., 1979)* or Sobol sequences *(Niederreiter, 1988)*. The second component of BO is the *acquisition function*, which describes the utility of evaluating the objective function at a given point.

BO proceeds by searching for the point with maximal utility, evaluating it on the objective function, and adding this input/output pair to the set of observations. The updated observation set is then used to produce a more informed model. The process then repeats, refining the model of the objective function with each new evaluation. This process is illustrated in Figure 2.2

2.2.2 *Gaussian Process Models*

OVERVIEW Surrogate models can be constructed using any number of data-driven machine learning techniques, including polynomial regression, support vector machines, or artificial neural networks *(Jin, 2005; Forrester and Keane, 2009)*. As BO relies on some quantification of uncertainty, and Gaussian process (GP) models *(Rasmussen and Williams, 2006)* are commonly used.

GP models are non-parametric models which are accurate even with small data sets and their predictions include a measure of certainty. In the active learning context of surrogate-assisted optimization a

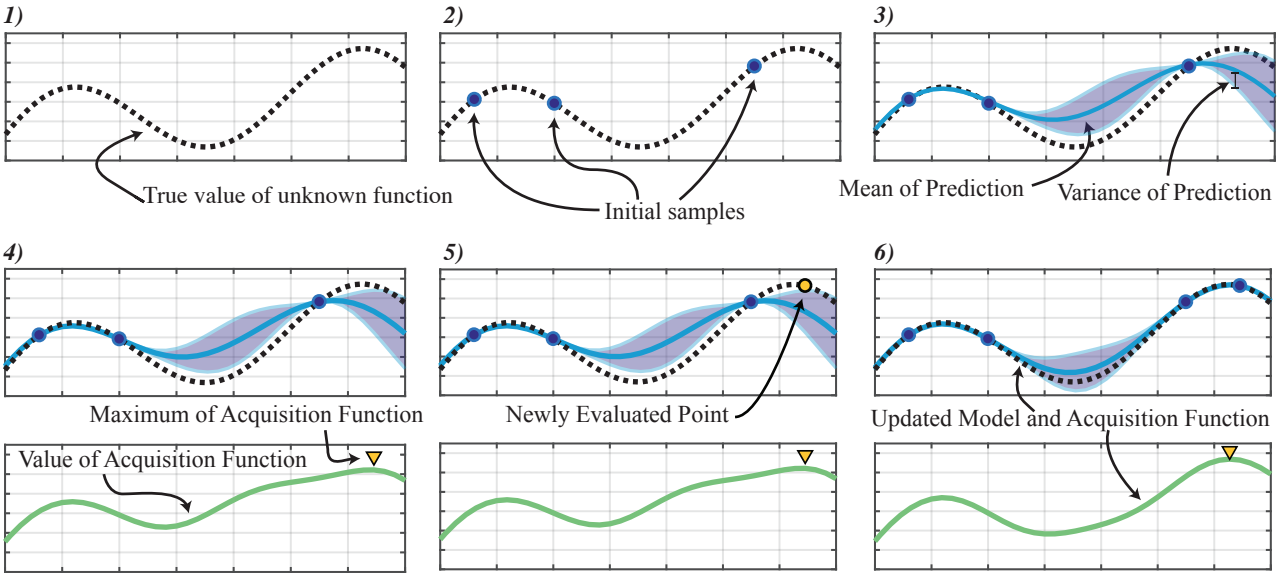

Figure 2.2: *Active Learning with Bayesian Optimization*
(1) The goal of Bayesian Optimization (BO) is to model the behavior of an unknown objective function in high performing regions. *(2)* This is done by first sampling points on the function randomly, *(3)* and then building a model of the function based on these points. A probabilistic model is used, giving us a mean prediction of the objective value at every point along with a variance, or confidence, of the prediction. *(4)* The value of an acquisition function is derived for new candidate points to sample: this function defines the utility of sampling at that point, a combination of high performance and high variance. *(5)* The point which maximizes the acquisition function is found, evaluated on the unknown objective function, and *(6)* the model and acquisition function is updated.

measure of model uncertainty is particularly useful, as this allows for the balancing of exploration and exploitation.

In much the same way as an artificial neural network can be thought of as a function that returns a scalar given an arbitrary input vector, a GP model can be thought of as a function that, returns the mean and variance of a normal distribution, with the variance indicating the certainty of the prediction. Gaussian process models use a generalization of the Gaussian distribution: where a Gaussian distribution describes a distribution of random variables, defined by mean and variance, a Gaussian process describes a random distribution of functions, defined by a mean function μ, and covariance function k.

$$f(x) \sim GP(\mu(x), k(x, x')) \tag{2.1}$$

GP models are based on assumptions of smoothness and locality: the intuition that nearby points will have similar behavior. A covariance

function k defines this relationship precisely in the form of a kernel. A common choice of kernel is the squared exponential function:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp\Big(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\Big) \tag{2.2}$$

The form of the squared exponential kernel limits the space of expressible models. It makes the assumption that the approximated function is smooth, that two points which are nearby in parameter space will have highly correlated function values, and that two distant points will have little correlation. Other kernels, such as sinusoidal or Matérn kernels, can encode different model assumptions and so alter the variety of models which can be expressed.

The relationship between points can be defined as a covariance. Given a set of observations $D = (x_{1:t}, f_{1:t})$ where $f_{1:t} = f(x_{1:t})$, we can build a matrix of these covariances. In the simple noise-free case we can then construct the covariance, or kernel, matrix K:

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \cdots & k(x_t, x_t) \end{bmatrix} \tag{2.3}$$

For a new point (x_{t+1}) we can derive the value $(f_{t+1} = f(x_{t+1}))$ from the normal distribution:

$$P(f_{t+1}|D_{1:t}, x_{t+1}) = \mathcal{N}\Big(\mu_t(x_{t+1}), \sigma_t^2(x_{t+1})\Big) \tag{2.4}$$

$$\mu_t(x_{t+1}) = \mathbf{k}^T\mathbf{K}^{-1}\mathbf{f}_{1:t} \tag{2.5}$$

$$\sigma_t^2(x_{t+1}) = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k} \tag{2.6}$$

giving us the predicted mean and variance for a normal distribution at the new point x_{t+1}. When the objective function is evaluated at this point, we add it to our set of observations D, reducing the variance x_{t+1} and at other points near x_{t+1}.

KERNEL FUNCTIONS The predictions of the GP model are based on the kernel matrix, and so the entire nature of the model is characterized by the kernel function used to construct it. The kernel function defines the influence of one data point on the estimation of others, and takes the form of a positive semi-definite distance function. Though many kernels have been proposed *(Rasmussen and Williams, 2006; Brochu et al., 2010; Snoek et al., 2012)*, the most commonly used is the squared exponential, a "bell curve" shaped function which exerts maximum influence when there is no distance to another point, and whose influence decays exponentially as the distance increases. The rate of this decay is parameterized by a *characteristic length scale*, ℓ which

defines the distance at which the influence of a point is near zero, giving the parameterized kernel:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \eta \exp\Big(-\frac{1}{2\ell}\|\mathbf{x}_i - \mathbf{x}_j\|^2 \Big) \tag{2.7}$$

where ℓ is the characteristic length scale and η the variance of the model, or how far the output can vary from the function's mean.

When a single length scale is used the kernel is isotropic, with the influence exerted by a point equally in all dimensions. In many cases points are more correlated in one dimension than another, in this case an anisotropic kernel can be formulated. A different length scale can be used for each dimension, with each optimized when fitting the GP in a process known as automatic relevance detection (ARD) *(Rasmussen and Williams, 2006)*.

HYPERPARAMETER FITTING Fitting hyperparameters of the kernel function, such as the length scale, is done by maximizing the marginal log likelihood of the model given the data. Given the function values $\mathbf{y}$, the individuals in the population $\mathbf{x}$, and compatibility kernel matrix $\boldsymbol{K}$, this is calculated as:

$$\log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T \left(\boldsymbol{K} - \sigma_n \boldsymbol{I}\right)^{-1} \boldsymbol{y} - \frac{1}{2}\log\left|\left(\boldsymbol{K} - \sigma_n \boldsymbol{I}\right)\right| - \frac{n}{2}\log 2\pi \tag{2.8}$$

where the first term penalizes models which do not fit the data, the second term penalizes the model's complexity (preferring simpler models), and the third term a normalization constant.

2.2.3 *Acquisition Functions*

As the active learning process of BO is dependent on selecting points with the highest utility, how "utility" is defined can be critical to the algorithm's performance. Balance must be maintained between exploration, evaluating points with high uncertainty, and exploitation, evaluating points which are likely to have high fitness.

Choosing new points to evaluate based solely on predicted fitness is too greedy, and will result in premature convergence on local optima. At the other extreme, evaluating points with the least confidence will decrease the uncertainty of our models globally, but it is also wasteful: with only limited resources improving accuracy around optima should be prioritized, not the search space as a whole. New points for evaluation should be chosen where the model predicts both high fitness and where model uncertainty is high. How this balance is struck is defined the choice of the acquisition function.

PROBABILITY OF IMPROVEMENT (PI) As BO is typically employed to find a global optima, a reasonable approach is to select points based on the probability that they are better than the current best point, or their probability of improvement (PI) *(Kushner, 1964)*. A point predicted to have a large performance improvement over the current optima with little uncertainty the PI is high; if a point is predicted to have a small performance improvement or has high uncertainty the PI will be low.

In its purest form PI is a purely exploitative strategy, with even a small improvement with low variance favored over a large improvement with large variance. In practice a trade-off parameter is often introduced, allowing the designer to control the weighting of exploration and exploitation. Probability of improvement is calculated as:

$$\begin{aligned} \mathrm{PI}(\mathbf{x}) &= P\left(f(\mathbf{x}) \geq f\left(\mathbf{x}^{+}\right) + \xi\right) \\ &= \Phi\left(\frac{\mu(\mathbf{x}) - f\left(\mathbf{x}^{+}\right) - \xi}{\sigma(\mathbf{x})}\right) \end{aligned} \tag{2.9}$$

where Φ is the normal cumulative distribution function, which takes into account the difference between the predicted value of the new point $\mu(\mathbf{x})$ and the current optima $f(\mathbf{x}^{+})$, the variance at that point $\sigma(\mathbf{x})$, and ξ controlling the explore/exploit trade off by varying the importance of the difference versus the variance.

EXPECTED IMPROVEMENT (EI) PI only takes into account whether or not a new point is better than the global optima, favoring a very certain small improvement over a less certain large improvement. Expected improvement (EI) *(Mockus et al., 1978)* was introduced to consider the magnitude of improvement, not only the probability. As in PI, an additional parameter is introduced to control the relative influence of uncertainty, give the EI calculation *(Brochu et al., 2010)*:

$$\begin{aligned} \mathrm{EI}(\mathrm{x}) &= \begin{cases} \left(\mu(\mathrm{x}) - f\left(\mathrm{x}^{+}\right) - \xi\right)\Phi(Z) + \sigma(\mathrm{x})\phi(Z) & \text{if } \sigma(\mathrm{x}) > 0 \\ 0 & \text{if } \sigma(\mathrm{x}) = 0 \end{cases} \\ &\text{where} \quad Z = \frac{\mu(\mathrm{x}) - f\left(\mathrm{x}^{+}\right) - \xi}{\sigma(\mathrm{x})} \end{aligned} \tag{2.10}$$

where ϕ is the probability density function and Φ is the cumulative distribution function of a normal distribution.

UPPER CONFIDENCE BOUND (UCB) The *upper confidence bound* (UCB) *(Srinivas et al., 2010)* is an intuitive and straightforward acquisition function. UCB judges new points optimistically, favoring uncertainty under the assumption that higher uncertainty hides a

potentially higher reward. UCB can be defined as a weighted sum of the mean (μ) and uncertainty (σ) of the prediction, where both a high mean and large uncertainty are favored, and their relative emphasis tuned by the parameter κ:

$$UCB(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}) \tag{2.11}$$

Proposed as part of the GP-UCB algorithm, UCB minimizes regret and maximizes information gain in multi-armed bandit problems, the prototypical exploration/exploitation balancing problem *(Srinivas et al., 2010)*, and performs competitively with EI and PI *(Brochu et al., 2010; Calandra et al., 2016)*.

Notably, the UCB calculation is based only on the underlying model — not the current optima. The work presented in this dissertation is concerned with finding diverse solutions not single optima, making UCB the most appropriate choice in our predictive modeling problems. Improvement-based acquisition functions are simply not suitable for finding multiple optima — once a global optima is found the expected improvement of all other solutions will be zero.

2.3 GENERATIVE MODELS FOR REPRESENTATION

2.3.1 *Autoencoders and Dimensionality Reduction*

Dimensionality reduction is the process of reducing the number of features used to describe data. This reduction can be accomplished by extracting or constructing a set of new, and fewer, features based on the original features of the data. Dimensionality reduction takes place within a framework composed of two components: an "encoder" which produces new features from the original features, and a paired "decoder" which reproduces the old features from those new features. Put another way, the encoder compresses the data from the initial space into a "latent" or encoded space. The decoder then decompresses the the features encoded in the latent space back to the initial space. These different spaces are typically defined by a vector of values: encoding maps a longer vector to a shorter one, and decoding maps that shorter vector back to a vector of the original length.

The process of dimensionality reduction is typically a "lossy" one — information is lost during the encoding process which cannot be recovered by the decoder. The broad goal of dimensionality reduction is to find the encoder/decoder pair that retains the most information when encoding the data and introduces the least amount of error when decoding the latent representation. The difference between the original data and the data produced after encoding and decoding is the *reconstruction error*.

The most common method of dimensionality reduction is principal component analysis (PCA) *(Wold et al., 1987)*. PCA constructs new

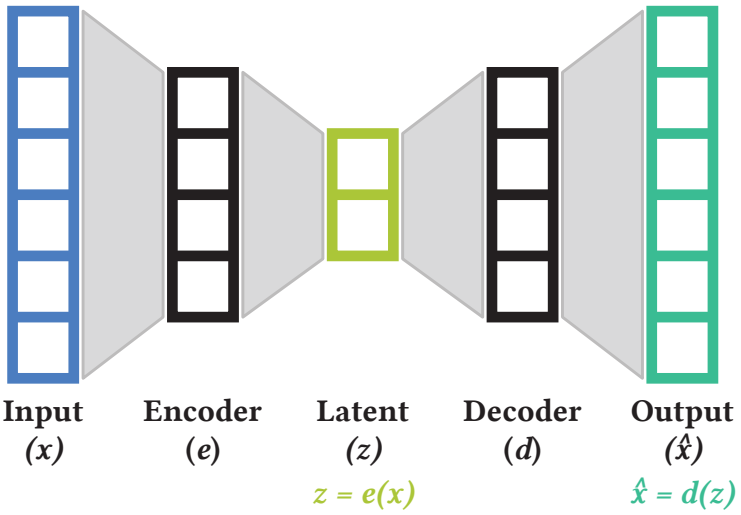

Figure 2.3: *Architecture of an autoencoder*
An autoencoder is composed of two neural networks, an encoder network (e) and a decoder network (d). Input data (x) is mapped by the encoder network into a latent vector (z). This latent vector is in turn mapped by the decoder to a an output vector ($\hat{x}$). Both networks are trained together to minimize the difference between the input and the output ($\|x - \hat{x}\|^2$).

features from linear combinations of the original features. The goal of PCA is to describe the original data set by a combination of new orthogonal features, or principal components. These features are iteratively constructed, such that each principal component describes the largest amount of variance in the data, until a given number of principal components have been used, or a satisfactory amount of variance is explained (e.g. 95%).

As PCA is a linear conversion of the data from one coordinate system to another the encoder $\mathcal{E}$ can be expressed as an $[N \times M]$ matrix, where N is the number of principal components, or latent variables, and M is the number of features is in the original data. The decoder $\mathcal{D}$ is then simply the transpose of this encoder matrix.

$$z = x * \mathcal{E} \tag{2.12}$$

$$\hat{x} = z * \mathcal{D} \tag{2.13}$$

where x is the original data, z the principal components or latent vector, and $\hat{x}$ the reconstructed data.

Autoencoders (AEs) *(Hinton and Salakhutdinov, 2006)* perform dimensionality reduction using two neural networks, one for the encoder and one for the decoder. Data is fed to the encoder network to produce a lower dimensional output, and this output is in turn fed to the decoder network which produces data of the same dimensionality as the input. The combined networks from a single architecture (see

Figure 2.3) which is trained end-to-end to minimize the reconstruction error, defined as:

$$loss = \|x - \hat{x}\|^2 \tag{2.14}$$

Though in the ideal case the data is perfectly reconstructed, in practice the informational "bottleneck" of a lower dimensional latent space is meant to allow only the most structured and common aspects of the data to be reconstructed.

Just as the linear transformations performed by PCA can be understood as a series of matrix multiplications, so too can the processing of data by neural networks. In the case where both the encoder and decoder networks have only one layer and linear activation functions the form of the dimensionality reduction is the same as PCA: an $N \times M$ encoder matrix and a $M \times N$ decoder matrix.

PCA finds the projection with the smallest information loss, but the matrices found by PCA are just one possible solution — many encoder/decoder pairs can exist which have optimal reconstruction errors. In AEs there is no requirement that features are orthogonal or independent, and the networks used typically have multiple layers and non-linearities, allowing far more possible mappings than can be found by PCA. It is even possible that a sufficiently complex AE architecture could encode the data perfectly — even with only a single latent dimension. Such an AE could essentially memorize the data, encoding each data point as a value and then decoding it perfectly.

Memorization of the data in this way satisfies the loss function perfectly, but sidesteps our actual goals. By compressing the data with an autoencoder we hope to (1) find structures in the latent space which we can interpret and exploit, and to (2) find a a reduced representation that captures the structure of the data. In the case of an AE which memorized the data the latent space would be so unstructured as to be useless, it is essentially a look-up table. And though the case of memorization is an extreme case, it is illustrative of the underlying problem: without explicitly searching for a well structured latent space, we cannot expect one.

2.3.2 *Variational Autoencoders (VAEs)*

In the ideal case an AE would produce a latent space which is a smooth, compressed representation of our data. To produce new data from the same distribution we could simply take a random point in the latent space, and new solutions could be evolved by moving in latent space. In practice, we cannot assume that the latent space produced by our AE is well organized: when the AE overfits the input data other points may be meaningless, and points nearby in latent space may be decoded to very different points in the original space, making search

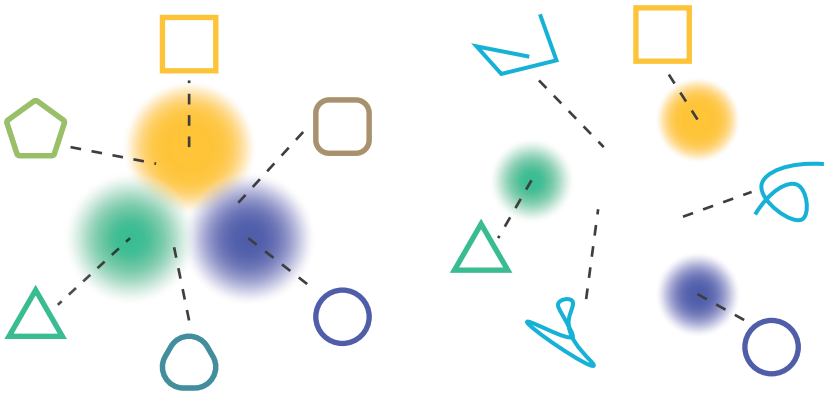

Figure 2.4: *Idealized vs Typical Autoencoder Latent Space.*
Left: The ideal latent space is compact and local: a randomly sampled point will produce a new solution similar to the training data, and a point sampled between two known data points will be an interpolation of the two.
Right: In reality the latent space produced by an autoencoder is often non local, newly sampled points are not only not interpolations between existing data points, but may not be valid or realistic data points at all.

within the latent space infeasible. The difference between our ideal latent space and the typical AE latent space is illustrated in Figure 2.4.

These problems arise because we train the AE for reconstruction error, not a well-structured latent space. Variational autoencoders (VAEs) *(Kingma and Welling, 2014)* are explicitly regularized to induce a well-structured latent space suitable for use as a generative model and representation of the data.

VAEs encode the input as distributions rather than points in latent space. Structure in the latent space is then induced by constraining these distributions to be close to Gaussian. When training a VAE the input is mapped to a normal distribution in latent space, defined by a vector of means and variances. A point from this distribution is sampled, decoded, and the resulting reconstruction error is used to train the network.

Encoding the input as a normal distribution has a regularizing effect, but this regularization can also circumvented: a distribution with zero variance is also a distribution, even if it really is a point. To prevent the collapse of these distributions, reverting our architecture to a standard AE, an additional regularization term is introduced into the loss function.

The loss function of a VAE is composed of a reconstruction term, which is the same as with the AE, and a regularization term:

$$loss = \|x - \hat{x}\|^2 + KL\left[\mathcal{N}\left(\mu_x, \sigma\right), \mathcal{N}(0,1)\right] \tag{2.15}$$

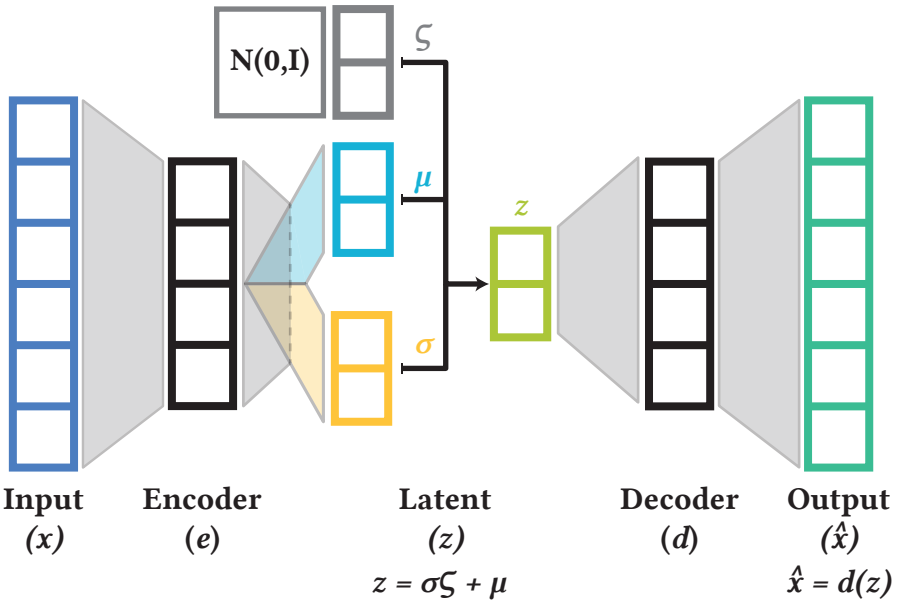

Figure 2.5: *Architecture of a Variational Autoencoder*
The encoder network of a variational autoencoders encodes the input as a multivariate Gaussian distribution defined by a vector of variances and means. When training the VAE a point from this multivariate distribution is sampled randomly. The decoder is unchanged from that of the classic autoencoder, mapping from a latent vector back to an output. The weights of the networks which determine the mean, variance, and decoder are trained to minimize the difference between the input and output, and to minimize the difference of the encoded distributions to a unit normal distribution as defined by the KL divergence.

The regularization term is the KL divergence *(Kullback and Leibler, 1951)* of the latent distribution to a normal distribution. Forcing the data to fit a zero mean normal distribution induces the data points to be encoded near each other, and combined with unit variance, encourages the distributions overlap. A regularized latent space creates smooth transitions between data points, allowing for a continuous latent space that can be reliably sampled and easily searched. This brings us closer to the idealized latent space from Figure 2.4.

To encode and distributions rather than points requires modification of the encoder component of the autoencoder architecture (Figure 2.5). Rather than encoding input data directly to a latent vector of real-valued numbers, it is encoded to a multivariate Gaussian distribution defined by a vector of means and variances. The encoder defines the parameters of the distribution which a latent vector is sampled from. The decoder remains unchanged: it still takes a point in latent space as input and decodes it.

The latent vector that is fed to the decoder is obtained by sampling the encoded distribution. As we cannot backpropagate error through random sampling, the "reparameterization trick" is used —

the deterministic and probabilistic components of the sampling are separated. This allows backpropagation of error through the deterministic components, the mean and variance, while still providing random values to the decoder. The latent vector z passed to the decoder is the combination of these three components:

$$z = \mu + \sigma * \zeta \tag{2.16}$$

where μ is a vector means, σ variances, and ζ drawn from $\mathcal{N}(0, I)$. By encoding the points as distributions, and training the network to match these distributions as normal, we create a regularized latent space that is suitable for use as a representation.

2.3.3 *Generative Adversarial Networks (GANs)*

Generative Adversarial Networks (GAN) *(Goodfellow et al., 2014)* are an alternate approach to generative modeling. Rather than performing dimensionality reduction, GANs pose the training of a generative model as a game between two networks, a generator network and a discriminator network. The generator network tries to create fake data; the discriminator tries to detect fake data. Competition between these networks drives improvement until the fake data is indistinguishable from the real data.

Each network takes turns training. The discriminator network is fed data from two sources: real data taken from a training set and fake data created by the generator network. The network is then trained to distinguish the real from the generated instances. The generator network uses feedback from the discriminator network to learn to create fake data. The generator network takes random noise as input, and outputs a fake data instance. This instance is given to the discriminator for classification, and the loss calculated from the accuracy of the classification.

The original loss function proposed for GANs *(Goodfellow et al., 2014)* is Minimax loss, derived from the cross-entropy between the real and generated distributions:

$$\overbrace{E_x[\log(D(x))]}^{\text{expected value over real data}} + \overbrace{E_z[\log(1 - D(G(z)))]}^{\text{expected value over generated data}} \tag{2.17}$$

where the first term E_x is the expected value over all real data instances, with $D(x)$ the discriminator's estimate of the probability that an instance of real data is real; and the second term E_z the expected value over all generated instances, where $G(z)$ is the generator's output when given noise z and $D(G(z))$ is the discriminator's estimate of the probability that a generated instance is real. The generator's goal is to minimize this loss function, while the discriminator's goal is to maximize it.

Modern GANs typically use two loss functions, instead of one, based on Wasserstein loss *(Arjovsky et al., 2017; Gulrajani et al., 2017)* based on the "earth movers distance" *(Rubner et al., 1998)*, a measure of distance in the space of probability distributions. Discriminators in Wasserstein GANs do not directly classify instances, and rather than outputting a probability between 0 and 1 can input any value, with higher values indicating real instances. In this case the discriminator (or critic) loss is simply:

$$D(x) - D(G(z)) \tag{2.18}$$

and the generator loss is only:

$$D(G(z)) \tag{2.19}$$

where $D(x)$ is the critics output for real instances, $G(z)$ the generators output when given noise z, and $D(G(z))$ the critics output for a fake instance.

Besides success in contexts such as photo-realistic image generation *(Karras et al., 2019a)*, GANs have been used to learn representations which can then be optimized. "Latent variable evolution" approaches, which optimized input of GAN generator network with evolutionary algorithms have been shown to be able to produce fingerprints usable for adversarial attacks *(Bontrager et al., 2018b)*, robot controllers *(Jegorova et al., 2019)*, faces with adjustable features *(Zaltron et al., 2019)*, and procedural content generation *(Volz et al., 2018; Schrum et al., 2020a,b)*.

2.3.4 *GANs vs. VAEs*

Both VAEs and GANs are able to learn a data distribution and generate new data from this distribution. The samples generated by state-of-the-art GANs are typically more "realistic" matches than those produced by VAEs, often described as "blurry" — less similar to any true instance. The higher fidelity of GANs comes at the cost of a fickle and unstable training process. As GANs are based on an adversarial procedure they do not converge, only reach an equilibrium, and equilibrium that is vulnerable to mode collapse. In order to best fool the discriminator network the generative network may specialize in producing a only a small subset, or modes, of the data distribution. In contrast the training of VAEs is stable, and the loss function designed to ensure that all modes of the distribution are covered.

In this manuscript we train generative models on evolved datasets, *as they evolve*. These models will be trained in small bursts as part of the evolution loop, not once to produce a final product. Integration of model training into the algorithm puts a premium on the efficiency

and stability of training. These generative models will be used as an encoding for optimization, and coverage of all modes are critical to the expressivity of the encoding. For these reasons we focus on VAEs in this work, though in theory any properly configured generative model could serve equally well.

3

SURROGATE-ASSISTED NEUROEVOLUTION (SA-NEAT)

Some figures, text, and ideas in this section are based on the following works which have either been previously published or are in review:

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2018). "Data-efficient neuroevolution with kernel-based surrogate models." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 85–92.

3.1 INTRODUCTION

Neuroevolution (NE), the optimization of neural networks through evolutionary algorithms, has proven to be effective in both machine learning *(Whiteson and Stone, 2006; Aaltonen et al., 2009)* and evolutionary robotics *(Stanley and Miikkulainen, 2002; Doncieux et al., 2015a)*, and its flexibility has made it a standard approach for experiments in embodied cognition *(Pfeifer and Bongard, 2006)* and open-ended evolution *(Lehman and Stanley, 2008, 2011a)*. Recent work has shown that even in deep neural networks, where millions of weights must be optimized, evolutionary techniques are a competitive alternative to gradient descent, demonstrating a surprising ability to scale *(Salimans et al., 2017; Zhang et al., 2017)*. With a body of existing NE research, on topics such as exploration and overcoming deception, already being leveraged on deep learning problems *(Conti et al., 2017; Such et al., 2017)*, NE is poised for a surge in interest.

The main challenge for deploying NE techniques in many applications is that they require many fitness evaluations — too many for most realistic use cases. Deep neural networks, for instance, require long times to train even when large computational resources are brought to bear; in the case of robotics, there is a limit to the amount of interaction that is possible with the physical system.

A common approach to optimization in computationally expensive domains is to use approximate models of the objective function, or surrogate models *(Jin, 2005; Forrester and Keane, 2009; Brochu et al., 2010; Cully et al., 2015)*. These models are created through an active learning process that aims at selecting points that are both promising in term of fitness and the likelihood to improve the predictions of the surrogate model. The typical loop alternates between selecting the best point to evaluate on the target system and retraining the model to take the

knowledge gained from this new point into account. Machine learning techniques are then used to construct surrogate models which map the genotype space to predicted fitness values *(Jin, 2005; Forrester and Keane, 2009)* .

Creating such a mapping is challenging when evolving neural networks, however: In cases where the topology and weights are both evolved, the dimensionality of the input space is not constant, and the dimensions themselves carry different meanings. Put differently, the surrogate model must be able to accept neural networks of varied layouts as an input, rather than a list of parameters.

Our first insight is that kernel-based methods, such as Gaussian process (GP) models and support vector machines, do not require that the inputs all have the same dimensions: they only require that some distance function between samples is defined. Distance measures designed for graphs, such as graph edit distance *(Sanfeliu and Fu, 1983)* could theoretically be used to compute the distance between neural networks, but in practice are far too slow, with complexity exponential in the number of vertices. Though approximate measures of graph edit distance have been developed *(Neuhaus et al., 2006; Riesen and Bunke, 2009)*, even these are too slow for use as part of every prediction in an evolutionary optimization algorithm.

Our second insight is that, when evolution is used to produce neural networks, we can glean additional information into the similarity of networks through their heredity. The leveraging of hereditary information to improve the optimization process is already done in the Neuroevolution of Augmenting Topologies (NEAT) algorithm *(Stanley and Miikkulainen, 2002)*, one of the most successful neuroevolution approaches. By tracking genes as they arise in the population, it is possible to create a meaningful and computationally efficient distance measure. NEAT uses this distance to cluster similar networks into species, *here we propose its use as part of a kernel for Gaussian process regression*.

In summary, the primary idea explored in this work is to trace the common genes of networks as they evolve, and use this information to create a distance measure which can be used in a kernel-based surrogate model. Surrogate-assistance techniques can then be used to create a data-efficient neuroevolution algorithm.

Broadly, the surrogate-assisted neuroevolution algorithm presented here proceeds as follows (Figure 6.1): (1) a set of minimal networks are evaluated and form the initial training set and population, (2) the distance between all individuals in the training set is computed with a compatibility distance kernel, and a GP model constructed, (3) the population is divided into species and evolved with NEAT, with the fitness of individuals approximated by the compatibility distance model, (4) individuals in each species are evaluated and added to the training set, and the process repeats from (2).

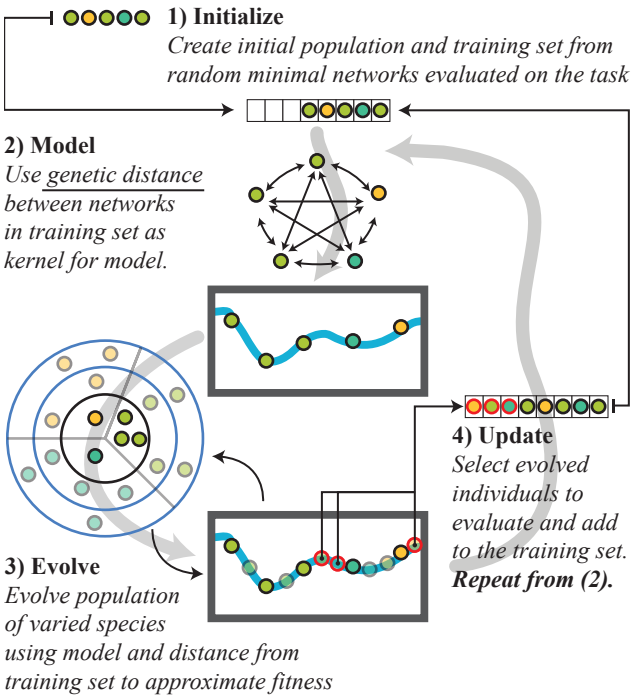

Figure 3.1: *Surrogate-Assisted NEAT Overview*
Surrogate-Assisted NEAT integrates the compatibility distance, used by NEAT for speciation, into a kernel for Gaussian process regression. This compatibility distance kernel allows for fitness approximation of networks whose topologies vary and grow more complex. Integrating surrogate-assistance techniques into NEAT reduce the number of samples to reach the same performance by several times.

3.2 BACKGROUND

Since its introduction in 2002 *(Stanley and Miikkulainen, 2002)* NEAT has become the standard for neuroevolution. The core promise of NEAT is an algorithm that simultaneously optimizes the topology and weights of a neural network, resulting in a high performing network whose topology is well-suited to the task. This is done by means of structural mutations, allowing the addition of nodes and connections during the evolutionary process. The hope is that by gradually introducing additional degrees of freedom, the networks produced by NEAT will expand to the appropriate level of complexity for the task — and in complex tasks allow for evolution to be open-ended.

NEAT has seen successes in domains from video game AI *(Stanley et al., 2005)* to optimal control *(Gaier and Asteroth, 2014b)*, to particle physics *(Aaltonen et al., 2009)*, and forms the basis and inspiration for a host of other innovations. It is the underlying algorithm for the evolution of compositional pattern producing networks *(Stanley, 2007)* which were in turn applied to the indirect encoding of large scale networks with the HyperNEAT algorithm *(Stanley et al., 2009)*.

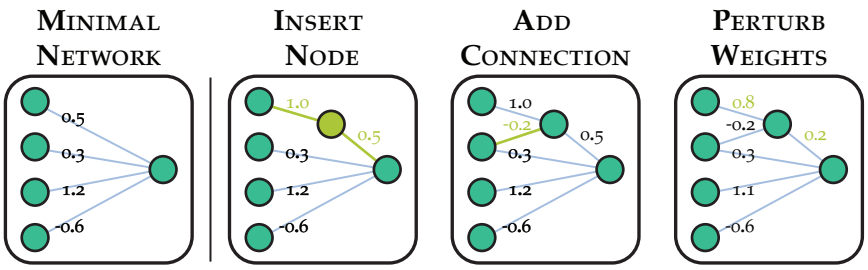

Figure 3.2: *Mutation Operators in NEAT*
NEAT begins with a population of minimal networks: inputs connected directly to outputs with no hidden nodes and random weight values. Mutation operators optimize the weights and topology of the networks simultaneously: hidden nodes can be inserted, connections added, and weights values perturbed.

The ability of NEAT to produce networks of increasing complexity has made it an ideal tool for exploring open-ended evolution and novelty-based search *(Lehman and Stanley, 2008, 2011a)*.

EVOLVING COMPLEXITY NEAT's core capability of network complexification is achieved with a few simple operators. When mutating a network in addition to perturbing the weights of the network's connections, nodes or connections may also be added (Figure 3.2).

At first glance these operators seem sufficient to evolve networks of increasing complexity — but the emergence of more complex structures runs counter to the underlying logic of evolutionary optimization. The self-reinforcing dynamic of high performing individuals having more offspring focuses search on the highest fitness areas of genotype space. In a population of highly optimized networks, any network in which new unoptimized structures arise is likely to have worse fitness — and so unlikely to be selected to pass on their genes to the next generation. If not only a single new node or connection but several are required to achieve higher fitness the chance drops dramatically. The primary hurdle to achieve complexification then is not how to add new structures, but how to preserve more complex individuals in the face of selection pressure.

The first measure to nurture new structures is to add them in a fitness neutral way, moving in the genotype space without moving in the phenotype space *(Barnett, 2001; Ebner et al., 2001)*. Adding additional degrees of freedom without altering the fitness of the individual allows the structure to be slowly optimized and spread throughout the gene pool. This maintains the desirable property of small changes in the genotype resulting in small changes in the phenotype. To add a node in a minimally disruptive way, an existing connection is 'split': the existing connection is disabled and a new hidden node is created in its place (Figure 3.2, left). This new node is linked with two

new connections to the source and destination nodes of the replaced connection. The incoming connection is assigned a weight of 1 and the outgoing connection the same value as the original connection. In the idealized case of a linear activation function this would result in no change to the function of the network. In practice sigmoidal activations are typically used so the networks function does change, but the effect of this change is minimized.

SPECIATION Introducing structural mutations in a fitness neutral fashion only protects new structures as they arise — for them to reach their full potential they must be protected as they are tuned. An explicit diversity maintenance strategy based on the *fitness sharing* technique is used *(Goldberg and Richardson, 1987)*. The basic principle of fitness sharing is to reduce the fitness of individuals who share the same region of genotype space. This prevents crowding of solutions around a single optima and encourages exploration of the entire fitness landscape.

Though the inspiration and original formulation of the diversity maintenance measures used in NEAT come from these fitness sharing procedures, in practice it is easier to understand as a partitioning of the population into multiple subpopulations, or species, which compete against each other as a group. Species are judged by the average fitness of their members, and awarded a proportional share of offspring in the next population. So if the average fitness of the three species in the population are 2, 3, and 5, they will be awarded $\frac{2}{10}$, $\frac{3}{10}$, and $\frac{5}{10}$ of the children in the next population. Critically, the number of individuals in the each species is not considered in this calculation.

Species level competition for offspring can be thought of as a way of allocating resources to optimization within varied regions of the search space. Each species acts as its own independent population, with selection, mutation, and crossover taking place within the species to produce the number of offspring allocated by the species level competition. In this way search for optimal structure and weights takes place across the genotype space in parallel, prevents large species from dominating the population, protects newly developed species, and concentrates new child solutions in the most promising species.

The mechanisms of fitness sharing and speciation require the partitioning of the population into clusters. When the genotype space is straightforward, such as a vector of real valued numbers, calculating the similarity of different individuals is trivial. In the case of graphs which grow in complexity we do not have a static or consistent genotype space, complicating the determination of similarity.

To address the need to compare dissimilar graphs NEAT assigns *innovation markers* to every new connection gene. Innovation markers are simply a running counter that uniquely identifies each gene as it arises. By comparing these markers similar structures in dissim-

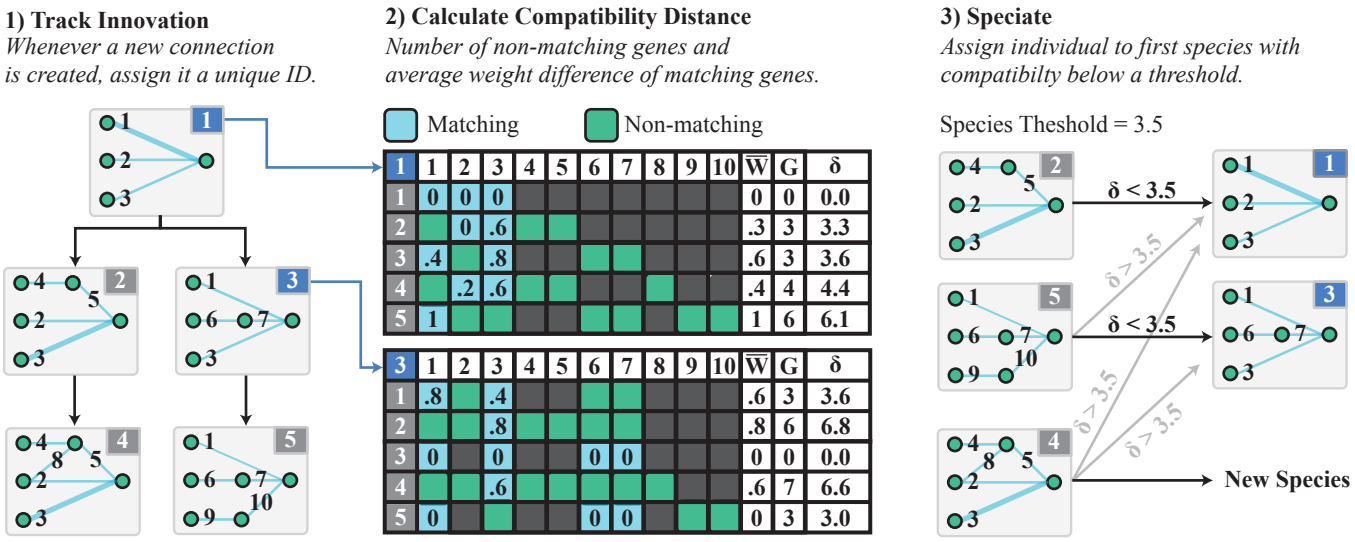

Figure 3.3: *Innovation, Compatibility Distance, and Speciation in NEAT*
By tracking genes as they arise NEAT allows for the efficient comparison of dissimilar networks. The "compatibility distance" between two networks is a weighted sum of the number of genes they do not share and the weight differences between the genes they do. Each species has a representative. Individuals are compared to these representatives, and are assigned to the first species whose representatives compatibility distance is below a threshold. If none exist the individual forms a new species.

ilar genotypes can be easily and efficiently identified, allowing the *compatibility distance* between two individuals to be calculated.

The canonical NEAT *(Stanley and Miikkulainen, 2002)* introduces several coefficients and normalization factors which provide additional degrees of freedom in how exactly this value is calculated, but it can be simplified to:

$$\delta(\mathbf{x_i}, \mathbf{x_j}) = c_1 \cdot G(\mathbf{x_i}, \mathbf{x_j}) + c_2 \cdot \overline{W}(\mathbf{x_i}, \mathbf{x_j}) \tag{3.1}$$

where the compatibility distance δ between two individuals x_i and x_j is the weighted sum of the number of non-matching genes G and the average weight differences of matching genes W.

The compatibility distance is used by NEAT to cluster individuals into species. Newly created individuals are compared to representatives of each species found in the previous generation, and join the first whose distance is below a certain threshold. The entire process of complexification, compatibility calculation, and speciation is illustrated in Figure 3.3.

OVERVIEW The NEAT algorithm proceeds as follows: (1) An initial population of minimal networks is created. All inputs of these networks are connected to all outputs with randomly weighted connections, with no hidden nodes; (2) All individuals are evaluated on

the task and awarded a fitness value; (3) individuals are compared in turn to representative of each of the species from the previous generation and assigned to the first species whose *compatibility distance* falls below a threshold — creating a new species if none is found; (4) individuals within each species compete and are recombined to produce a number of offspring awarded proportional to the mean fitness of their members. The algorithm then repeats from (2).

3.3 APPROACH

3.3.1 *Compatibility Distance Kernel*

We would like to use GP models to approximate the fitness function and generate a measure of utility for a surrogate-assisted neuroevolution algorithm. Though the estimates of GPs are typically based on distance between samples in parameter space, they are a kernel-based method, and as such require only *some* distance measure between samples.

In the case of neuroevolution, where the topology of the network is evolved along with the values of the weights, we do not have a static or consistent parameter space. As the population of networks grow and change, the parameter spaces they are define in diverge into varied dimensions with inconsistent meanings, leaving standard distance measures such as Euclidean distance unusable.

If a meaningful distance measure was found, then GP models could be used. Neural networks are a class of directed graph, and there already exist measures to compare graphs, such as graph edit distance *(Sanfeliu and Fu, 1983)*. Unfortunately, even approximate graph edit distances are too expensive to compute for every prediction *(Neuhaus et al., 2006; Riesen and Bunke, 2009)*.

This problem of measuring the similarity between the dissimilar networks was already solved by NEAT by the system of comparing innovation numbers to compute a *compatibility distance* between individuals (see Section 3.2). This compatibility distance is used by NEAT to cluster similar individuals into species, and we can use it as a kernel for our GP, allowing us to perform predictions across dissimilar structures.

To produce the kernel matrix of the GP we use a compatibility distance kernel function which returns the squared exponential compatibility distance between individuals:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{1}{2}\delta(\mathbf{x}_i, \mathbf{x}_j)^2\right) \tag{3.2}$$

where δ is the compatibility distance between individuals x_i and x_j.

The compatibility distance calculation (equation 3.2) only compares average weight distances and common connections. The precision of such predictions may be limited — but the underlying assumption,

that the more similar two individuals are the more similarly they can be expected to perform, holds. The rough predictions produced by the predictive model provide enough information to guide the search to select higher fitness individuals to produce more offspring.

When training a GP model, its hyperparameters are tuned to make the known observations most likely given the model, balancing accuracy and simplicity. We tune two hyperparameters of our kernel: the characteristic length scale (ℓ), which can be thought of as the smoothness of the model and the variance (η), how far the output signal varies from the function's mean. Integrating these hyperparameters give us a kernel of the form:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \eta \exp\Big(-\frac{1}{2\ell}\delta(\mathbf{x}_i, \mathbf{x}_j)^2 \Big) \tag{3.3}$$

These hyperparameters θ are optimized by maximizing the log likelihood of the fitness values $\mathbf{y}$ given the individuals in the population $\mathbf{x}$ and compatibility kernel matrix $\boldsymbol{K}$:

$$\log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T \left(\boldsymbol{K} - \sigma_n \boldsymbol{I}\right)^{-1} \boldsymbol{y} - \frac{1}{2}\log\left|\left(\boldsymbol{K} - \sigma_n \boldsymbol{I}\right)\right| - \frac{n}{2}\log 2\pi \tag{3.4}$$

Typically, gradient-based optimization is used to maximize the likelihood, but this is not possible here because the compatibility distance is not differentiable. Instead, we use the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which has been proven as effective at optimizing Gaussian process model parameters as gradient-based methods in other contexts *(Chatzilygeroudis et al., 2017)*. In addition to the kernel specific parameters ℓ and η the mean (μ) and signal noise (σ) are also tuned.

The training and prediction process can be summarized as follows: (1) All individuals in the training set are compared using NEAT's compatibility distance metric to produce a covariance matrix of their similarity; (2) the hyperparameters of the model are optimized with CMA-ES to maximize the likelihood of the data given the model; and (3) prediction can be calculated based on the model and distance to the individuals in the training set. This training and prediction process is illustrated in Figure 3.4.

3.3.2 *Surrogate-Assisted NEAT*

Predictions based on a GP model with a compatibility distance kernel can identify the most promising individuals to test and the most promising genotype regions to explore. By judiciously sampling these individuals we can improve the accuracy of our models in optimal regions and perform the same simultaneous topology search and

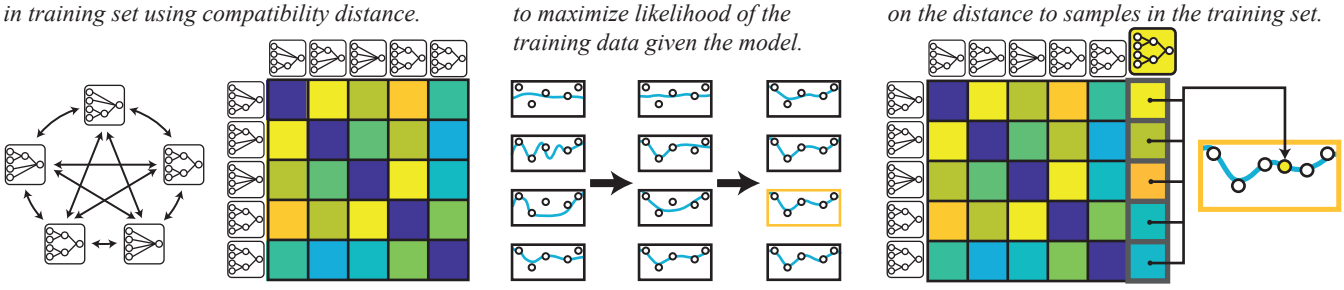

Figure 3.4: *Predicting Performance from Compatibility Distance*
To predict performance based on compatibility distance a kernel matrix must first be created by comparing every individual in our training set to every other with the kernel function. We then train the model hyperparameters to maximize the likelihood of the training set given the model. Fitness for an unknown individual can then be predicted using the compatibility distance from each individual in the training set and the found model hyperparameters.

weight optimization as NEAT, but with greater data-efficiency. Much of the core algorithmic machinery of NEAT can be maintained, but several adjustments must be made to place NEAT into a surrogate-assisted framework.

Preliminary work leading up to this thesis *(Gaier et al., 2018b)* introduced operators that allowed NEAT to take advantage of the compatibility distance Gaussian process models outlined above. Here we will reintroduce these methods, and simultaneously present a streamlined alternative.

INITIALIZATION This surrogate-assisted variant of NEAT begins just as the original version of NEAT, by initializing a set of minimal networks and evaluating them on the objective function. These initial samples and their fitness bootstrap the algorithm, forming the initial training set of the model and the initial population. The distance between all samples is computed and the GP model is trained.

SURROGATE-ASSISTED EVOLUTION The population is evolved according to the standard mechanisms of NEAT: individuals are grouped into species, a number of offspring are assigned to each species based on the mean fitness of their members, tournament selection is used to select parents from each species and variation is performed within each species to produce a new population.

The compatibility distance between the newly produced individuals and all individuals in the training set is then calculated. Based on the model and this distance, we calculate the utility of sampling (i.e. evaluating) each new individual. We reward individuals with high

predicted fitness and high uncertainty, using the UCB acquisition function. We then repeat the evolutionary process, grouping the new individuals into species, and use this utility value in place of fitness to determine the number of offspring awarded to each species and the winners of the intra-species tournaments which determine the parents of the next generation.

INFILL AND MODEL UPDATE Surrogate-assisted evolution is repeated a number of times before new samples are selected to be evaluated. These new samples are then added to the training set. The entire process of selecting individuals for evaluation, evaluating them, and adding them to the training set we refer to as *infill*.

We limit the training set to a maximum size, and if adding new samples would extend it beyond that size the oldest samples are replaced. This sliding window approach to our training data serves dual purposes. The first is to keep our models relevant to the current individuals being evolved: as the genotypes become more complex the distance from older, simpler individuals becomes less relevant. Older individuals will not only have lower fitnesses, but as the population explores new spaces older individuals will contain many genes which do not exist in the current population, providing little benefit to prediction.

The second benefit of a limited training set is the computational advantage of performing prediction based on a smaller training set. New individuals must be compared to every individual in the training set, and the resulting matrix of distances between the training set samples must be inverted when creating the Gaussian process model, an operation with complexity cubed in the number of samples *(Rasmussen and Williams, 2006)*. A limited training set of recent individuals ensures a computationally efficient model focused on relevant regions.

3.3.3 *Coping with Complexification*

Modeling in a space which continually grows more complex introduces additional challenges. While in a fixed space every additional sample better prepares the model to predict the next, in an expanding and moving space it does not. The complexity of the genomes increase in such a way that the starting and later populations may share few genes, and so have little predictive power. In the majority of cases, to create accurate models requires many samples per dimension. Genes, and so unique dimensions, are added at a far faster rate than they can be explored. In sparsely populated regions, modeling inaccuracies can result in artificial minima being created by the model. Such inaccuracies can cause us to underestimate the performance of unevaluated samples, and so never choose them for evaluation or infill. In catastrophic cases the disparity between the size of space we are sampling

and the sparsity of samples can result in a model which cannot make any informed decision at all, resulting in an algorithm which explores randomly. In the following sections we present strategies for coping with the mounting complexity of the space.

INFILL Bayesian Optimization is formulated to achieve the goal of locating a single global optimum. Here instead we are attempting to search simultaneously in several different spaces, in the form of varied species. In order to search effectively we must find a way of appropriately allocating our resources, in the form of evaluations on the task, across these different species.

The straightforward approach is to split our resources equally across the most promising species: evaluating the highest utility individual of each species until our budget for that infill iteration is exhausted. This method explores all species at the same rate, even when one is much more promising than the others. A potentially more effective approach could be to instead align the resources we spend on exploring each species with that species' prospects.

To address this resource allocation problem we look back to the principles already used in NEAT. Speciation and fitness sharing are critical components of NEAT. Beyond their stated goal of nurturing and protecting innovation, they also provide a way of focusing resources in promising search regions. The underlying logic of fitness sharing dictates that small high-performing species will expand, large average-performing species will contract, and poor-performing species disappear altogether.

Just as fitness sharing is used to determine the share of offspring awarded to each species during evolution, we propose applying the same principle to determine each species' share of samples to evaluate at infill. The mean utility of each species is calculated and a number of evaluations awarded in proportion. That number of individuals from each species are evaluated on the objective function. In this way new promising species are quickly explored, but well explored species slowly abandoned if no progress is made.

POPULATION MAINTENANCE As generations of surrogate-assisted evolution repeat, the population drifts farther away from known individuals — the regions where reasonable predictions can be made. In typical cases of surrogate-assisted optimization this is not a concern: all individuals occupy the same space and predictions become more accurate as the solution space is explored. With a complexifying genome, however, new dimensions are introduced faster than they can be efficiently explored.

The problem of the number of dimensions expanding at a pace our models sampling cannot match implies two strategies: slow the expansion of dimensions or increase the amount of samples collected.

We examine each strategy with two approaches: (1) reintroducing known samples back into the population, and (2) adaptively adjusting the number of new samples we evaluate at every infill iteration.

By reintroducing known individuals back into the population we inject known genetic dimensions back into the population, pulling the algorithm into better understood space. To do this whenever we update the model, we also add one member of the training set for every member of the population, with the most recent added first, effectively doubling the breeding pool for that generation. This larger collection of individuals is divided into species and recombined to form a new population of the standard size. Much of the new population will have known individuals as one or both parents, and this brings the population back towards known genetic dimensions, allowing more accurate predictions of their fitness.

If the algorithm stops improving converges on a local optimum and stagnates we "resolve" the population, replacing fitness approximations with true values. If enough generations pass without improvement, the entire population is evaluated on the task, revealing any individuals in the population which would achieve higher fitness but were never chosen for evaluation.

If no better individuals are found then the speciation and recombination of NEAT repeats, and the entire population again is evaluated on the task. Every individual evaluated is added to the training set, and this NEAT evolution continues until either a better solution is found or the entire training set is replaced with new individuals. At that point the GP model is reconstructed and the algorithm begins again with a diverse, complex, but known population. With the search space once again well-modeled the process of surrogate-assisted evolution, model update, and population update resumes.

The second approach to keeping the population within known genetic spaces is simply to increase the density of sampling as exploitation of the current explored niches becomes less fruitful. In this case whenever new samples are evaluated we check whether any of these new samples improves on our current best solution. In the case that it is better, we reduce the number of samples taken at the next infill iteration. In the case that none of our new samples improve on our all time best we increase the number of samples taken at the next infill iteration.

Adaptively adjusting the number of individuals to evaluate allows sampling to keep pace with the needs of optimization, collecting data more densely when improvements are difficult to detect, and lightly when gradients of improvement are easily found. In the case that our model is very accurate and we are regularly finding better individuals the algorithm will tend towards a 1+1 surrogate-assisted EA; in the case that our models are very poor the algorithm will tend

toward original NEAT, removing the need for an additional operator to 'resolve' the population.

Overview of Fixed vs. Adaptive Approaches

We group the strategies for coping with complexification into two approaches: a *Fixed* approach which samples each species evenly with a fixed infill budget, reintroduces members of the population at each infill iteration, and 'resolves' the population when it becomes stagnant (this is the approach presented in *(Gaier et al., 2018b)*); and an *Adaptive* approach which apportions evaluations to each species based on fitness, and adapts the number of evaluations at each infill iteration. These approaches are illustrated in Figures 3.5 and 3.6, and their difference summarized in Table 3.1 below.

Table 3.1: Surrogate-Assisted NEAT Variants Summary

Algorithm	**Initialization**	**Model**	**Evolution**	**Infill**	**Population Update**	**Recovery**
Fixed	Minimal Networks	Sliding window	Highest utility individuals win tournament	1 per species	Reintroducion of evaluated individuals	Evaluate entire population
Adaptive	Minimal Networks	Sliding window	Highest utility individuals win tournament	Proportional based on fitness sharing	Elitism and recombination only	Adaptively increase number of evaluated individuals

3.4 RESULTS

BENCHMARK: CART-POLE SWING-UP We test our approach to surrogate-assisted neuroevolution first on a classic benchmark control problem, the cart-pole swing-up. The system begins with a cart on a two dimensional track with a pole hanging below it, with the objective of swinging the pole into an upright position and maintaining it in a balanced state. This task is more difficult than benchmarks used in many evolutionary computation publications, such as pole-balancing, and cannot be solved with a linear controller *(Raiko and Tornio, 2009)*, requiring networks to grow beyond their initial minimal state.

The known state of the system is the position and velocity of the cart, and the angle and angular velocity of the pole. Inclusion of a bias node results in a total of 5 inputs, with a single output node specifying a command to the system as a percentage of the maximum force. The cart-pole system used here is composed of a 2 kg cart and a 0.5 m pole weighing 0.5 kg. The maximum force which can be applied to the cart is 10 N, with control signals sent to the system at every 0.02 seconds, for a total of 5 seconds.

Controllers are rewarded for the most consecutive time steps in which the pole is held upright. If, for example, the pole is held upright for 25 time steps, falls, then is swung back up and held upright for an additional 15 time steps the controller is only awarded a fitness of

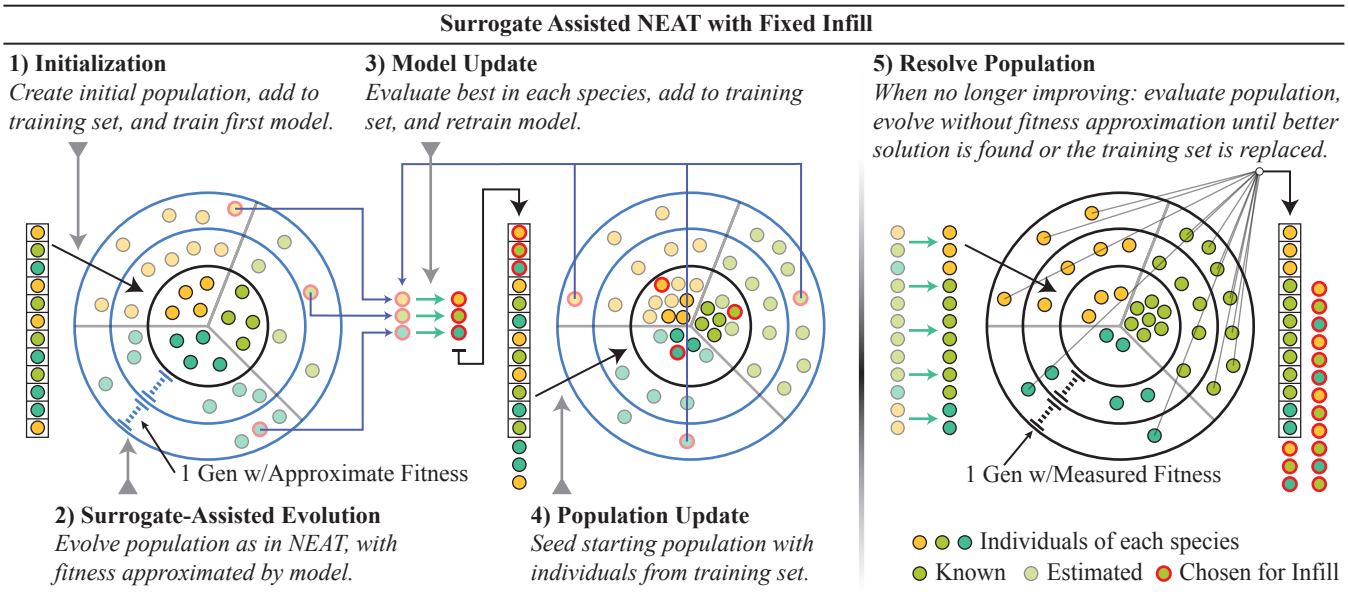

Figure 3.5: *Surrogate-Assisted NEAT (Fixed)*
1) An initial population of minimal networks is used as a training set, serving as the basis for the initial surrogate model.
2) The population evolves according to the NEAT algorithm, with fitness replaced by the approximations of the model.
3) From each species, the individual with the highest predicted performance is selected and evaluated on the task. These new samples are placed in the training set, replacing the oldest samples if the training set has reached its maximum size.
4) The training set and population are combined to form a new population, and the process repeats from step 2.
5) When fitness is no longer improving, the entire population is evaluated and added to the training set. If none of these individuals are an improvement, a new generation is produced with NEAT, which is evaluated and added to the training set. Evolution continues in this way until a) a better individual is found, or b) the entire training set has been replaced with new individuals. The model is then retrained and the algorithm resumes from (2) with this new population and training set.

25, not 40. This avoid the case of rapidly spinning the pole to achieve middling fitness value. Fitness is only awarded for time steps in the second half of the trial, for a maximum fitness of 100.

NEAT has a large number of hyperparameters, too many to test and tune exhaustively. Instead we conducted preliminary testing with different levels of variation per generation, based on the hyperparameters presented in the canonical NEAT article *(Stanley and Miikkulainen, 2002)*. The probabilities to add nodes and connections, reenable nodes, perform crossover, and mutate weights were scaled by $2, 1, \frac{1}{2}, \frac{1}{4}$, and $\frac{1}{8}$: preliminary tests showed NEAT's best performance when variation was scaled by $\frac{1}{2}$ and so these hyperparameter values were used.

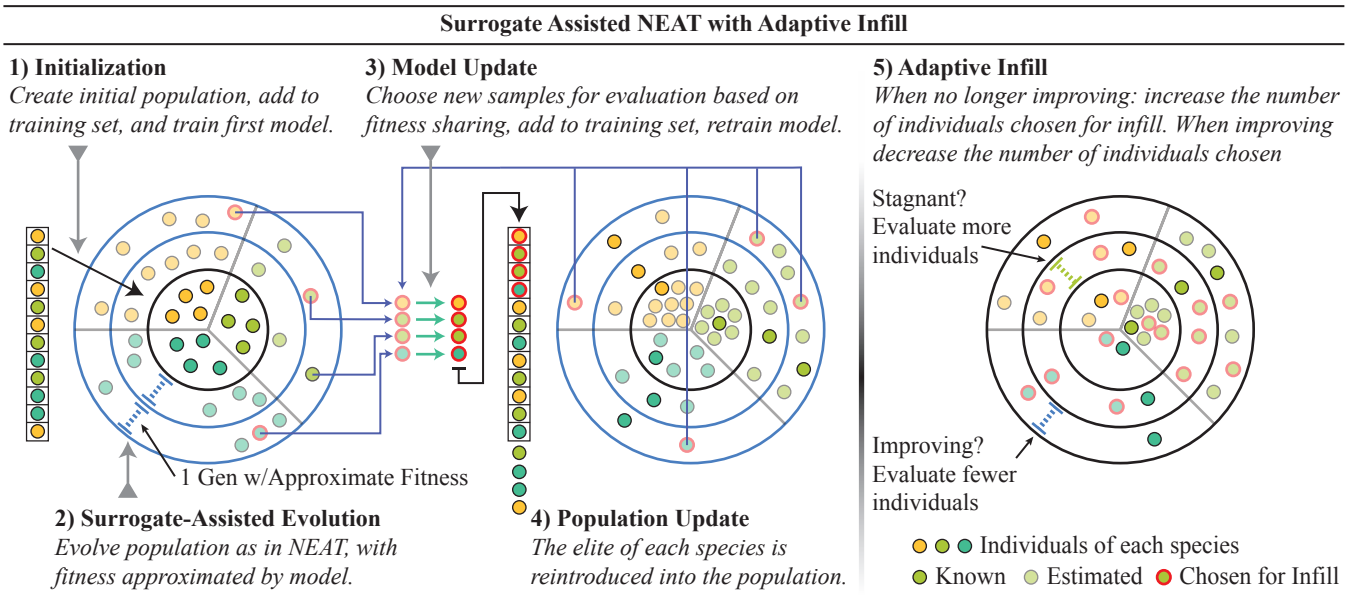

Figure 3.6: *Surrogate-Assisted NEAT (Adaptive)*
1) An initial population of minimal networks is used as a training set, serving as the basis for the initial surrogate model.
2) The population evolves according to the NEAT algorithm, with fitness replaced by the approximations of the model.
3) The best individuals of each species are chosen to be evaluated and added to the model. The number of individuals chosen from each species is based on the mean fitness of each species and the number of individuals of that species already in the training set.
4) The best found individual in each species, the species elite, is copied unchanged into each new population, but never chosen for infill.
5) When fitness is not improving, the number of individuals chosen for infill increases. When fitness is improving, we choose fewer individuals for infill.

In the fixed infill variant of SA-NEAT 4 generations of evolution took place before selecting 4 infill individuals to add to the population. These were taken from the top 4 species, except in the case where less then 4 species were present, in which case the highest performing individuals were taken in their place. In the adaptive infill variant 4 individuals were selected at the first iteration and after that this number was adjusted after every infill iteration.

A training set of 512 individuals was maintained, and the population "resolved" if 128 individuals were added to the training set without improvement. To keep the same amount of variation in one sampling iteration of SA-NEAT as would occur in a single generation of NEAT, the rates of variation were decreased by $\frac{1}{4}$ ($\frac{1}{8}$ of the hyperparameters in *(Stanley and Miikkulainen, 2002)*). Table 3.2 outlines the hyperparameters and their relationship:

Table 3.2: Hyperparameter Values and their Derivation

# of Species	-	4
Gens Per Infill	-	4
Population Size	-	128
Variation	Base / Gens Per Infill	(Published / 8)
Inds Per Infill	# of Species	4
Training Set	Inds Per Infill × Pop Size	512
Stagnation	Population Size	128

3.4.1 *Data-Efficiency*

A comparison of performance between NEAT and both variants of the SA-NEAT approach on the swing-up task is shown in Figure 3.7. We compare only the number of fitness evaluations performed on the cart-pole simulator: fitness predictions using the surrogate model are not counted.

Tracking the fitness of the best performing network at each evaluation (Figure 3.7, top left) we find that the variants of fixed SA-NEAT perform much the same in the early stages – only overtaking NEAT once a 'resolve' operation clarifies a more complex space. The adaptive infill variant, however, consistently outperforms NEAT at every stage, with even the bottom quartile of runs outperforming the median NEAT run.

The number of connections in the networks which solved the task (Figure 3.7, top right) are quite similar in the Adaptive SA-Neat case and the standard NEAT algorithm. In the Fixed SA-NEAT case complexification continues at a more rapid rate, and the resulting networks are more bloated. Though in relative terms the networks use nearly double the number of connections, in absolute terms the difference is not great.

We have defined a solve state for our task, and evaluate each approach based on the number of evaluations required to reach this state (Figure 3.7, bottom left). We see that both surrogate assisted methods solve the algorithm with fewer evaluations than the original NEAT, typically only requiring half the interaction time.

3.4.2 *Model Performance*

We are able to perform more data-efficient optimization by applying our surrogate-assisted version of NEAT, but it is not clear that this is to the model's credit. Substantial changes have been made to NEAT operation – it could be that these improvements are result of these changes, not the predictive power of our models.

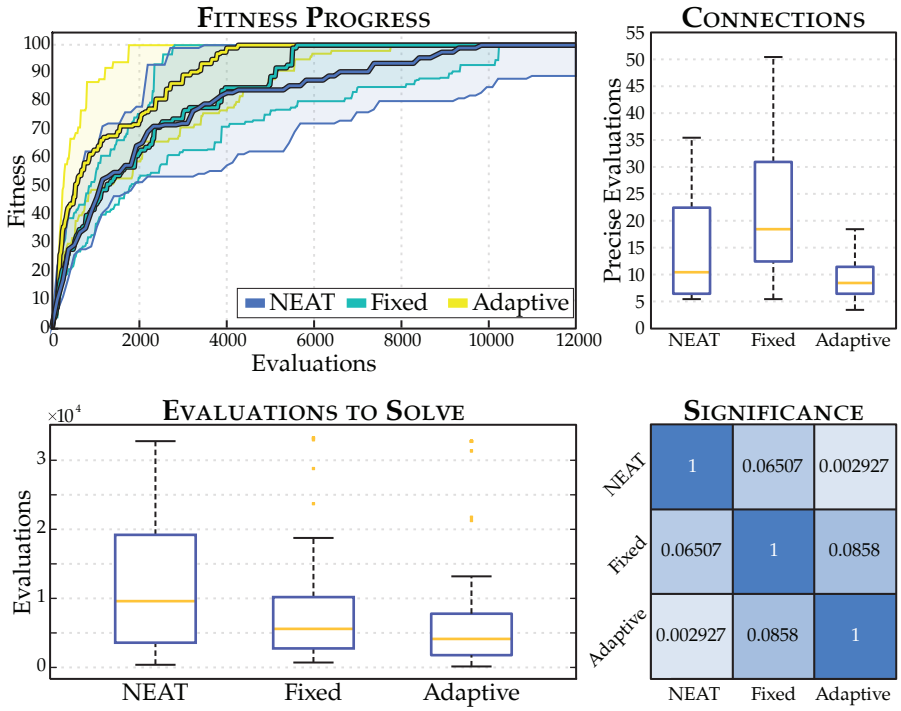

Figure 3.7: *Cart-Pole Swing-Up: Optimization Performance*
Optimization performance of NEAT (*NEAT*) and Surrogate-Assisted NEAT with fixed infill (*Fixed*), or adaptive infill (*Adaptive*). Performance measured over 50 replicates.
Top left: Fitness progress of each algorithm per evaluation. Dotted line indicates median fitness, with shaded regions indicating one quartile above and below the median.
Top right: The number of connections in the networks which first solved the task in each network.
Bottom: Evaluations required to solve the task, and the significance of their difference.

To validate that it is indeed the model predictions which provide us the advantage, we introduce two experimental algorithm variants: one which makes use of a perfect model, and one which uses a random model. In both experiments the adaptive SA-NEAT algorithm is used, but rather than the compatibility distance Gaussian process (CD-GP) model a substitute model is used.

To better understand the upper bound of our algorithm's performance we introduce a perfect *Oracle* model: whenever a prediction is required of the model we evaluate the solution in the simulator and return as a 'prediction' the true fitness value. These evaluations do *not* count towards the evaluation budget, but are only a way of creating and idealized model whose predictions are always correct.

In order to estimate a lower bound in addition to a model with perfect information, we introduce a model with no information. To determine if it is not our model which is providing the benefit, but simply a more robust algorithm, we also test with a *Random* model:

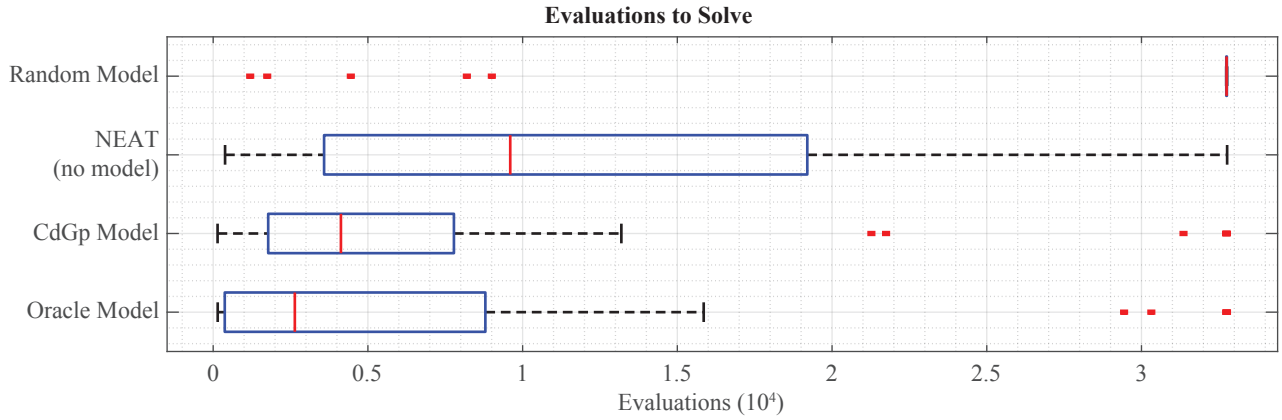

Figure 3.8: *Model Comparison*
Comparison of evaluations needed to solve cart-pole swing-up with various modeling benchmarks. Using the 'adaptive' variant of SA-NEAT for optimization the compatibility distance Gaussian process model (*CD-GP*) is replaced with a model which gives random predictions (*Random Model*) or a model which always gives a perfect prediction (*Oracle Model*). The CD-GP model provides considerable benefit over the random model, despite the difficulty of the modeling problem. Though the median performance of SA-NEAT when using the perfect model is better than when using the Gaussian process model, the difference is not significant ($p = 0.1365$ using two-tailed Mann-Whitney U test), implying the rank accuracy of the models is sufficient to drive the evolutionary process. Performance of standard NEAT are also shown for comparison. Results shown are a distribution over 50 replicates.

whenever a prediction is requested from the model this model simply returns a random number between 0 and 1.

The number of evaluations required by SA-NEAT to solve the task using these models is shown in Figure 3.8, with the number of evaluations when using the CD-GP model and the original NEAT shown for reference.

When a random model is used, we see that algorithm performance collapses as expected, though surprisingly several outliers *do* manage to solve the task even when only noise is given as model feedback.

The Oracle model, despite its perfect accuracy, delivers only an incremental improvement over the CD-GP model. This meager improvement can be more easily understood in the context of how predictions are used by the algorithm. Comparisons between individuals, from relative species fitness to individual tournaments, are based solely on rank. As fitness is only considered in this cardinal fashion, any additional predictive power beyond that which improves rank accuracy provides no additional benefit. Furthermore, in the case of tournaments within species, even an inaccurate ranking within the population is easily forgiven, as only the comparison of a few individ-

uals must be accurate. The most common error in such tournaments, the mistaken ranking of individuals which have very similar fitness, is also the least consequential. This allows the trend toward higher fitness to be maintained even with lower accuracy models.

GP models make fitness predictions which are accurate enough to make informed decisions about which individuals and species are promising and which are not. The rank-based methods used within SA-NEAT are particularly forgiving of model inaccuracies, and even perfect models provide only marginal benefit.

3.4.3 *Scalability*

CONTINUOUS CONTROL: QUADRUPED ANT To test the SA-NEAT approach on a higher dimensional problem, we compare its performance on the 3D quadruped 'ant' running task. This ant running domain was introduced as a high-dimensional continuous control task for Deep Reinforcement Learning in *(Schulman et al., 2015)*. The system has a state space of 28 dimensions and an output space of 8 joint torque values: one for the hip and knee of each of the 4 legs. The initial NEAT minimal network connecting all inputs to outputs then has 224 weights, compared to the 5 weights of the cart-pole system.

The standard OpenAI gym *(Brockman et al., 2016)* implementation used was used together with the open-source physics simulator Bullet *(Coumans and Bai, 2016–2018)*. The simulation was run for 250 time steps[1], with fitness awarded for moving forward with minimal effort and maximum smoothness:

$$\text{fitness} = v - 10^{-6}\|u\|^2 - 10^{-3}\|f_{impact}\|^2 + 0.05 \tag{3.5}$$

where:

$$v = \text{forward velocity} \tag{3.6}$$
$$u = \text{vector of joint torques} \tag{3.7}$$
$$f_{impact} = \text{impact forces on feet} \tag{3.8}$$

The same hyperparameters used for NEAT and SA-NEAT in the swing-up task were used here. As the ant task has no explicit solve state, and is much more expensive to simulate than the cart-pole, we limit the number of evaluations to 8000 and perform only 20 replicates. As the adaptive variant of the SA-NEAT outperformed the fixed version, we compare only it to NEAT.

1 The Bullet implementations of RL environments are more difficult than the more commonly used closed-source MuJoCo environments. In addition 250 time steps is significantly fewer time steps than is typical for this task in reinforcement learning experiments, and so results are not directly comparable to those in the literature. The purpose of these experiments is only to establish the benefits w.r.t. NEAT: more thorough comparisons with previous work will be presented in a future publication.

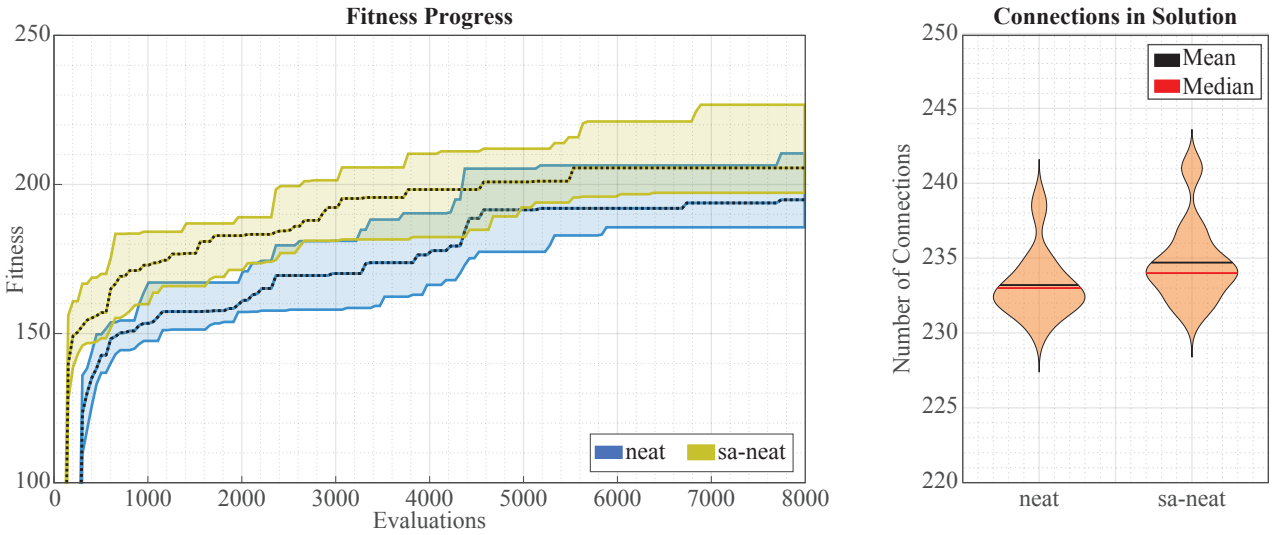

Figure 3.9: *Quadruped Ant*
Comparison of best fitness values found over 20 trials of SA-NEAT and NEAT on the Ant 3D quadruped running task. Fitness progress of each algorithm per evaluation us shown, with the dotted line indicating the median best fitness over all replicates, and the shaded regions indicating one quartile above and below the median.

RESULT Even in this higher dimensional problem SA-NEAT outperforms NEAT (Figure 3.9). With networks of similar complexity, the median performing solutions found by SA-NEAT after 3038 evaluations reach the same performance as those of NEAT after 8000, a savings of more than half. This not only confirms our earlier experiment, but also shows that SA-NEAT is able to navigate a high dimensional weight space and search the space of possible topologies.

While the swing-up benchmark is not a trivial task the space of solutions, even in the complexifying case, is relatively limited. With a minimal topology of five inputs and one output it begins as only a five dimensional problem. The 3D quadruped, on the other hand, begins in a space that is more than 200 degrees of freedom. As the compatibility distance kernel is independent of the dimensionality of the underlying genotype, our models are still able to make useful predictions in this space with only hundreds of samples.

3.5 DISCUSSION

We introduced a surrogate-assisted variant of NEAT, SA-NEAT, as a data-efficient method of performing neuroevolution in computationally expensive problems. By taking advantage of the phylogenetic information produced as a byproduct of the evolutionary process, we created a new kernel to judge similarity of neural networks based on their shared genes. By using GP models built with this kernel we

are able approximate the performance of individuals, allowing us to achieve similar results as NEAT with a fraction of the evaluations.

Fewer evaluations does not necessarily guarantee faster optimization: when the fitness function is evaluated in simulation, there is always a trade-off between the cost of modeling and evaluation. By using a sliding window to limit the model to using only recently evaluated samples we bound its complexity, and it is possible that both accuracy and performance could be further improved with even more purposefully constructed models. Gaussian process models are expensive to construct and perform predictions with, and as only correct ranking of solutions is required, simpler, faster models may suffice. The key component of the compatibility distance measure could just as easily be applied to other non-parametric or kernel models, such as k-nearest-neighbors or support vector machines.

In the presented approach, though species diverged into varied and distant genealogies, a single training set and model were used for prediction. Due to the squared exponential relationship in the kernel, individuals in more distant species have little if any effect on the predicted performance, yet are still considered in the comparison. Producing surrogate models with individuals only from within a single species would reduce the needed number of comparisons. Apart from computational concerns, species specific models could also have more predictive power, as the hyperparameters of the model could more accurately reflect the particular genotype region, rather than their likelihood over the entire training set.

NEAT is also used to evolve compositional pattern producing networks (CPPNs) *(Stanley, 2007)*, indirect encodings used to produce neural networks *(Stanley et al., 2009)*, images *(Secretan et al., 2011; Nguyen et al., 2015a)*, and 2D and 3D objects *(Clune and Lipson, 2011; Gaier, 2015; Lehman et al., 2016)*. Whether our approach is as successful in evolving indirect encodings remains to be seen, but as many engineering domains rely on expensive simulations, a data-efficient method of evolving CPPNs would allow their application in real world design problems.

To create truly interesting artifacts, the ability to evolve the solution structure itself is necessary. Necessary but expensive —— especially in real-world domains. Creating solutions which vary and grow in is a powerful capability of evolutionary computation. A surrogate-assistance method which supports complexification allows us to leverage this ability of evolution even in expensive domains. Combining data-efficient machine learning with neuroevolution ensures that complexity building, diversity preserving, novelty seeking, deception avoiding abilities of evolutionary approaches can still be applied, regardless of the complexity of the challenges presented.

4

QUALITY DIVERSITY AND EXAPTATION

Some figures, text, and ideas in this section are based on the following works which have either been previously published or are in review:

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2019). "Are quality diversity algorithms better at generating stepping stones than objective-based search?" In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 115–116.

4.1 INTRODUCTION

This manuscript is focused on the development of algorithms which can efficiently produce sets of solutions which are both diverse and high performing. Though diversity and performance are two separate objectives, it would be a mistake to see them as trade-off. The path to high performance is often not a straight line through ever better solutions. In these cases exploring a diversity of paths is not a distraction to improving performance — it is a necessity.

In initial experiments optimizing aerodynamic designs with objective search and NEAT we were unable to evolve meaningful complexity. Once a simple aerodynamic shape was found, for example a wedge, it was difficult to discover anything else. Finding more complex shapes, such as teardrops or wings, required first exploring less aerodynamic designs — which objective search discarded. This premature convergence reminded us of previous experiments illustrating NEAT's difficulty in producing images *(Woolley and Stanley, 2011)*.

The failure of NEAT to reproduce target images was laid at the feet of objective-based search, with novelty search techniques presented as a panacea. Novelty search could overcome deceptive fitness signals by ignoring all fitness signals, and instead concentrate on collecting interesting "stepping stones" which would allow a greater range of solutions regardless of their performance.

Innovation is often the result of exaptation — the reuse of an existing solution to a new purpose — giving such algorithms great potential for discovering truly novel and interesting artifacts. Curious for such an oft-cited claim though, there is has been little analysis of the "stepping stone" phenomenon in the literature. If novelty search-inspired approaches, including QD, really collect stepping stones they should be able to tackle the image matching task which objective-search failed at so completely.

In this chapter we will examine if QD indeed collects intermediate if not immediately useful solutions, and in doing so solve problems unsolvable by objective search. We begin by revisiting the processes of biological and technological innovation that inspired the idea of stepping stone in evolutionary computation, followed by a detailed examination of the illumination process of MAP-Elites to see if it truly collects stepping stone to solve the image matching task. Though preliminary, even a cursory study of such a profound phenomenon is useful for providing new insights and understandings, or at the very least confirming our intuitions.

EXAPTATION The route to the solution of complex design problems often lies through intermediate "stepping stones" which bear little resemblance to the final solution *(Stanley and Lehman, 2015)*. By greedily following the path of greatest fitness improvement, objective-based search often overlooks and discards stepping stones which might be critical to solving the problem. In these deceptive cases, where the path to high-performing regions leads through poor-performing way points, optimization can fail dramatically *(Woolley and Stanley, 2011)*.

Natural evolution can be a source of inspiration to better understand how complex problems can be solved in ways other than iterative improvement. How traits can be improved and refined through natural selection is intuitive, but it is difficult to see how *new* traits can arise — especially those which only provide a selective advantage when fully formed. Without a designer new traits cannot be created to fulfill some need, evolution must work with what it has: function must follow form. When innovation does occur, it tends to arise as an *exaptation (Gould and Vrba, 1982)*, a co-option of an existing trait for a new use.

Flight did not arrive at a stroke. While there is considerable disagreement about its origins and development *(Segre and Banet, 2018)*, there is little debate that feathers were instrumental. The story of the evolution of feathers from cylindrical follicles to highly complex structures is not a simple one *(Prum, 1999)*, but fossil evidence shows that feathers predate flight *(Jones et al., 2000; Norell et al., 2002)*. The original purpose of feathers has been variously explained as heat regulation *(Regal, 1975)*, display *(O'Connor and Chang, 2015)*, repelling water *(Dyck, 1985)*, and others. Whatever the feather's original use, we know that it was dramatically co-opted for aerodynamic purposes. Entire body plans were molded around exploiting these new capabilities — turning reptiles into hollow-boned birds. It is by this process of co-opting existing traits for new functions that evolution is able to innovate without inventing.

The history of technology and invention is likewise littered with examples of exaptation. Some inventions were accidents recognized by their creators to have other uses: microwave ovens were the result of

magnetron experiments at a defense contractor *(Osepchuk, 2009)* and Teflon a byproduct created when testing new refrigerants *(Plunkett, 1986)*. Some were designed for one use, but repurposed by others in a different context: the CD-ROM was patented in 1970 to solve the problem of wear and tear on vinyl records, only to later be adapted and refined for data storage *(Dew et al., 2004)*; the world's then darkest pigment, Vantablack, was created by a satellite company to solve an optics calibration problem before being adopted by the art world *(Maynard, 2016; Le, 2020)*. Exaptation is so important in drug design that the the pharmaceutical industry actively searches for new purposes of existing drugs. "Drug repositioning", is widespread and successful: well-known treatments for conditions from hair loss and erectile dysfunction to Alzheimer's and Parkinson's, among dozens of others, were adapted from drugs designed for unrelated ailments *(Ashburn and Thor, 2004)*.

Just as a drug's side effects can become their chief benefits, in biology new uses for an organism's existing traits may be revealed when they shift from one environmental niche to another, in a process known as ecological fitting *(Janzen, 1985)*. The jointed skeletons of ancient arthropods provided them with the mechanical components necessary for methods of locomotion that were not reliant on being submerged in water, allowing them adapt quickly to life on land *(Cowen, 2013)*. Similarly, traits which were required only to survive in one niche may prove to be wildly successful in another, as in invasive species like cane toads *(Phillips and Shine, 2004)*, Africanized bees *(Roubik, 1980)*, and kudzu *(Forseth and Innis, 2004)*.

Novel traits may also arise by finding function for structures with no previous use. "Spandrels" *(Gould, 1997)*, are structures with no particular advantage or disadvantage but which arise as a consequence of other adaptations. Supporting the Irish elk's (*Megaloceros giganteus*) 45 kg antlers required adaptation of the vertebrae, resulting in a prominent hump on its back (Figure 4.1). These spandrel humps were further enlarged and, as we know from Cro-Magnon cave paintings, decorated with long brightly colored hairs, presumably for mating display *(Lister, 1994)*.

Dramatic departures from tuned body plans are still possible under certain conditions. Small changes in genes governing development can have drastic effects, acting as "macromutations". The clustering patterns of flowers on stems are based on only one or few loci *(Gailing and Bachmann, 2000; Moritz and Kadereit, 2001)* and the exposed kernels of 'ears' of corn, which are the result of a change in just 5 gene loci *(Wang et al., 2005)*. Developmental mutations can make wholesale changes to the body plan, such as fruit fly antennae or wings replaced by legs *(Dietrich, 2000)*. These 'hopeful monsters'*(Goldschmidt, 1982; Theißen, 2006)*, which mark a radical phenotypic departure, are an extreme example of the power of developmental mutations.

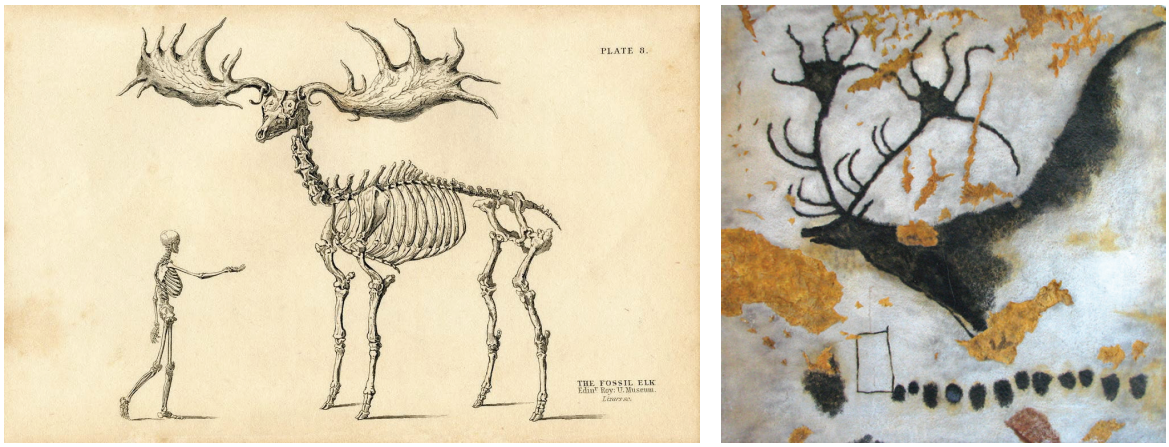

Figure 4.1: *Spandrels of the Irish Elk.*[1]
Left: Supporting the weight of the Irish elk's antlers required other adaptations, namely the elongated vertebrae above the shoulders.
Right: The resulting hump on back of the Irish elk was repurposed as part of a mating display, with long bright colored hairs.

While such extreme mutations are rare — and successful ones even more so — rarity does not imply unimportance. When such mutations shift individuals into new ecological niches, they have the potential to open up entirely new evolutionary lineages. The implausibility of the gradual evolution of the turtle body plan, and their sudden appearance in the fossil record, leads biologists to consider them the progeny of hopeful monsters *(Rieppel, 2001)*. Some biologists subscribe such importance to the impact of hopeful monsters as to credit them as a possible explanation for the origins of flowering plants and the Cambrian explosion *(Theißen, 2006)*.

Such macromutation seems counter to the idea that novel structures do not arise from adaptation. But the mutation of a hopeful monster is not an adaptation to its environment, but a *maladaptation* which shifts the organism into a different ecological niche. Once in this different mode of living additional adaptations arise in response to selection pressure, fitting the organism to its environment and making it less monstrous. In this way developmental mutations of hopeful monsters fit the underlying theme of exaptation: a *functional* shift in a trait opens up new avenues for conventional methods of adaption to follow.

We see from biology that innovation comes not by the gradual accumulation of positive improvements but by functional changes in preexisting traits that open up new evolutionary trajectories. But evolution has no way of predicting what traits will be useful in the future, or what traits might set the stage for subsequent modifications.

Such uncertainty also plagues the process of human innovation. In order to understand the processes of technological development and human creativity economists and social scientists have also examined the role of exaptation *(Mokyr, 1991; Dew et al., 2004; Andriani and*

Cattani, 2016). Through this lens they trace innovation to firms which (1) accumulate knowledge and capability without anticipation of their later use, and (2) apply their existing expertise to new areas. Can algorithms built in line with these principles do more than adapt, but innovate?

Early efforts to design algorithms which explicitly leverage exaptation *(Mouret and Doncieux, 2009)* identified that multiple selection pressures were necessary to explore genotypes with no immediate benefit. Quality diversity (QD, see Section 2.1.2) algorithms introduce additional selection pressures by requiring solutions to compete only with phenotypically similar solutions — effectively rewarding solutions with less common phenotypes.

MAP-Elites (see Section 2.1.3) implements this pressure for phenotypic diversity through an explicit niching scheme, creating a dynamic akin to ecological niching *(Janzen, 1985)*: initially a family of solutions is only able to survive in a small cluster of niches, competing amongst themselves and becoming well-adapted for their particular niche; this population serendipitously discovers a genotype which, with small variation, is successful across a range of niches and so rapidly expands; now widespread and subject to varied selection pressures based on their niche, the population fragments as they adapt to their respective niches, and the cycle repeats. This process can be seen as a method of collecting knowledge and refining expertise while simultaneously testing it in contexts it was not designed for.

QD algorithms were designed to with the intention of collecting stepping stones, in the form of varied phenotypes, which could be put to use in the future even if they were not optimal in the present. Producing a diversity of approaches is designed to foster the serendipitous discoveries of exaptation — but little work has been done to examine if this actually occurs. In this preliminary study we examine the optimization process of QD, looking specifically for evidence of stepping stone collection, using a test case famous for frustrating objective-based search.

4.2 BACKGROUND

THE PICBREEDER EXPERIMENT The deficiencies of a search based only on improving an objective was made explicit in *(Woolley and Stanley, 2011)* with an illustrative example drawn from the collaborative evolution project Picbreeder *(Secretan et al., 2011)*. In Picbreeder users are presented with a set of images, and prompted to select one or more to act as parents for a next generation of evolved images. The parents are combined and mutated to produce a new set of images, which are once again presented to the user for selection. In this way the users evolve increasingly complex images, which can be shared with a community of other users to use as starting points for their

own explorations. The results range from abstract shapes to images resembling butterflies, cars, dolphins, and skulls.

Retracing the steps that users took to arrive at these complex images is both fascinating and instructive. The family tree of an image resembling a skull is shown in Figure 4.2.

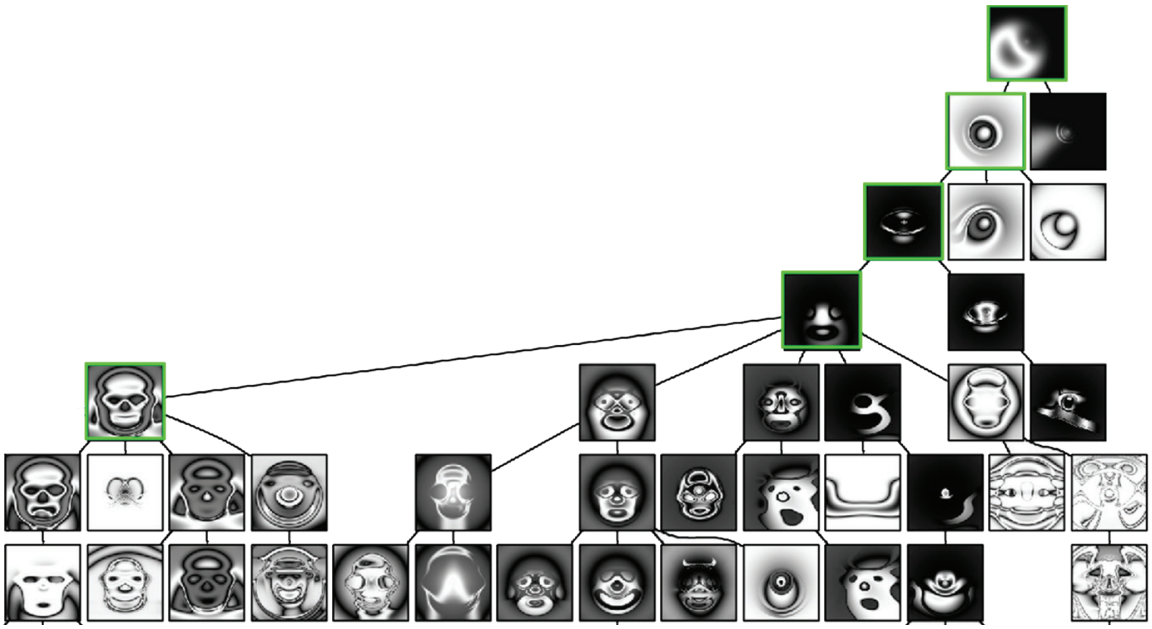

Figure 4.2: *A Picbreeder Family Tree.*
A family tree of individuals found by user driven selection on the Picbreeder website. Ancestors are connected to progeny by lines, though some intermediate links are omitted. Through users undirected wanderings face-like images arose from stepping stones that bear little resemblance to later forms. *(Secretan et al., 2011)*

Examining this genealogy we see that the early ancestors of the skull bears little resemblance to their eventual progeny. Many of these early abstract shapes are the source of other Picbreeder images as well, and it is apparent that the users did not select them with the intention of creating a face. When a image with vague facial features was discovered users created a huge variety of face like shapes, including the skull. This story of serendipitous discovery and refinement is repeated for every one of the more complex images — and this serendipity is a necessary component of their discovery.

In *(Woolley and Stanley, 2011)* the authors attempted to find the same images found in Picbreeder using automated evolution in place of human selection. Interestingly, even though the same method of representing and varying images was used, the algorithm failed at reproducing all but the simplest images. The automated algorithm chose as parents the closest pixel-by-pixel match to the target image, and it was this "single-minded approach" which the authors blamed for the failure.

This finding echoes ideas explored by novelty search *(Lehman and Stanley, 2011a)*, a thought-provoking alternative to objective-driven search. In its purest form, novelty search abandons traditional objectives in favor of exploration of phenotypic space, such as robot behaviors or design features. By exploring the space of possible behaviors selection pressure is biased toward "novel"solutions rather than only those which bring the algorithm closer to a stated goal. Novelty-based methods operate by collecting a variety of "stepping stones" which may lead to better solutions, rather than searching for better solutions explicitly.

Looking for "interesting" results is an approach which mirrors the one taken by users of Picbreeder, up to a point: users explore the space of images with different goals, — or no goal at all — until they find an image which resembles something they recognize, and then purposefully hone that resemblance. A directed component is similarly necessary to attack this picture matching task. Combinations of novelty search and objective search, known as Quality Diversity (QD) algorithms *(Cully and Demiris, 2017)* borrow from both fields to produce a large number of high-quality, but qualitatively varied solutions to a problem.

Just as organisms shift into, then adapt to, new modes of living and users explore Picbreeder by looking first for the interesting, then sharpening them to resemble the familiar, QD algorithms discover then refine functionally different solutions. MAP-Elites *(Mouret and Clune, 2015)* (see Section 2.1.3) uses a niching process to collect individuals with varied behaviors, and optimize them within their own niche. Adaptations which improve performance in one niche may prove to be the necessary stepping stone to improve performance in another. We will examine if such an algorithm — which collects stepping stones, improves them, and constantly tries to apply knowledge in one function niche to another — can tackle problems which require innovation, like the Picbreeder image matching task.

EVOLVING IMAGES The evolution of images in Picbreeder is performed by evolving directed graphs called Compositional Pattern Producing Networks (CPPNs) *(Stanley, 2007)*. CPPNs operate much like neural networks: given a pixel grid they take as inputs the coordinates of a pixel (e.g. x,y) and output a value to color this pixel (e.g. a RGB value or gray scale intensity). In this way an image is encoded as a mathematical function of the input coordinates, and so a single genotype can be expressed as an image of any resolution. Nodes of a CPPN have a variety of activation functions, such as sin and Gaussian, allowing for the emergence of repetitive and symmetric patterns. Their similarity to neural networks allows CPPNs to be evolved with the NEAT *(Stanley and Miikkulainen, 2002)* algorithm (see Section 3.2), and so increase in complexity by adding nodes and connections.

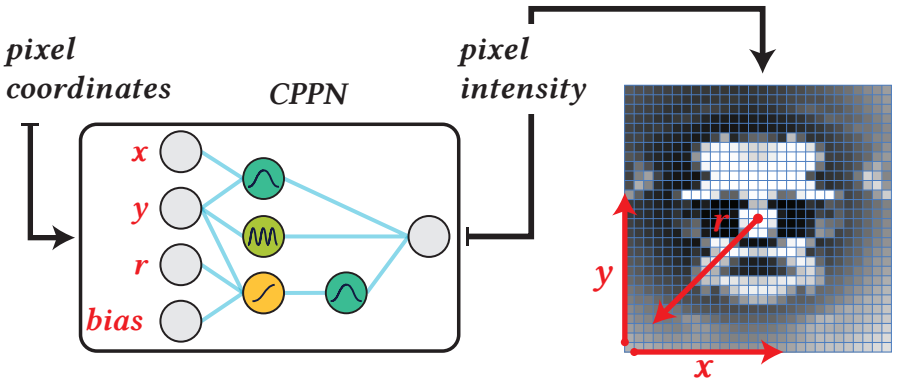

Figure 4.3: *Expression of a CPPN Genotype.*
CPPNs are a special class of neural network with weighted edges and a variety of activation functions. As input CPPNs take a set of coordinates and output a value for each coordinate position. In the image generation examined in this chapter these input values are the x-coordinates, y-coordinates, and distance to the center of the image (r) and the output is a pixel intensity.

CPPNs were used in both Picbreeder and Wooley's subsequent attempts to match the resulting images *(Woolley and Stanley, 2011)*. Since their introduction they have seen application not only for generation of 2D images *(Nguyen et al., 2015a; Nguyen et al., 2016)*, but also 3D shapes *(Clune and Lipson, 2011; Lehman et al., 2016)*, video game content generation *(Hastings et al., 2009)*, soft robot design *(Cheney et al., 2014)*, and one of the earliest methods of optimizing large neural networks *(Stanley et al., 2009)*.

Despite these successes CPPNs have properties which can make them difficult to optimize. In NEAT, when a new node is added care is taken to add it in a fitness neutral way. Such incremental changes preserve a valuable property for optimization: small changes in the genotype result in small changes in the phenotype. The varied activations of CPPN nodes means that the addition of a single node or connection can cause dramatic changes to the phenotype.

CPPNs created with fixed structures and activation functions trained with supervised learning methods *(Ha, 2016; Ekern, 2019)* [2] have shown to be able to perform image matching tasks much better than evolutionary methods which grow structure like NEAT.

The success of alternate representations in image matching tasks should not cloud the purpose of our experiments. The wild discontinuities in the CPPN search landscape can impede gradual methodical progress towards a goal – but is also fertile ground for hopeful

2 It is arguable whether these approaches, which abandon core features of CPPNs such as varied activation functions and complexification, should be referred to as CPPNs or simply 'coordinate to pixel' neural networks — but they are inspired by CPPNs and their creators label them as such.

monsters.[3] While objective search has been shown to fail in such landscapes, user-driven selection has shown dramatic success, revealing the potential rewards of their effective navigation.

4.3 APPROACH

We explore the ability of quality diversity algorithms to find solutions in a domain that requires traversing low fitness stepping stones. We tackle the problem of matching images found by users on the Picbreeder website — the same problem which so dramatically illustrated the failure of objective based search in *(Woolley and Stanley, 2011)*.

The goal of the optimization process is the reproduction of a target image found through a process of user-driven evolution. Users on the Picbreeder website found images by selecting CPPNs from a list as parents – acting as the fitness function of a CPPN evolution algorithm. We take one image found in this search and attempt to match pixel for pixel by algorithmic optimization of CPPNs.

We replicate the work of *(Woolley and Stanley, 2011)* by attempting to find this target image with NEAT. In NEAT genetic diversity is maintained through a process of speciation — this allows for more complex CPPNs to survive even if they have lower fitness, but each subpopulation is still repelled away from low fitness regions. The constant pressure towards higher fitness prevents the kind of meandering exploration which resulted in the original discovery of the image, and is blamed for the approach's failure.

Novelty search was proposed as an antidote to this failure of objective search, but the search for novelty alone will not help us to arrive at a similar image. The space of images is essentially unbounded — so while we will not be led astray by a fitness signal there is no signal to lead us to the target image either.

The promise of quality diversity algorithms is to fruitfully combine the search for novel behaviors with the search for high performing solutions. To illustrate the capabilities of QD to guide evolution we apply MAP-Elites *(Mouret and Clune, 2015)* to the CPPN image matching problem.

The same CPPN representation and variation operators used in Picbreeder and NEAT are used with MAP-Elites, only the structure of the population and selection operators are altered. MAP-Elites maintains an archive of solutions, known as elites, stored in "niches". New solutions are created by mutating these elites, which act as the parent population. Child solutions are then evaluated and assigned

3 Hopeful monsters are typically connected to mutations in developmental genes — which fits neatly with the developmental inspiration of CPPNs.

a niche based on their features [4]. If this niche is already occupied by an elite, the individual with the higher fitness is placed in the niche and the other discarded. By repeating this process a set of increasingly optimal solutions are discovered for every combination of features. We refer to the archive of elites as a whole as a "map".

We explore two features, one genotypic and one phenotypic, which define the dimensions of the map. The first, genotypic complexity, is simply the number of connections in the network. Maintaining a population with a diverse number of connections protects genetic diversity. Individuals which recently evolved additional structures are protected from having to compete with their small and heavily optimized cousins. Reserving a place in the population for simpler networks prevents the entire population from bloating to a large size — and ensures that there is always a pool of lower complexity solutions to branch off from.

Typically phenotypic or behavioral features are hand designed for the domain, but to avoid injecting too much domain knowledge into the algorithm we attempted to design a generic feature descriptor. Periodically all of the images in the map are compiled and used to train an autoencoder *(Hinton and Salakhutdinov, 2006)*; when a new image is produced we compress and reconstruct the image with this autoencoder, examining the pixel-by-pixel error of the reconstruction. Images similar to those which were used to train the autoencoder will be reconstructed well, those different poorly. The reconstruction error can be thought of as a generic approximation of "familiarity." Reconstruction error can also be thought of as a measure of how easily an image can be compressed, providing a measure of phenotypic complexity.

4.4 RESULTS

We compare the ability of NEAT and MAP-Elites to match a target image of a skull found in Picbreeder. [5] CPPNs using linear, Gaussian, signed sigmoid, signed step, and cosine activation functions were evolved. A 28x28 pixel substrate was used, with each pixel having four dimensions: x and y coordinates over the range -2 to 2, the distance to the center ($\sqrt{x^2 + y^2}$), and a bias always equal to 1. The output node of the CPPN is set to an unsigned sigmoid, bounding the output between 0 and 1, and this output is taken as the intensity of the image.

4 As we are discussing images we use 'feature' interchangeably with the term 'behavior' more commonly used in the literature.

5 Though extensive analysis was only performed using the 'skull' target image, individual experiments were done with the other selected Picbreeder images in *(Woolley and Stanley, 2011)* with similar results. The skull was both the most difficult in that study, and produced the most interesting images. We did not have the original CPPNs that produced the images, and instead the images were copied from screen shots, downsampled to 28x28 resolution, and used as targets.

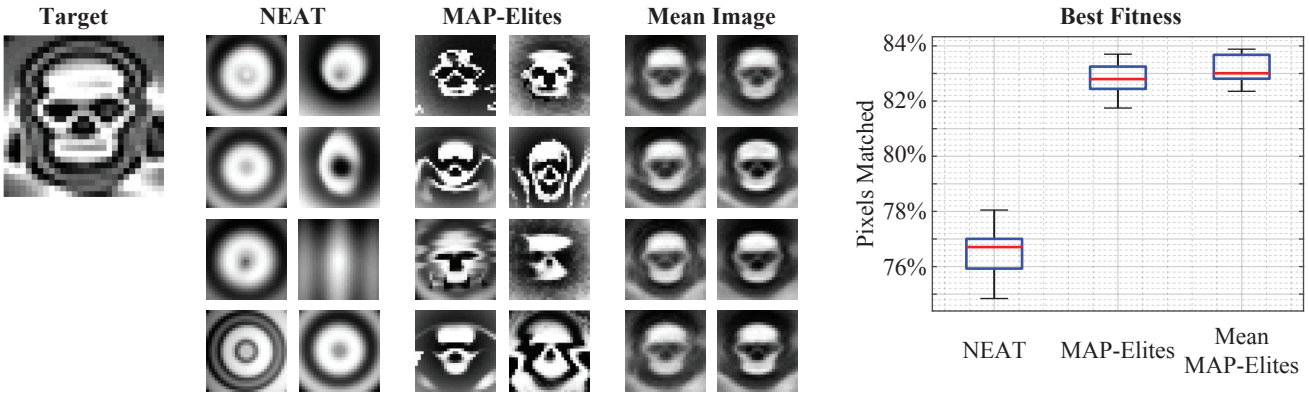

Figure 4.4: *Picbreeder Image Matching with NEAT and MAP-Elites.* From left to right: the target image, a sample of the highest performing images found by NEAT, MAP-Elites, and the mean of all images at the end of the run of MAP-Elites. Images shown are the end result of distinct runs. Far right: distribution of highest fitness found by each approach over 16 runs.

Fitness is simply the root mean square error of the difference in pixel intensities of the produced and target image.

Each algorithm was run 16 times, and the closest pixel match recorded. The distribution of the best fitness found over all runs is shown in Figure 4.4. To illustrate the meaning of this difference qualitatively we show a selection of high fitness images from each of the two approaches, each drawn from a different run.

NEAT is unable to find more than a basic circular shape, as reported in previous experiments *(Woolley and Stanley, 2011)*. Deviating from this shape decreases fitness, and these solutions are less likely to continue to the next generation. In contrast, MAP-Elites collects a variety of shapes, so possible stepping stones are kept. The difference in the final results are stark. Compared to the formless blobs produced by NEAT, MAP-Elites consistently produces a variety of skull images.

QD produces varied solutions to a problem, and in this diversity differing aspects of the problem are solved. This variety can be combined to powerful effect: if the mean pixel value of all elites in the map is taken this "mean image" is a closer match then the best single image, and much more recognizable to the human eye.

To understand if this improvement in performance is the result of collecting stepping stones we have to look not only at the final images, but the history of how they were produced. By examining the lineage of a high performing individual from a MAP-Elites run we find evidence for an optimization path which winds through poor performing stepping stones. Figure 4.5 shows the fitness of each ancestor of the final performing solution, with the corresponding phenotype of the ancestor shown at selected points.

We see that the path to high performing solutions traverse multiple low fitness stepping stones. The first image shows that optimization

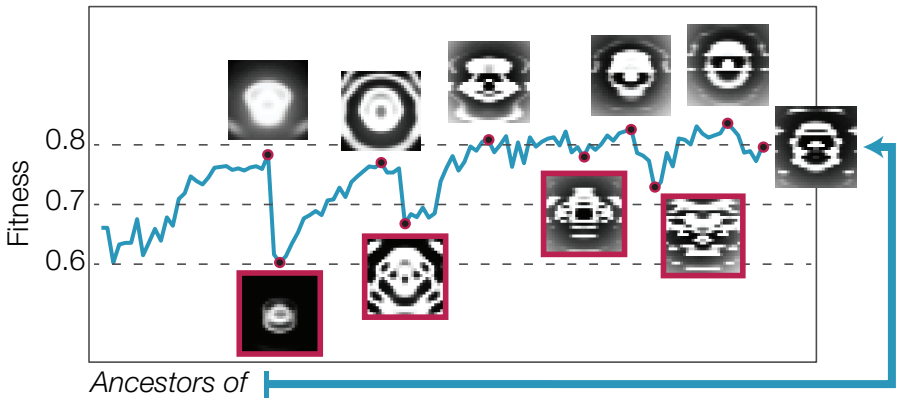

Figure 4.5: *Fitness History of a High Performing Individual*
The fitness of every ancestor of a high performing solution. Selected phenotypes are shown to illustrate peaks and valleys of fitness.

can quickly reach the kind of blob found by NEAT through iterative fitness improvements. It can be expected that this high performing dead end would have dominated a population in the NEAT algorithm, but that its strange child – with a fitness worse than even it's randomly initialized ancestors – would have been swiftly culled. But this low performer likely filled an empty niche in MAP-Elites, and so became a source of new individuals which would eventually overtake its high performing parent.

This is the story this graph repeats, over and again. Every decrease in fitness is a move into a different niche – a move which saves a child from having to compete with its stronger parent. Within a niche this individual can improve its performance, so long as it maintains whatever characteristics landed it in that niche in the first place. It is this diversion of optimization into varied niches that allows the cultivation of the poor performing individuals which are later revealed to be critical stepping stones in the development of the highest performers.

Ideally, stepping stones are not only bumps in the road toward the global optima, but jumping off points for a range of adaptations. Though the niches in MAP-Elites protect individuals from global competition, when a new innovation arises in the population, if broadly useful it can quickly spread to the other niches. The individual whose lineage we traced in Figure 4.5 was part of a large group of solutions that occupied the high performing regions of the final map. This family tree of individuals and their ancestors is shown in Figure 4.6.

A recent common ancestor of all of these high performing solutions (in red) is *not* a particularly good solution – in fact, as seen in Figure 4.5 it represents a steep decline in the fitness from its ancestors. Nevertheless, the progeny of this poor performing stepping stone evolved into a large variety of high fitness solutions which took over the highest performing niches in only a handful of mutations.

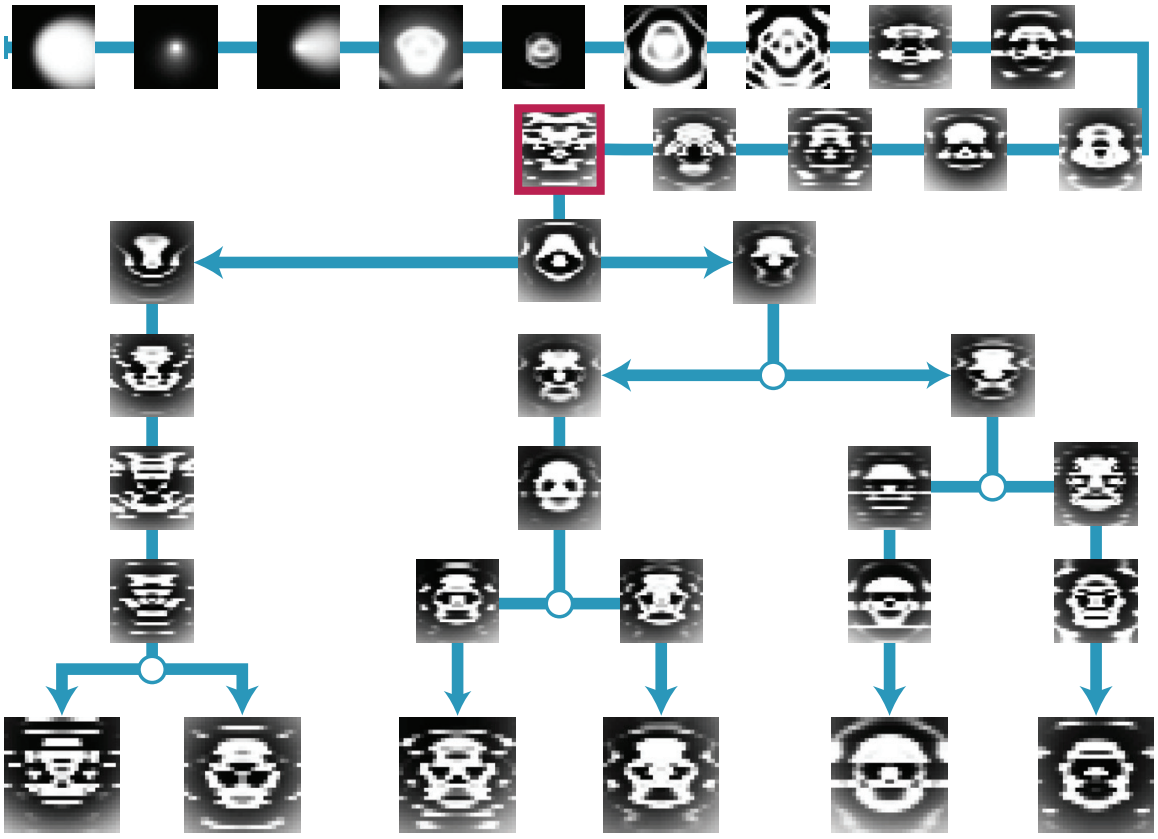

Figure 4.6: *Family Tree of High Performing Individuals*
The genealogy of a group of high performing individuals with a common, poor performing, ancestor (in red). Each link below this common ancestor represents a single mutation, links before the common ancestor are only an abbreviated selection.

4.5 DISCUSSION

In this picture synthesis experiment, a QD algorithm (here, MAP-Elites) clearly outperforms objective-based search, which suggests that QD algorithms are be better at generating stepping stones than objective-based search. Nevertheless, the images produced by MAP-Elites are not a perfect match with the target image.

Qualitative analysis of the ancestry of individuals confirmed intuitions about the winding path taken by top performers — and why these solutions are difficult for purely objective-based search algorithms to reach. Here we only examined the lineage of only a few selected high performers, an analysis of population-wide genealogy over time could be even more enlightening. Such quantitative analysis of the population dynamics could shed light on to what degree exaptation versus more straight-forward optimization is responsible for progress in the population as a whole. Such insights could lead to new perspectives on existing algorithms and to new approaches that further encourage and leverage exaptation.

Generalization of any analysis of algorithm performance in QD algorithms is complicated by the hand-designed nature of diversity metrics. The entire character of an optimization problem is altered by the diversity metric — and how it combines with the representation and objective function. Progress towards generic or learned diversity metrics would make comparisons across domains more meaningful, and so allow us to discern broader principles in QD. Learned metrics like those proposed in *(Cully, 2019)*, *(Liapis et al., 2013)*, or the familiarity measure proposed here are a step in the direction, but properly judging their abilities on test problems depends on a better understanding on a firmer understanding of the characteristics of desirable optimization behavior in QD.

Intriguingly, in every run the mean of all found solutions in the map produced by MAP-Elites was greater than the performance of any individual solution. This could be specific to the problem: judging distance per pixel gives us a fitness function where every pixel is independent of all others. Most problems cannot be so easily decomposed: when teaching a robot to walk, the first step determines the proper movements for the last. The difficulty of this domain rests in the CPPN representation, if we could manipulate one pixel at a time it would trivial. That we can learn, on average, the correct pixel value is likely a way of sidestepping difficulties in the representation.

Regardless, this still points to the potential usefulness of the body of solutions produced by QD taken as a whole. A corpus of varied and high performing solutions could be used as guidance for optimization, or as a the raw material for generative models. Data generation and representation exploration paradigms inspired by these observations are further explored in Chapter 6.

The path to truly interesting artifacts is rarely a straight line. We know from observing the course of nature and society that innovation often come from functional adaptations of existing creations: feathers began as insulation and microwave ovens as radar magnetrons. Even seemingly useless knowledge can eventually find use – curiosities such as logic machines that prove theorems or the nature of the atomic nuclei can turn into computers and nuclear power. Unlocking the power of innovation requires a class of algorithm, like QD, which can collect new knowledge without immediate benefit, adapt existing capabilities to new purpose, and make the occasional leap to entirely novel approaches.

5

SURROGATE-ASSISTED ILLUMINATION (SAIL)

Some figures, text, and ideas in this section are based on the following works which have either been previously published or are in review:

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2017a). "Aerodynamic design exploration through surrogate-assisted illumination." In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, p. 3330.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2017b). "Data-efficient exploration, optimization, and modeling of diverse designs through surrogate-assisted illumination." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 99–106.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2018). "Data-efficient design exploration through surrogate-assisted illumination." In: *Evolutionary computation* 26.3, pp. 381–410.

5.1 INTRODUCTION

Design optimization techniques are often thought of by their creators as the final step in the design process. Imagining that their techniques will be used to push the limits of performance, they judge success by the ability of an algorithm to refine a design to its most optimal form *(Thompson, 1996; Obayashi and Tsukahara, 1997; Renner and Ekárt, 2003; Hornby et al., 2011)*, with the ultimate goal of outperforming the best engineers.

If, however, the goal of these algorithms is to support designers in discovering the best possible designs, the emphasis on optimal performance may be misplaced. In an interview study by Autodesk *(Bradner et al., 2014a)*, it was found that the most common use of computational design tools was *not* at the end of the design process to hone existing designs. Instead the engineers, designers, and architects interviewed reported that they more commonly used optimization tools at the beginning of the design process to explore the space of possible designs. By using optimization tools to produce a range of design alternatives, designers explore differing design concepts, allowing them to examine the trade-offs each alternative represents. The designs generated are a consequence of the problem definition, solution constraints, and the assumptions inherent in the design's representation. Once these factors are reconsidered and adjusted, the optimization algorithm is run again to generate new designs, and the process is repeated.

Designers use this generative design cycle to explore and describe complex design spaces. Using high-performing solutions as concrete way points, they develop a vocabulary to better describe and understand design spaces that are often rendered opaque by their complexity. Armed with this understanding, designers work within these discovered design regions, refining designs in light of considerations which are difficult to formalize, including intangibles such as aesthetics.

The most commonly used method of producing this variety of high performing designs is multi-objective optimization *(Deb, 2003; Deb and Srinivasan, 2006)*. Instead of searching for a single solution multi-objective optimization produces a Pareto front of non-dominated solutions and, when the objectives are in conflict, each design represents a trade-off between the objectives *(Deb, 2003)*. During the explorative process, however, interest for designers often lies not only in the optimization of all objectives, but in the effect of different design features on performance. Exploration with multi-objective approaches becomes particularly problematic when the objectives are *not* in conflict, and only a few variations of a dominant design theme results. In addition, focusing on pure optimization may lead to solutions which "overfit" the objectives, whereas a designer could use expert knowledge to recognize that a sub-optimal design – according to the objectives – is actually better suited for the application.

To probe the search space for interesting designs and design principles, new algorithms created specifically for exploration, known as quality-diversity algorithms, could be applied *(Lehman and Stanley, 2011a; Lehman and Stanley, 2011b; Mouret and Clune, 2015; Pugh et al., 2016)*. One such algorithm, MAP-Elites *(Mouret and Clune, 2015)*, explicitly explores the relationship between user-defined features and performance to produce low-dimensional "maps": given features deemed interesting or important, such as weight or structural strength, MAP-Elites produces a large set of high-performing solutions which span the possible variations of those features. This *illumination* process reveals the performance potential of the features in various combinations.

While MAP-Elites is effective at finding diverse high-performing solutions, the search process requires a tremendous number of evaluations. The illumination process which produced the repertoire of hexapod controllers in *(Cully et al., 2015)*, for example, required twenty million evaluations. In applications such as structural optimization or fluid dynamics, where a single evaluation can take hours, this is simply infeasible.

In computationally expensive problems it is common to use surrogate models, that is, approximate models of the objective function based on previously evaluated solutions *(Jin, 2005; Forrester and Keane, 2009; Preen and Bull, 2016; Shahriari et al., 2016)*. These models are refined through iterative evaluation of new solutions based on an *acquisition function*, which balances exploitation and exploration to

improve accuracy in high-fitness regions. By substituting expensive objective functions with these computationally efficient approximations, the optimization process can be greatly accelerated.

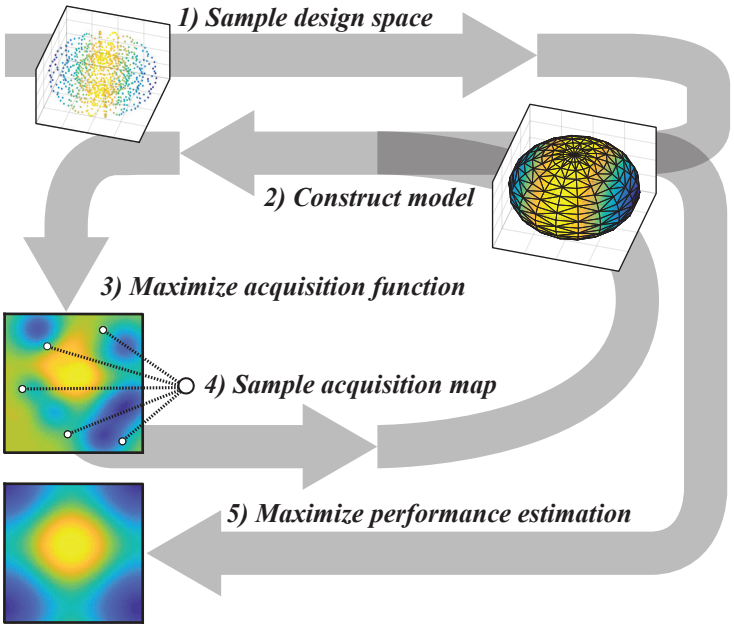

Figure 5.1: *Surrogate-Assisted Illumination (SAIL)*
1) Sample design space to produce initial solutions. *2)* Construct model of objective function based on samples. *3)* Maximize the acquisition function, balancing exploitation and exploration, in every bin of the feature space with MAP-Elites, producing an *acquisition map*. *4)* Draw samples from the *acquisition map* to test on the objective function. Repeat steps 2-4 as computational budget allows. *5)* Maximize fitness, as predicted by the resulting model, to produce a *prediction map* with MAP-Elites.

The key insight leveraged in this work is that surrogate models can not only lessen the burden of *expensive* evaluations, but also *numerous* evaluations. Incorporating surrogate-assistance techniques into the evaluation-heavy illumination process has the potential to make MAP-Elites efficient enough for use in computationally expensive problems, and accelerate it in even inexpensive problems by orders of magnitude.

In this article, we present the Surrogate-Assisted Illumination (SAIL) algorithm to improve the efficiency, and so expand the applicability, of MAP-Elites. The value of integrating surrogate models into illumination relies on reducing computational cost while maintaining MAP-Elites' original capabilities, resulting in an algorithm that is:

- *Divergent* - Solutions vary across a user-defined continuum;
- *Accurate* - Predicts objective function in high-performing regions;
- *High-Performing* - Produces near-optimal solutions;
- *Data-Efficient* - Performs even when functions evaluations are expensive or limited.

SAIL is a Bayesian optimization corollary for illumination, leveraging modeling techniques to accelerate MAP-Elites. In general terms this interaction between illumination and modeling proceeds as follows (Figure 5.1): a surrogate model is constructed based on a set of initial solutions, MAP-Elites is used to produce solutions that maximize the acquisition function in every region of feature space, yielding an *acquisition map*, new samples are then drawn from the acquisition map and evaluated, and these additional observations are used to improve the model. This acquisition process is repeated to produce increasingly accurate models of the high-fitness regions of the feature space. Performance predictions of the model can then be used by MAP-Elites in place of the objective function to produce a *prediction map* of estimated optimal designs in all feature regions.

5.2 APPROACH

We introduce Surrogate-Assisted Illumination (SAIL) as a method analogous to Bayesian optimization (BO), but targeting the purposes of quality-diversity. While the goal of BO is to model the behavior of the objective function in the highest performing region, illumination expands this requirement. The goal of illumination is not to find a single optimum, but optima with every combination of features: not a single point, but a slice with one dimension per feature. The goal of SAIL is then to predict the behavior of the objective function in the high-performing regions of this feature slice. Producing models which can perform these predictions requires finding high-utility solutions which cover these feature dimensions.

These high-utility solutions are found by MAP-Elites. Maximizing the acquisition function in every feature bin produces an *acquisition map*, a set of high-utility solutions that span the feature dimensions. As utility is derived using only the Gaussian process model, an acquisition map can be produced with minimal computation. To place solutions in the map requires that features as well as fitness be calculated. In design cases, features can typically be cheaply derived without evaluation. In some domains, such as evolutionary robotics, it may not be possible to cheaply extract features, as they are often behaviors exhibited during evaluation. In such cases, features as well as fitness would need to be approximated.

To define utility SAIL uses the UCB acquisition function (see Section 2.2: *Surrogate-Assisted and Bayesian Optimization*) rather than other common acquisition functions such as Expected Improvement (EI) and Probability of Improvement (PI) *(Brochu et al., 2010; Calandra et al., 2016)*. These acquisition functions rely on comparisons to the current optimum, while UCB is based only on the confidence of the underlying model. As SAIL solves numerous localized problems in parallel, it requires an acquisition function independent of the global optimum.

If compared globally, solutions in less optimal regions of the map would have a vanishingly small probability of improving on the global optimum, and because many bins will not contain existing evaluated solutions, it will not always be possible to perform local comparisons against optima within a bin.

The acquisition map acts as a collection of candidate solutions for evaluation. As our goal is to create a model that is accurate on the entire feature slice, it is necessary to evaluate high-utility solutions which cover the slice in its entirety, not only at the highest utility point. Just as a Sobol sequence can be used to evenly sample the parameter space, it can also be used to evenly sample the acquisition map. Feature coordinates are drawn from a Sobol sequence, and the elite contained in the corresponding bin of the acquisition map is selected for evaluation. Any number of new solutions from the acquisition map can be selected by drawing the next sets of coordinates from the Sobol sequence. The selected solutions are then evaluated on the objective function, and the resulting input/output pairs added to the model. The illumination process is then repeated with this more accurate model, creating a new acquisition map.

The model refined through this iterative illumination and modeling process can be used to produce a *prediction map*. By adjusting the acquisition function so that it does not reward uncertainty, MAP-Elites will produce a map which includes the model's best guess of the optimal design in each bin. This map provides an informed estimate of the relationship between features and performance, and as only the surrogate model is required, this prediction map can be produced with minimal computation.

The SAIL algorithm is more precisely defined in Algorithm 2. An initial set of individuals is created using a Sobol sequence *(Niederreiter, 1988)* to ensure our model is based on solutions which evenly cover the *parameter* space. These individuals and their performance form the set of observations which are used to construct the initial GP model. An empty acquisition map is then created and filled with the individuals from the observation set, along with their utility as judged by the acquisition function. These individuals are taken as the starting population for MAP-Elites (see Algorithm 1) which then illuminates the map as described in Section 2.1.2: an elite is selected and mutated to produce a child, it is assigned a bin based on its features, and then competes for the bin if it is not occupied. This illumination process repeats for a number of iterations, and results in an acquisition map of elite individuals who maximize the acquisition function in their bin. A *prediction map* can then be produced by maximizing only the predicted mean performance in each bin with MAP-Elites, using the evaluated individuals as a starting population.

Algorithm 2 Surrogate-Assisted Illumination (SAIL)

1) Create Initial Gaussian Process Model
$\mathcal{X} \leftarrow Sobol_{1:G}$ ▷ *Initialize with G solutions drawn from Sobol sequence*
$\mathcal{P} \leftarrow PE(\mathcal{X})$ ▷ *Precisely Evaluate (PE) solutions to get performance*
$\mathcal{GP} \leftarrow Gaussian\_process\_model(\mathcal{X}, \mathcal{P})$ ▷ *Train GP model*

2) Produce Acquisition Map
while *precise evaluation budget not exhausted* **do**
  $acq() \leftarrow UCB(\mathcal{GP}(x))$ ▷ *Use UCB of prediction as fitness*
  $(\mathcal{X}_{acq}, \mathcal{P}_{acq}) =$ MAP-ELITES$(acq(), \mathcal{X})$ ▷ *Create acquisition map*
  $\mathbf{x} \leftarrow \mathcal{X}_{acq}(Sobol_{iter})$ ▷ *Select solutions from acquisition map for PE*
  $\mathcal{X} \leftarrow \mathcal{X} \cup \mathbf{x}$ ▷ *Add evaluated solutions to observation set*
  $\mathcal{P} \leftarrow \mathcal{P} \cup PE(\mathbf{x})$ ▷ *Along with their performance*
  $\mathcal{GP} \leftarrow Gaussian\_process\_model(\mathcal{X}, \mathcal{P})$ ▷ *Train GP model*
end while

3) Produce Prediction Map
$prediction() \leftarrow mean(\mathcal{GP}(x))$ ▷ *Use mean of prediction as fitness*
$(\mathcal{X}_{pred}, \mathcal{P}_{pred}) =$ MAP-ELITES$(prediction(), \mathcal{X})$ ▷ *Prediction map*

5.3 RESULTS

5.3.1 *Benchmark Case: 2D Airfoil*

We first demonstrate the capabilities of SAIL on a classic design problem of airfoil shape optimization. SAIL is designed with computationally intensive domains in mind, but this relatively inexpensive domain allows for deeper analysis and evaluation of the algorithm's performance. The lower computational cost of evaluations allows for the exhaustive experimentation that allows us to confidently verify that in this case: (1) the designs found by SAIL are near optimal in all regions of the feature space, (2) the models created by sampling with SAIL are an order of magnitude more accurate in high-fitness regions than those produced by uniform sampling of the parameter space, and that (3) SAIL produces high-performing solutions in every feature region with the same computational budget required by a standard black-box optimizer to find a single design. This level of analysis is only possible in this less expensive cases, where extensive optimization and search can be performed.

SETUP We give SAIL the task of producing a set of 2D airfoils with minimal drag which still maintain the lift and area of a base airfoil. Quadratically increasing penalties are introduced into the fitness function to ensure that these constraints are followed with little deviation. The high performing RAE2822 airfoil was chosen as

our base, with foils evaluated at an angle of attack of 2.7°, Mach 0.5, and a Reynolds number of 10^6.

While the area of the foil can be directly measured without aerodynamic tests, the drag[1] (C_D) and lift (C_L) must both be approximated. The UCB of the drag prediction is taken as the drag component of our fitness function:

$$drag(x)' = \mu_{\text{drag}}(x) + \kappa\sigma_{\text{drag}}(x) \tag{5.1}$$

For lift, as we do not reward individuals for high lift, only penalize them from deviating from a target, we treat the lift prediction problem not as one of regression, but instead as one of classification. Two classes of solution are considered, those which violate the constraint and those which do not. Individuals are penalized based on the probability that they will have a lower lift than our base foil and so belong to the invalid class. This probability is calculated according to the mean and variance supplied by our GP model:

$$penalty_{lift}(x)' = 1 - P(C_L(x) < lift_{base}) \tag{5.2}$$

Combining these predictions gives us a utility measure for a solution x, used as a fitness function in SAIL, formally defined as:

$$fitness(x) = drag(x) \times penalty_{lift}(x) \times penalty_{area}(x) \tag{5.3}$$

where $drag(x) = -log(C_D(x))$

$$penalty_{lift}(x) = \begin{cases} \left(\frac{C_L(x)}{lift_{base}}\right)^2, & \text{if } C_L(x) < lift_{base} \\ 1, & \text{otherwise} \end{cases} \tag{5.4}$$

$$penalty_{area}(x) = \left(1 - \frac{|area - area_{base}|}{area_{base}}\right)^7 \tag{5.5}$$

Airfoils are encoded using the airfoil-specific Parametric Sections (PARSEC) parameterization *(Sobieczky, 1999)*. PARSEC allows the direct parameterization of features such as the radius of the leading edge or the curvature of the upper surface, and so allows a large variety of designs to be expressed with a small number of parameters. The ten parameters used to define an airfoil in these experiments are shown in Figure 5.2.

In these experiments two PARSEC descriptors, the height of the airfoil (Zup) and the location along the airfoil of that highest point (Xup) are used as features of variation for exploring the design space.

1 C_D values are quite small, and are converted to log scale when training models and calculating fitness

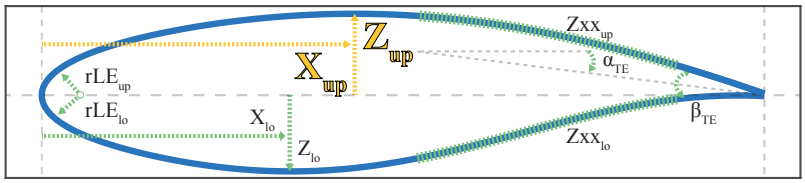

Figure 5.2: *The Ten Parameters Used to Define an Airfoil*
Airfoils are defined by a set of features using the PARSEC encoding. These features include: the leading edge radius of the upper and lower curves (rLE_{up}, rLE_{lo}), the location of highest and lowest point (Z_{up}, Z_{lo}), the location of highest and lowest point (X_{up}, X_{lo}), the curvature of the upper and lower sides (Zxx_{up}, Zxx_{lo}), and the angle and arc radius of the trailing edge (α_{TE}, β_{TE}). The dimensions of variation explored (X_{up} and Z_{up}) are colored in gold.

Selecting parameter values as features of variation allows the results of SAIL to be compared with those of standard optimization algorithms designed to find a single solution. These algorithms have no concept of 'features', but by restricting the search within given parameters ranges, the regions of feature space explored can also be restricted.

SAIL is compared to (1) the original MAP-Elites algorithm, (2) the black-box optimization algorithm Covariance Matrix Adaptation Evolution Strategy (CMA-ES) *(Hansen and Ostermeier, 2001)*, and (3) a surrogate-assisted variant of CMA-ES (SA-CMA-ES). As the primary objective of SAIL is to achieve results with data-efficiency, the unit of comparison used is the number of function calls, or precise evaluations (PE) required to achieve a certain result. SAIL is given a total computational budget of 1000PE. 50PE is used to evaluate a set of designs produced through parameter sampling, which forms the basis of the initial GP model. At each acquisition iteration (Algorithm 2: lines 6-13) 10 additional individuals are chosen from the acquisition map, evaluated, and incorporated into the GP model. This result is compared with the designs produced by MAP-Elites when it is given a computational budget of 10^5PE.

Optimality of designs is judged in comparison to the black-box optimization algorithm CMA-ES. By restricting the allowed parameter ranges, the search is confined to a single feature bin, and CMA-ES is given a budget of 1000PE to produce a design in each bin. To control for the ease of modeling the problem we also employ a surrogate-assisted variant of CMA-ES, denoted SA-CMA-ES, to solve each of these subproblems. Within a single bin 25 initial solutions are sampled from the parameter space and used to produce an initial GP model. CMA-ES is then used to maximize the same UCB-based acquisition function used by SAIL, described above. The single found optimum is evaluated and the new individual incorporated into the GP model. This process is repeated for a total of 100PE.

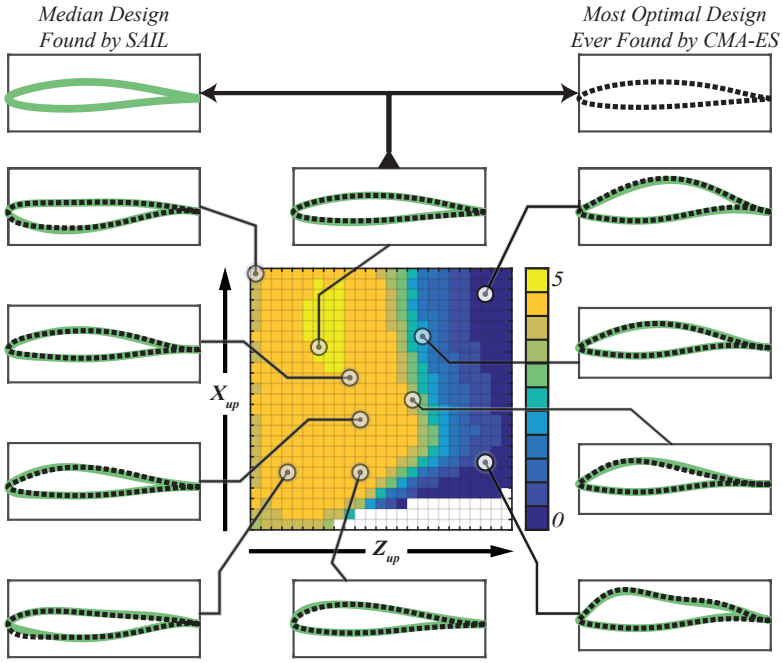

Figure 5.3: *Design Space Overview Produced with Sail*
Prediction map produced by SAIL after 1000PE, bins colored by fitness. Fitness is dominated by the drag component: $-log(C_D)$, e.g. a fitness of 5 is equivalent to a C_D of 10^{-5}. The border contains designs in selected bins: median-performing designs found by SAIL in green, best designs ever found in *all* CMA-ES runs in black.

The 2-dimensions of the feature space (Z_{up} and X_{up}) are each divided into 25 partitions, for a total of 625 bins. Bins with the highest point at the trailing edge of the wing (high Z_{up} and low X_{up}) could not be found due to geometric constraints inherent to the PARSEC representation *(Padulo et al., 2009)*. Only the remaining 577 bins were considered in statistical comparisons. Each approach was replicated 20 times with different random seeds.[2] Unless otherwise mentioned, all values given are medians over all replicates.

EFFICIENT DESIGN SPACE ILLUMINATION In Figure 5.3 the median-performing prediction map produced by SAIL is shown. When each bin is color coded by the fitness of the design within, it provides an intuitive overview of the relationship between airfoil features and performance: the height of the airfoil (Zup) has the most influence, with

2 One replicate, including precise evaluation of all designs in the intermediate prediction maps used for data gathering, with 8 cores of a Intel Xeon 2.6GHz processor required: SA-CMA-ES:32h, CMA-ES:80h, SAIL:12h, MAP-Elites:14h Where possible, standard implementations of algorithms were used, including the CMA-ES as published by Hansen *(Hansen, 2008)*, the Gaussian processes for machine learning toolbox by Rasmussen *(Rasmussen and Nickisch, 2016)*, and the XFoil airfoil solver of Drela *(Drela, 2013)*.

the location of the highest point (Xup) having a more nuanced effect, largely dependent on the height of the airfoil. Around the border the median-performing design found by SAIL in a bin is shown in green, along with the most optimal design ever found by CMA-ES in black. The designs are very similar, though found by SAIL in a single 1000 evaluation run and by CMA-ES over many runs using more than 11.5 million evaluations[3].

MODEL ACCURACY To evaluate the accuracy of the produced models, after the final sample was collected a prediction map was produced. Each design in the prediction map was then precisely evaluated and the true C_D and C_L compared to the prediction of the model. The median results are shown in Figure 5.4. On the majority of samples the surrogate is reliably accurate, with more than 90% of drag ($log(C_D)$) predictions and more than 80% of lift (C_L) predictions within 5% of their true value. Drag errors are clustered in the same region of design space, a region where the flow simulator was less likely to converge and produce valid results.

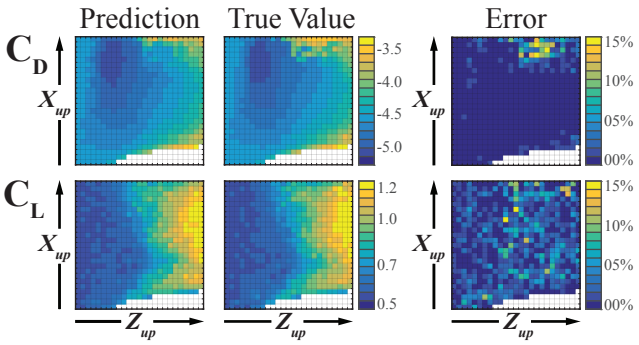

Figure 5.4: *Drag and Lift Predictions Per Bin*
Predicted and true values of drag ($log(C_D)$) and lift (C_L) for designs in each bin after 1000PE. Median values over 20 replicates.

The purpose of our models is to estimate performance in the optimal regions of the search space. To test their accuracy in this high fitness slice, we measure their ability to predict the performance of the best designs found by CMA-ES in each bin. We compare models built using a naive sampling of the parameter space with a Sobol sequence *(Niederreiter, 1988)* to sampling done using acquisition maps produced by SAIL. These acquisition maps are produced by maximizing three different acquisition functions: the mean or variance alone, and the UCB, a combination of the mean and variance (see Section: 5.2). The accuracy of each model's drag prediction on the best design in every bin is then measured at various stages of the sampling process (Figure 5.5).

3 20 replicates × 577 bins × 1000 precise evaluations

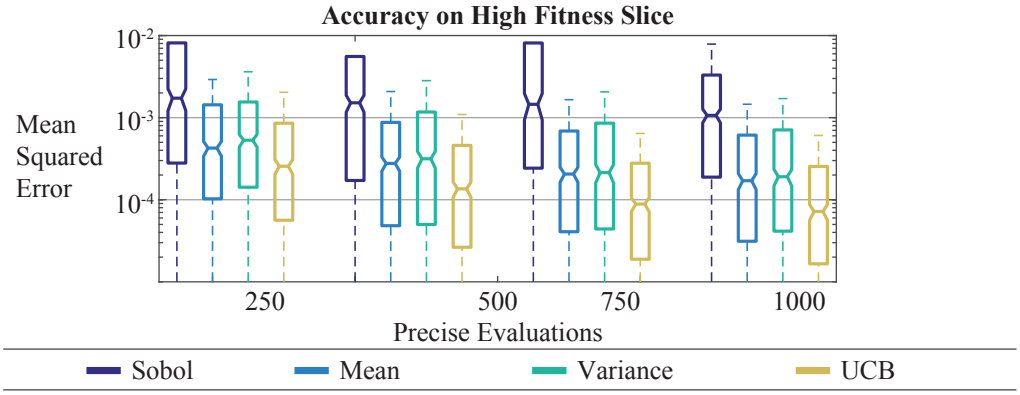

Figure 5.5: *UCB Acquisition Outperforms Acquisition for Performance or Variance Alone*
Mean squared error (log scale) of drag prediction on optimal designs found by CMA-ES. Models were constructed using designs: 1) sampled from parameter space using a Sobol sequence, or selected from acquisition maps where individuals were optimized to maximize their predicted 2) mean, 3) variance, or 4) Upper Confidence Bound (UCB). Though all acquisition functions produced more accurate models than uniform parameter sampling, UCB improved by a full order of magnitude on high-fitness solutions.

By concentrating the sampling process on either high performing solutions or on reducing overall uncertainty we are able to produce better performing models than evenly sampling the parameter space. When both uncertainty and performance are considered, that is, when using the UCB, SAIL produces models that are an order of magnitude more accurate than uniform sampling.

PERFORMANCE The optimality of the designs is limited by the accuracy of our models, but the accuracy of our models does not ensure the optimality of the produced designs. We could be modeling an average performing region very well, but this would not truly reflect the fitness potential of the design space. And though our goal is not to directly compete with algorithms designed to find one optimal solution, to accurately portray the design space it is critical that the solutions found are representative of the best designs in their region.

We compared the designs found in each bin by SAIL after 1000PE to the best design found by CMA-ES after all 20 runs for 1000PE in each valid bin ($\approx$ 11.5 million PE in total). Figure 5.6 shows the median values of the prediction map, the true performance of those median designs, the optimal performance found after 20 runs of CMA-ES, and the fitness difference between these optimal values and those found by SAIL. The fitness potential of the feature space is well illuminated, with found designs performing within 5% of optimal in nearly half of bins, and the general relationship between features and performance accurately portrayed.

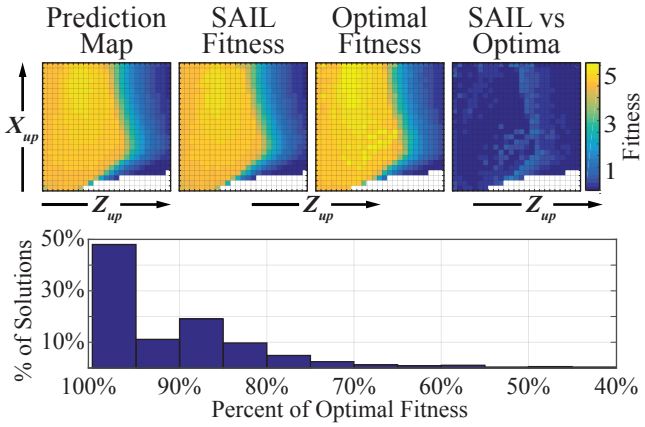

Figure 5.6: *Performance of Designs Found By SAIL*
Top: Median predicted and true performance of designs found by SAIL with a budget of 1000PE compared with performance of optimal designs found by 20 runs of CMA-ES per bin (≈11.5 **million** PE)
Bottom: Optimality of SAIL designs per bin.

SAIL is designed as a data-efficient variant of MAP-Elites, and so its ability to find optimal solutions must be examined in this context. Beyond MAP-Elites there are few feature space exploration algorithms, and so for additional comparison we look to the standard black-box optimizer CMA-ES. As CMA-ES is not designed for use across a multitude of subproblems, the number of evaluations required to produce an optimized feature map is highly dependent on the number of bins in the map.

More informative than comparing performance across the entire map is to compare SAIL to the performance of CMA-ES within a single bin. While it is not expected that SAIL will compete with an algorithm like CMA-ES in finding a single solution, it allows us to put SAIL's optimization performance in context. As optimization progress may vary according to the bin, we take the single bin performance as the map performance divided by the number of bins. We include a surrogate-assisted variant of CMA-ES (SA-CMA-ES) (see Section 5.3.1) to control for the ease of modeling the problem. Figure 5.7 illustrates the comparison of performance per precise evaluation.

With the same evaluation budget required by CMA-ES to find a near optimal solution in a *single* bin, SAIL finds designs of comparable performance in *every* bin. Comparisons between CMA-ES and MAP-Elites and their surrogate-assisted variants SA-CMA-ES and SAIL reveal that the gains from surrogate-assisted optimization are even greater for MAP-Elites than for the traditional optimizer. While surrogate-assistance improves the efficiency of CMA-ES by an order of magnitude, even when MAP-Elites is given an evaluation budget two orders of magnitude greater than SAIL it cannot produce solutions of similar performance.

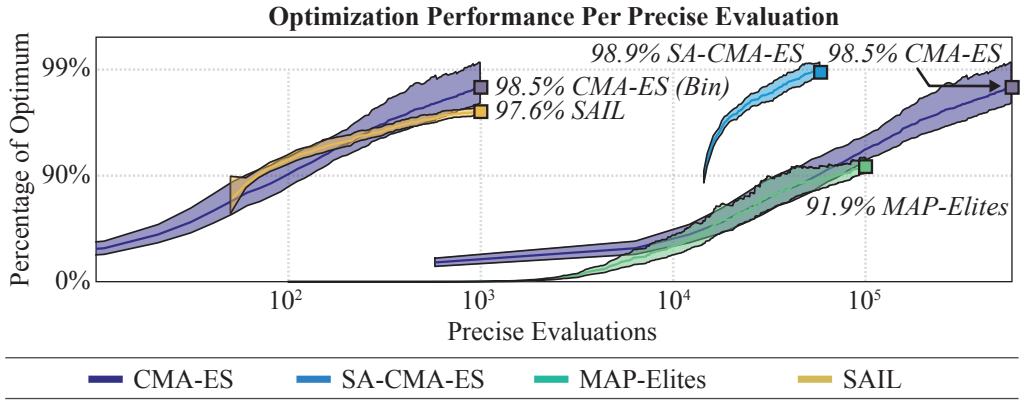

Figure 5.7: *Data-efficiency of Optimization Over the Entire Design Space*
Computational efficiency (log scale) of CMA-ES, SA-CMA-ES, SAIL, and MAP-Elites measured in precise evaluations. Median performance across all bins as percentage of optimum (log scale). For SAIL and SA-CMA-ES, performance is only recorded after initial sampling. As the optimum is the highest performance over *all* runs, the median CMA-ES run does not reach this value. *Bin*: median progress towards optimum of every bin. Bounds indicate one standard deviation over 20 replicates.

5.3.2 *Illumination of 3D Aerodynamics Design Spaces*

To further explore the capabilities of SAIL, we choose a more demanding task: the optimization of aerodynamic shells for recumbent bicycles, or velomobiles. These streamlined vehicles hold human-powered speed (144.17 km/h, *(International Human Powered Vehicle Association, 2017)*) and distance (680 km in 12 hours, *(World Human Powered Vehicle Association, 2017)*) records due to their highly tuned aerodynamics.

Due to constraints such as rider movement and comfort, three-wheeled designs built for distance races often have non-intuitive shapes (Figure 5.8). These high-performing but odd designs suggest a domain rich in interesting design concepts. To encode the design of the shell a parameterized encoding could be created, as was done with PARSEC for airfoils, or a more general-purpose solution could be applied, by deforming an existing design. To isolate the capabilities of SAIL from the capabilities of a given representation we examine and compare the result for both encoding approaches.

While the PARSEC representation is composed of specifically engineered features that are known to be important for the performance of airfoils, most design domains have not been as intensely researched. It is more useful to be able to define a realistic set of features which do not directly correspond to the parameter values of the representation. Here we explore the curvature and volume of the designs, neither of which directly correspond to parameter values of our encodings.

Figure 5.8: *Record Setting Velomobiles Have Non-intuitive Shapes.* The record setting Milan velomobile eschews the intuitive bullet shape in favor of a form which decreases frontal area while still providing space for the riders feet and knees. Bumps on the sides guide and direct the flow that splits over this uneven surface, reduces the effects of side winds, and adds rigidity to the thin carbon fiber shell.

By illuminating the design space according to features which are unaligned with the parameters of the encodings we demonstrate that SAIL is doing more than parallel search in a partitioned search space. SAIL produces designs that vary across a spectrum in a low-dimensional feature space, illuminating the relationship between features and performance in a way largely independent of the method by which the designs are encoded.

PARAMETERIZED DESIGN To produced a parameterized encoding for the smooth form of the velomobile shell, we use a series of 2D airfoil-like curves defined with the PARSEC encoding *(Sobieczky, 1999)* (*see Section 5.3.1: 2D Airfoil Optimization*).

Viewed from the top (XY), the vehicle is defined as a symmetrical airfoil (Figure 5.9, *Top*). The side profile (XZ) is defined by three curves: one curve, unchanged in every design, which defines the bottom of the vehicle, and two parameterized curves for the top, one in the center of the vehicle (Figure 5.9, *Mid*) and one which forms a *Ridge* for the knees (Figure 5.9, *Ridge*). A final curve connects this ridge to a flat bottom forming the view from the front (YZ) (Figure 5.9, *Rib*). This *Rib* is defined along a unit vector, and is scaled to remain consistent with the curves defined by the *Top* and *Ridge* curves along 32 sections. Ridges follow the same curve as the *Top* curve of the body. These 2D curves and how they are composed into a 3D shape is illustrated in Figure 5.9. All curves are connected to each other by splines. We only permit a subset of the curve parameters to be modified, limiting the number of representation parameters to 16. These degrees of freedom are enumerated in Table 5.1.

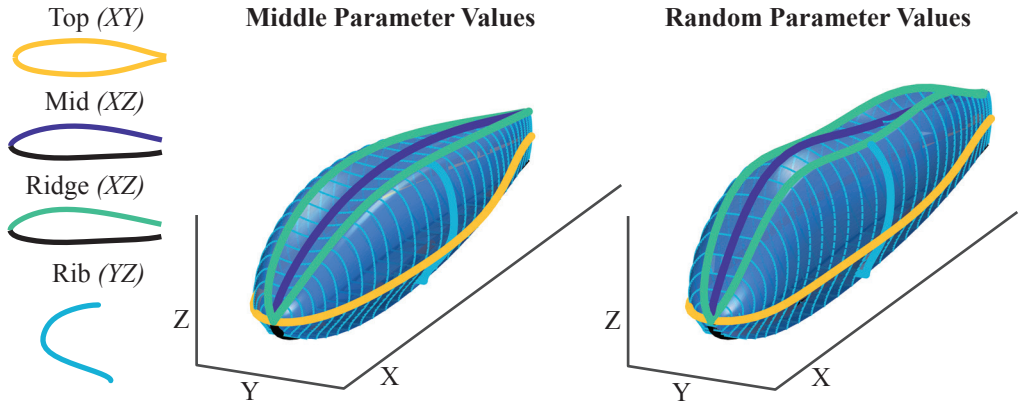

Figure 5.9: *Velomobile Design in 3D Produced from a Set of 2D PARSEC Curves* PARSEC curves are defined in two dimensions (*left*) and connected by splines. Top (XY) profile in yellow, side profiles (XZ) for the middle profile in purple, with raised knee ridges in green, the front (YZ) profile rib in cyan. This rib connects the side profile to a fixed base, in black, and is scaled to fit the width of the top profile at every section. The resulting 3D designs are shown when parameter values are set to the middle of the range (*middle*) and when random parameters are chosen (*right*).

Curve	PARSEC Parameter			
Top (XY)	r_{LE}	X_{up}	Z_{up}	Zxx_{up}
Mid (XZ)	r_{LE}	X_{up}	Z_{up}	Zxx_{up}
Ridge (XZ)	r_{LE}	X_{up}	Z_{up}*	
Rib (YZ)	r_{LE}	X_{up}	Z_{up}	Zxx_{up}
	α_{TE}			

Table 5.1: *The 16 Parameters Used to Determine the Shape of a Velomobile Shell.* Each PARSEC airfoil can be described by 10 or more parameters, but as we are only producing one curve rather than an entire foil, only the parameters which describe the top side are necessary. In addition, we adjust only a limited set of parameters for a total of 16 degrees of freedom.
(*) The height of the ridge is defined in relation to the height of the middle profile.

FREE FORM DEFORMATION As an alternative to the hand-designed parameterized encoding, we employ a deformation approach to design optimization that uses free-form deformation (FFD) *(Sederberg and Parry, 1986)*. FFD is a well-established technique in computer graphics and design, including evolutionary aerodynamic design optimization *(Samareh, 1999; Menzel et al., 2005; Sieger et al., 2012)*, which, unlike parameterized representations, allows the designer to begin with a prior design, such as an existing high-performing design, and further refine it. Deformations decouple the complexity of the design from the complexity of the representation: an intricate hand designed

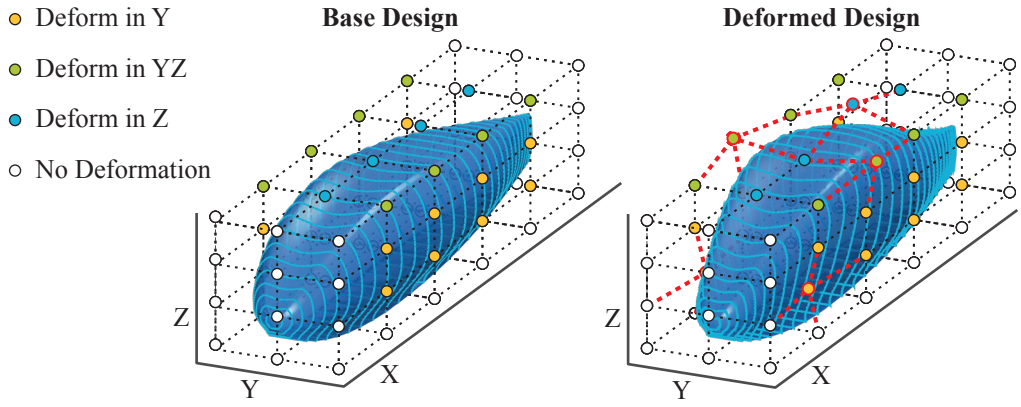

Figure 5.10: *Velomobile Design Created through Free-Form Deformation.* A control volume of evenly space points is constructed around the base velomobile shape. Active control points are shown in color: orange points deform along the Y axis, blue points along the Z axis, and green points move equally in the Y and Z axis. All points are stationary in the X dimension. Orange and green points move in tandem with those on the other side of the velomobile to enforce symmetry, resulting in a total of 16 degrees of freedom.
Left: The base design with no deformation, *Right:* A new design is created by adjusting three parameters which, once symmetry is enforced, moves two control points in Y, two control points in the YZ plane, and one control point in the Z dimension.

object can be deformed and optimized with only a few degrees of freedom, without the need to design a representation which can recreate the original design. In fluid dynamics applications meshes must be created for every design and this meshing is itself a difficult and time consuming process, which in many cases must be done by hand. However, when deformations are applied the mesh does not have to be recreated but instead likewise deformed, greatly simplifying the process, and so this onerous step can often be omitted.

To perform FFD on a design, it is first embedded into a lattice of control points. The mesh points of the design are converted into a local coordinate system based on these control points, so that any point X has (s,t,u) coordinates in lattice space

$$X(s,t,u) = X_0 + sS + tT + uU \tag{5.6}$$

where X_0 is the origin in lattice space and S,T,U dimensions that lie along the edges of the control volume. For any point inside the lattice s, t, and u are between 0 and 1. Control points are defined in a grid along the control volume as:

$$P_{\text{ijk}} = X_0 + \frac{i}{l}S + \frac{j}{m}T + \frac{k}{n}U \tag{5.7}$$

where l, m, n are the number of control points in the S, T, and U dimensions. When a control point is moved, these mesh points are also adjusted to maintain their position in relation to the control points. The influence of each control point on a point in the mesh is determined by a Bernstein polynomial blending function B. To get the deformed position in Cartesian space of a given point X_{ffd}, we first convert the point's location to (s, t, u) coordinates then compute shifts based on each control point:

$$X_{\text{ffd}} = \sum_{i=1}^{l} \sum_{j=1}^{m} \sum_{k=1}^{n} P_{\text{ijk}} B(s) B(t) B(u) \tag{5.8}$$

We design the control lattice and degrees of freedom of our FFD encoding in such a way as to keep it comparable to our parameterized representation. As a base shape we use the design produced by the parameterized encoding with every parameter at the center of its parameter range. Deformation control points can move in the positive or negative direction, and here are normalized so that at the center of the deformation range no deformation takes place. It follows then that when all parameter values of both encodings are at the center of their range, they produce identical designs. Our control lattice is constructed in an intuitive manner, surrounding the design in its entirety, with each dimension corresponding to one in Cartesian space.

Control points are placed to evenly divide the area surrounding the design into 6 segments in the X axis, 3 segments in Y, and 4 in Z (see Figure 5.10). We only actively manipulate a subset of the control points, and these we only move in a single dimension. The control points at the bottom of the velomobile are left unmoved, restricting deformations of the base, as is done in the parameterized case. The control points at the front and back of the design are left unmoved as well, keeping the start and end points of the design in line with that of the parameterized encoding. We allow the 4 center top control points to move up and down in the Z dimension, the 8 center side control points to move in and out in the Y dimension, and the 4 top corner points to move in both the Y and Z dimension, though only at the same rate. Enforcing symmetry in the Y dimension results in a total of 16 degrees of freedom, as in the parameterized case.

FEATURES Two dimensions of variation are explored: volume and curvature. While it is obvious that lower volume designs will produce less drag, it is just as obvious that a design with no volume is not optimal. A designer could determine the specific dimensions for a given configuration of machinery and rider, and then codify these as constraints for an optimization algorithm. SAIL instead produces a set of high-performing designs of varying volumes which the designer can browse. A design can then be selected which satisfies their constraints, or which could satisfy them with small adjustments.

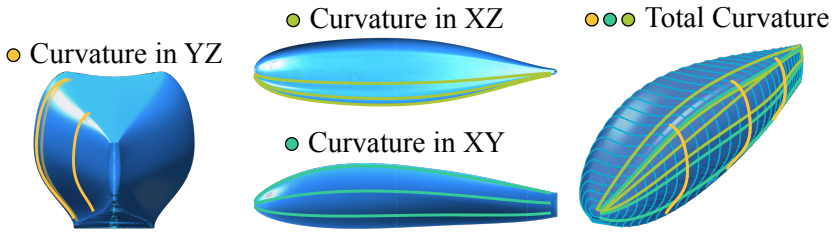

Figure 5.11: *Determination of Curvature from Selected Sections.* 3D curvature is estimated by calculating the 2D curvature of representative sections. Along the *yellow* lines we measure curvature in the YZ axis, along the *green* lines in XZ, and along the *blue* lines in XY. The total curvature is the mean curvature of all lines.

Rather than precisely measuring the three-dimensional curvature of the millions of designs produced in a single run of SAIL, for the sake of computational efficiency, we estimate the curvature based on a few fixed regions. We calculate the two-dimensional curvature along nine lines (shown in Figure 5.11): three in each of the XY, XZ, and YZ dimensions. We take the mean of these curvatures as an estimate of the curvature of the design. We calculate curvature K analytically from each pair of neighboring points along the line as $K = \frac{|x'y'' - y'x''|}{(x'^2 + y'^2)^{\frac{3}{2}}}$ where x and y are their Cartesian coordinates.

In high-performing velomobiles, curvature in the shell allows designers to minimize the effects of side winds while simultaneously reducing the frontal area and maintaining enough space to accommodate the rider's feet, knees, and the machinery of the bicycle. The addition of curvature in one part of the design may also require curvature in another, in order to effectively guide and reattach the airflow. In addition to aerodynamic concerns, curvature is introduced into designs to improve the structural integrity of the shell. The shell of the vehicle is thin, and so where the shell is flat it is also weak, weak enough to buckle and change shape at high speed or in high winds.

The ability to effectively explore features like curvature, which are not directly correlated to performance, but whose effects we are interested in, is a design exploration capability that is difficult to replicate with techniques that rely on producing variety through trade-offs, such as multi-objective optimization.

EXPERIMENTAL SETUP Properly evaluating three-dimensional designs requires a computationally intensive flow simulator, rather than the purpose built solver as was used for two-dimensional airfoils. The computational expense of using fluid dynamics simulations, however, means that evaluating every design in a prediction map at every step of the algorithm, as we did in the airfoil case, is infeasible. This computational cost prevents us from tracking the improvement of design

performance and modeling accuracy in the detailed way that was possible in the relatively inexpensive airfoil case.

Comparisons with a black-box optimization algorithm such as CMA-ES also presents some difficulty. Without a simple mapping between parameter and feature space, it is difficult to search within a single bin. The awkwardness of such a mapping aside, the computational cost alone of running CMA-ES several hundred times is prohibitively expensive. [4]

These difficulties make it infeasible to solve this problem with a traditional optimizer, so the comparisons made here are between the designs produced by SAIL using different encodings. In the 2D airfoil test case SAIL was shown to be capable of finding near-optimal designs, and we assume here that it likewise produces designs which are high-performing, even if not optimal. If different encodings produce designs with similar performance and reveal the same feature relationships this consistency will give us confidence in SAIL's performance.

We initialize SAIL with 200 samples drawn from a Sobol sequence and add 10 samples from the acquisition map at every illumination iteration, for a total of 1000 samples. We divide the feature space into a 25 X 25 acquisition map for a total of 625 bins. Fitness values are approximated using a single GP model with a squared exponential kernel with ARD which predicts drag force on the design. Though fitness values are approximated, the features of each design are derived precisely, and every design is fully constructed. All results shown are median values over 20 replicates We evaluate the fitness of the produced designs purely on aerodynamic criteria. Velomobile shells are judged by the drag force they produce when traveling at 20 m/s (72 km/h). Flow simulation is performed with the OpenFOAM Computational Fluid Dynamics toolbox *(OpenFOAM Foundation, 2017)*.

DESIGN PERFORMANCE The performance of the designs produced through SAIL (Figure 5.12) illustrates the strengths of each encoding. While the free-form deformation is able to produce higher performing solutions in the high-volume low-curvature regions of the feature space, it is not flexible enough, or its parameter ranges are too limited, to produce designs which have both very high volume and high curvature.

Comparison of performance maps reveals that the trend of high-curvature high-volume shapes performing poorly compared to low-curvature high-volume shapes is a quirk in the deformation encoding, not an underlying relationship. Few FFD solutions can create this level of curvature in the high-volume case, and are drawn from a small pool of possible designs with this combination of features. In

4 Assuming a budget of 1000 PE per run, an estimated 3000 hours (125 days) of computation would be required to run CMA-ES once in each of the 625 bins in a 25 X 25 prediction map, using 80 2.6ghz cores.

the feature/performance maps created with the parameterized representation, volume clearly dominates curvature, and high-curvature designs are at no clear disadvantage.

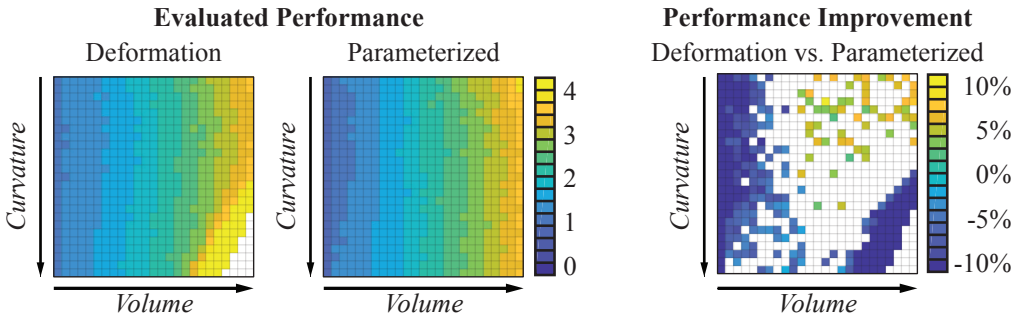

Figure 5.12: *True Performance of Designs Produced by SAIL with Different Representations.*
The resulting fitness when every design in the prediction map is evaluated in the simulator (fitness is drag force in Newtons, lower is better). Prediction maps created after running SAIL with a budget of 1000 precise evaluations for each encoding, with each bin containing the median fitness found over 20 replicates. Only significant comparisons ($p < 0.05$ determined by Mann-Whitney-Wilcoxon test) are shown, with lighter colors indicating better performance by the deformation encoding. Both encodings capture the strong relationship between volume and drag, but excel in different feature regions, with the deformation performing better in low-curvature, large-volume regions, but unable to express high-volume, high-curvature designs.

The feature region where the FFD encoding is unable to produce designs is an exception that highlights the similarity of performance of the encodings in the other regions. In more than two-thirds of bins the fitnesses of the designs found by the two encodings is within 5%, with those greater than 5% found at the edges of the feature ranges. This similarity demonstrates the capability of SAIL to illuminate the relationship between features and performance, independent of the representative power of the particular encoding.

MODEL ACCURACY Modeling performance in this more realistic problem is more difficult for several reasons: the encodings have more degrees of freedom (16 dimensions as opposed to 10 in the 2D airfoil experiments), the parameters of the encodings are not as closely linked to performance (as in 2D PARSEC airfoils), and the problem itself is much more complicated (three-dimensional versus two-dimensional flow). While we are not trying to predict the performance of any and all designs, we are still targeting a much larger set of solutions than is typical of surrogate-assisted optimization methods, which seek only to model performance around the global optimum.

Though the rank-based optimization process of MAP-Elites is forgiving of errors, and the bar for optimality in design exploration is lower than for pure optimization, to have confidence in our results we must first be confident that our models can predict performance with a degree of accuracy. At the end of each run of SAIL a prediction map was produced and every design evaluated in our flow simulator.

For both encodings the performance of the majority of designs was predicted within 5% of the values found in simulation (Figure 5.13, left). Though the FFD encoding was not able to produce designs in every feature region, overall the performance of the designs produced was easier to predict. The predicted performance of the majority of designs created using the deformation encoding was accurate within 0.05 N, while the majority of designs created using the parameterized encoding were only accurate within 0.10 N (Figure 5.13, right).

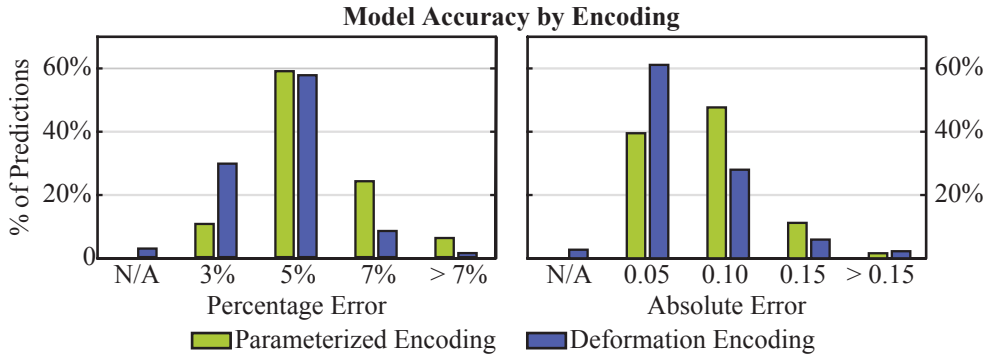

Figure 5.13: *Model Accuracy in Prediction Map by Encoding.* Distribution of prediction error of designs in the prediction maps produced after 1000 precise evaluations (median values over 20 replicates). In the case of the deformation encoding not all bins were explored, but are still counted toward the total so as not to distort the comparison between the encodings. Fitness values of designs ranged between 0.8 N and 5.8 N.

DESIGN EXPLORATION Different encodings may lead to different solutions to the same problem, but SAIL is able to find diverse, high-performing examples and accurately predict their performance regardless. Though a variety of solutions is produced by both the parameterized and deformation representations, the designs produced by each tend towards different themes. By examining the cross-sections of designs from each encoding in the same feature region, these biases are revealed (Figure 5.14).

The designs produced by the parameterized encoding are typically taller and thinner designs, lowering drag by reducing the frontal area which first hits the air. Designs produced by FFD have less flexibility at the leading edge, and earn higher fitness with smoother designs that guide flow from a larger frontal area. These strategies are shown in Figure 5.15 for high- and low-volume cases.

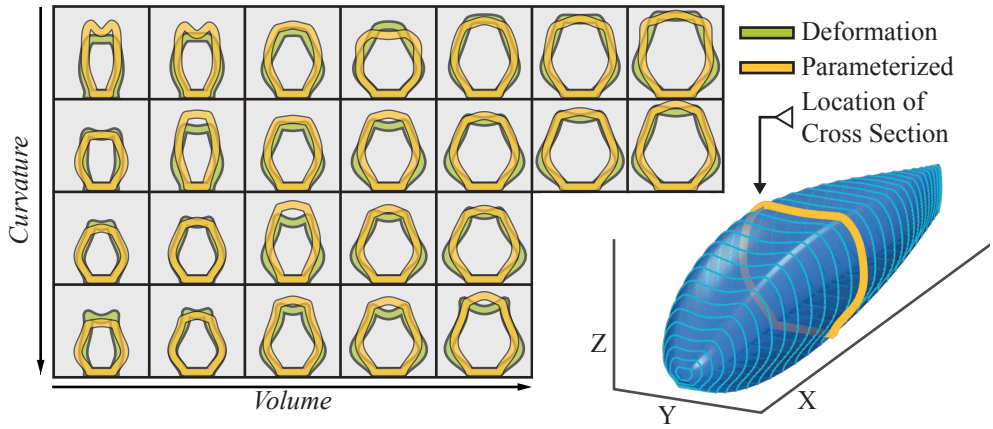

Figure 5.14: *Cross-Sections of Optimal Designs in Each Feature Region by Encoding.*
Middle cross section of optimal designs in selected feature regions, free form deformation in green, parameterized encoding in yellow. The different degrees of freedom to each encoding produce differing solutions for the same feature combinations.

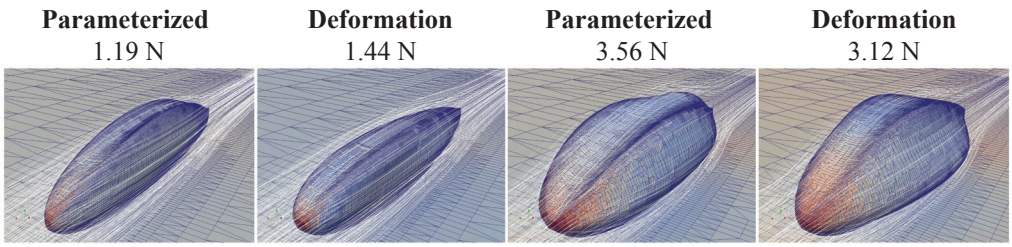

Figure 5.15: *Different Representations Lead to Different Design Approaches.*
Sample low-curvature designs produced by the parameterized and FFD encodings. Left: low-volume, Right: high-volume. Meshes colored by air pressure, with warmer colors indicating higher pressures. The parameterized encoding minimizes frontal area leading to a taller, thinner designs, reducing drag in the low-volume case. In the high-volume case the smoother designs created by FFD achieve higher performance.

While general design concepts can be discerned by browsing optimal designs, a more detailed understanding can be gained by viewing the parameters values of the optimal designs through a feature space lens. Figure 5.16 shows 16 maps, one for each variable of the parameterized encoding. Each map is colored according to the value of the respective parameter, showing the 16 values behind each design in a typical prediction map.

Visualizing the parameter values of the large number of designs produced by SAIL can allow designers to understand the interaction of parameters and features, and to tune their encodings by removing or introducing additional degrees of freedom or adjusting the range of existing parameters. When, for example, the optimal values are all at the edge of a parameter range, that range can be extended or shifted;

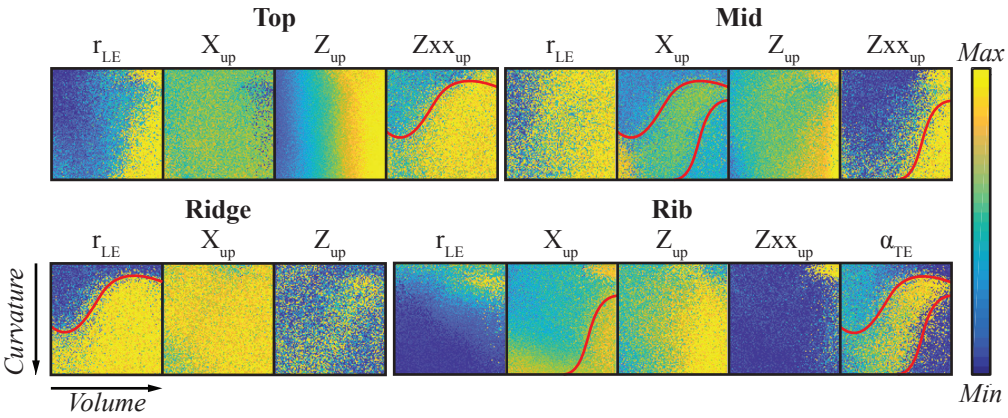

Figure 5.16: *Visualizing Parameter Values of Optimal Designs*
Parameter values of optimal parameterized designs in a 250 × 250 bin prediction map. Lighter colors indicate a value closer to the top of the parameter range. Visualizing the parameter ranges used in the optimal designs can aid designers in setting parameter bounds in an encoding. At a glance it is apparent for some parameters, such as the Ridge X_{up} of the Rib Zxx_{up}, the range should be shifted, as the optimal values are all at one extreme. The optimal values of the Top X_{up} are all mid range values, hinting that this range could be tightened for greater precision, while the noisy values of the Ridge Z_{up} may indicate that the range is too small. Visualizing the parameters of the optimal designs allows us to easily spot parameter relationships and design regions, such as the correlated design region borders in red, where sudden transitions in optimal values occur across multiple parameters in the same feature regions.

when the optimal values for a parameter are noisy then it follows that the value has little effect on fitness, meaning it is has no effect, or either unnecessary or the range is too narrow.

Correlations between parameters can also be easily detected, even visually. These correlations could be clues to underlying design concepts, allowing high-performing design prototypes to be identified *(Hagg et al., 2018)*. If parameters are correlated across the entire feature space they may be candidates for collapsing into a single degree of freedom, reducing the dimensionality of the problem and predisposing the encoding to faster convergence. With a large set of high-performing solutions statistical techniques, such as analysis of variance, could also be applied to analyze new representations in a way that would be impossible with only a handful of designs.

5.3.3 *Computational Cost*

Whenever fitness approximation techniques are applied, the trade-off between complexity of modeling and the expense of precise evaluation must be considered. In the case of illumination, because of the large

number of evaluations required, it is not difficult to tilt the balance in favor of modeling.

The cost of training and prediction with Gaussian process models can, however, become significant when larger data sets are considered. When applying GP models for exact inference, complexity is cubic in the size of the data set. Though starting from a small base cost, exact inference becomes infeasible when more than a few thousand samples are considered. This trend of increasing complexity can be seen in our own experiments (Figure 5.17), where the cost of training and prediction increases by nearly ten times from the first iteration to the last.

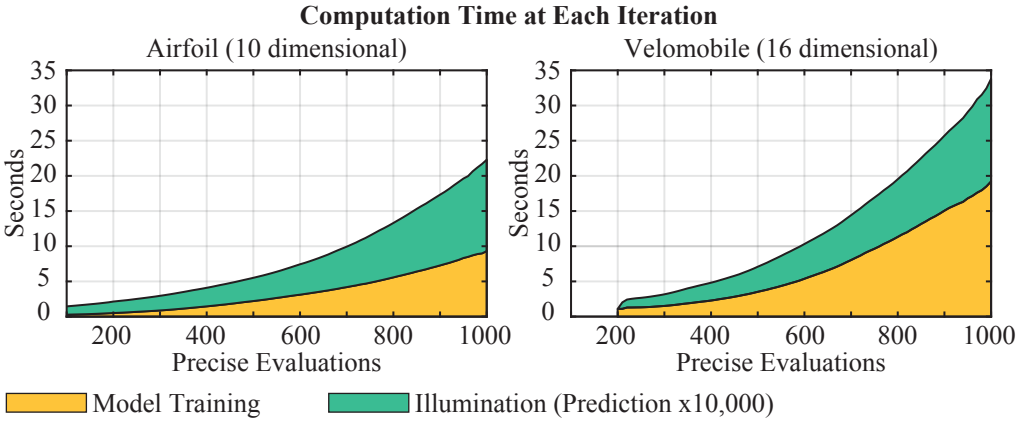

Figure 5.17: *Model Training and Prediction Costs in Each Domain.* Model training and prediction time over the course of a typical run, with 1 core of a 3.1 GHz processor (moving average). Values denote the time taken in a single iteration for computation costs of training a single GP model (*yellow*) and the time required to perform the 10,000 predictions used to create a single acquisition map (*green*).

SAIL was designed with expensive domains and small data sets in mind, and where data is more plentiful more sophisticated modeling techniques could be used to maintain performance. There are several methods which allow GPs to cope with larger data sets. Data can, for instance, be partitioned into separate groups *(Snelson and Ghahramani, 2007)*, or lower rank approximations can be made of the covariance matrix based around representative "inducing points" *(Quiñonero-Candela and Rasmussen, 2005; Quiñonero-Candela et al., 2010)*.

Though the computational cost of modeling and prediction increases cubically, we find that SAIL is still more efficient than using precise evaluations alone, even in inexpensive cases. In Figure 5.18 we examine the cumulative computational costs of model training, prediction, and precise evaluation. In the airfoil case one precise evaluation requires only a fraction of a second, and more time is spent on model training and prediction than on precise evaluation. At each iteration SAIL performs 10,000 predictions to illuminate an acquisition map,

with a total of nearly 1 million predictions over a single run. In our experiments, MAP-Elites was given a budget of 100,000 precise evaluations, or 6.5 hours of evaluation time. Even with this additional computation and exact results, the solutions found were still much worse than those found using 30 minutes of SAIL's combination of evaluation and modeling (see Figure 5.7 in Section 5.3.1).

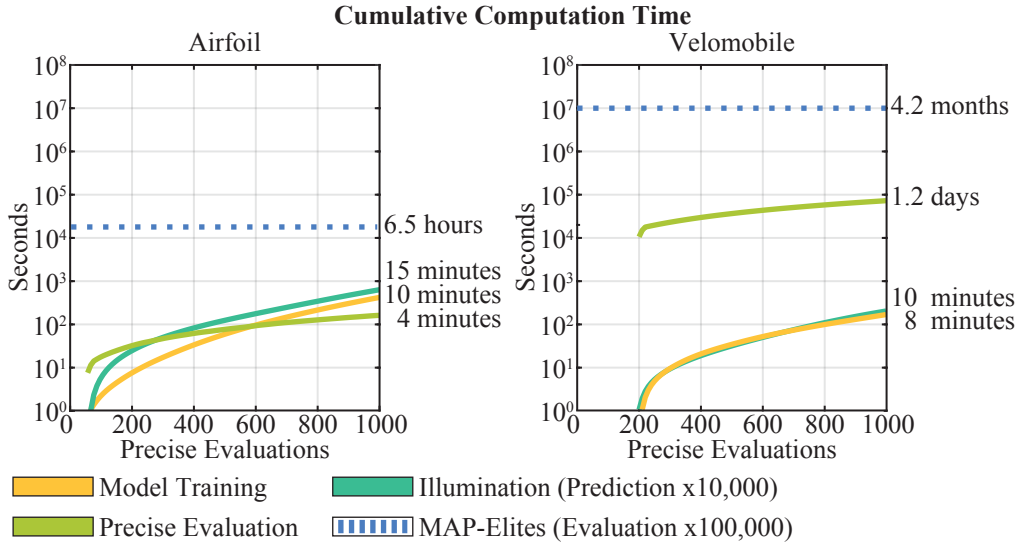

Figure 5.18: *Cumulative Computational Cost of SAIL Components.* Cumulative cost of each SAIL component in a typical run, using 1 core of a 3.1 GHz processor. Time in logarithmic scale. One illumination consists of 10,000 predictions. Model training and prediction time in the airfoil case includes both the drag and the lift models. Model training and Illumination are each performed 96 times in the airfoil case and 81 times in the velomobile case. 1000 precise evaluations are performed in both cases. For comparison the approximate computational cost of a 100,000 precise evaluation MAP-Elites run is also shown.

If the airfoil case illustrates the potential for SAIL to accelerate illumination even in inexpensive domains, the velomobile case demonstrates how SAIL can extend the reach of quality-diversity algorithms. In this more expensive case, SAIL spends more than a day of computation time evaluating 1000 designs and less than 20 minutes on model building and prediction. If we were to attempt to run MAP-Elites in this domain, the 100,000 evaluations would require more than 4 months of computation time. The illumination of the design space in this domain is *only* possible because of the surrogate modeling techniques in SAIL.

5.4 DISCUSSION

The SAIL algorithm produces a model of the objective function in high-fitness regions across the feature space, even with a limited computational budget. With the aid of these models, illumination algorithms can produce a diversity of high-performing designs which reveal relationships between features as well as biases in encodings.

Our experiments have shown that SAIL is effective without a carefully tuned domain specific encoding, which opens up the possibility of using it as a tool to assist in the creation and tuning of new encodings. Whether testing the capabilities of a newly designed representation, or iteratively improving an existing encoding, SAIL provides a way of understanding the inherent biases of a representation. Even if the encoding is destined for use in a more traditional optimization algorithm, SAIL allows a designer to visualize the variety of designs an encoding is able to express, the optima they are able to reach, and the ease of modeling their performance from their genotype. SAIL does not require powerful encodings, and could be used as a tool to create them.

In design exploration, the chief advantage illumination approaches have over multi-objective approaches is the ability to explore features which are not in opposition, or that have no relation to fitness at all. These features could be those which have an unknown effect on performance which designers would like to better understand, such as the percentage of a structure created with a new material, or features whose goal is not to improve performance, such as aesthetics.

Multi-objective optimization is used to power automated design exploration approaches known as 'innovization' *(Deb and Srinivasan, 2006)*. These approaches use optimization algorithms to reveal new design principles by searching for commonalities in sets of high-performing designs. The ability to produce designs in larger variety, and which vary across more dimensions of freedom, make illumination techniques an ideal fit to produce the required raw material of diverse high-performing solutions.

That prediction maps are most intuitively visualized in two dimensions does not limit the use of SAIL to only a pair of features. Prediction maps are created using a continuous model built from samples selected from the acquisition map. The acquisition map itself is merely a collection of candidate solutions for the model, and so has no direct connection to the prediction map. As such the form of the prediction map is not bound by the structure of the acquisition map: the resolution of the map could be different, feature ranges could be tuned to zoom in on a particular feature region, feature dimensions themselves could even be added or removed. For example, the 250×250 parameter maps in Figure 5.16 come from a prediction

map produced using a model created by running SAIL with a 25×25 acquisition map.

To allow for easier visualization, it is possible to run SAIL to build models using acquisition maps with many feature dimensions, and to then produce prediction maps which only examine the relationship between two of the features at a time. As the optimization algorithm which produces the prediction map is based only on the model, prediction maps can be produced with very little computation, even at high resolution. The acceleration provided by surrogate-assistance enables SAIL to not only act as a data-efficient method to perform illumination, but also allows the production of maps which are essentially continuous.

It may be necessary or advantageous to estimate the behavior or features of a given solution in addition to their performance. In many cases where a data-efficient version of MAP-Elites would be most useful, such as robotics, the feature or behavior description is obtained during evaluation, e.g. how active each leg of a hexapod is during a gait with a given controller *(Cully et al., 2015)*. Even in design cases, where simulation may not be necessary to derive features, instantiating the design from the genome can still represent a comparatively large computational expense, given that it must be done millions of times over the course of the algorithm. In these design cases models which estimate features could further accelerate SAIL, and given these samples ready availability could be made very accurate.

Integration of additional surrogate models could do more than accelerate. In our experiments when a design was simulated but did not converge, whether due to numerical instabilities or odd geometries that a simple solver like XFoil was not designed to model, the result was simply discarded and the next design taken in its place. If used on truly expensive problems, this approach is insufficient: we cannot afford to simply throw away data. Additional models could be used to estimate the likelihood of a design to converge in simulation, guiding the illumination process towards more robust designs. In evolutionary robotics similar approaches have been proposed to produce controllers which bridge the reality gap, avoiding solutions which are unlikely to work in the real world in the way that they do in simulation *(Koos et al., 2013)*.

Though MAP-Elites has shown remarkable potential, the intensive computation it requires precludes its use in many domains. By pairing MAP-Elites with surrogate assistance, a Bayesian optimization equivalent for illumination is created. By enabling illumination in computationally expensive domains, SAIL opens up new avenues for experiments and applications of quality-diversity techniques, especially in design.

The capability to rapidly understand the performance potential of the design space through concrete high-performing examples is

a potential boon to designers. SAIL not only accelerates the generative design cycle, but allows the effect of user-defined features to be examined, adding new flexibility to cooperative human-machine design exploration. Generative design tools which consider more than objectives, such as SAIL, can help designers explore what is possible, beyond what is optimal.

6

DATA-DRIVEN ENCODINGS (DDE)

Some figures, text, and ideas in this section are based on the following works which have either been previously published or are in review:

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2020). "Discovering Representations for Black-box Optimization." In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. Vol. 11.

6.1 INTRODUCTION

The method of encoding solutions is one of the most critical design decisions in optimization, as the representation defines the way an algorithm can move in the search space *(Rothlauf, 2006)*. Work on representations tends to focus on encoding priors or innate biases: aerodynamic designs evolved with splines to encourage smooth forms *(Olhofer et al., 2001)*, Compositional Pattern Producing Networks (CPPNs) with biases for symmetry and repetition in images and neural network weight patterns *(Stanley, 2007; Stanley et al., 2009)*, modularity induced in evolved neural networks *(Doncieux and Meyer, 2004; Mouret and Doncieux, 2008; Durr et al., 2010)*, or neural network structures which encode strong enough biases to perform without training *(Gaier and Ha, 2019)*.

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

Given a large set of example solutions, representations could be learned from data instead of being hand-tailored by trial-and-error: a learned representation would replicate the same biases toward performance and the same range of expressivity as the source data set. For instance, given a dataset of face images, a Variational Autoencoder (VAE) *(Kingma and Welling, 2014)* or a Generative Adversarial Network (GAN) *(Goodfellow et al., 2014)* can learn a low-dimensional latent space, or encoding, that makes it possible to explore the space of face images. In essence, the decoder which maps the latent space to the phenotypic space learns the "recipe" of faces. Importantly, the

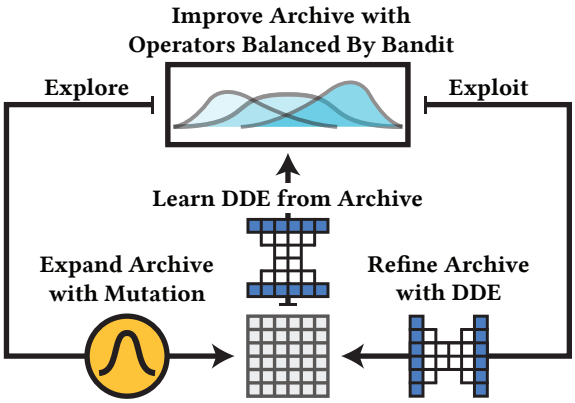

Figure 6.1: *Data-Driven Encoding MAP-Elites* (DDE-Elites) searches the space of representations to search for solutions. A data-driven encoding (DDE) is learned by training a VAE on the MAP-Elites archive. High fitness solutions, which increase the bias of the DDE toward performance, are found using the DDE. Novel solutions, which increase the range of solutions which can be expressed, are found using mutation operators. UCB1, a bandit algorithm, balances the mix of these explorative and exploitative operators.

existence of such a low-dimensional latent space is possible because *the dataset is a very small part of the set of all possible images*.

However, using a dataset of preselected high-performing solutions "traps" the search within the distribution of solutions that are already known: a VAE trained on white faces will never generate a black face. This limits the usefulness of such data-driven representations for discovering *novel* solutions to hard problems.

In this paper, we propose the use of the MAP-Elites algorithm *(Mouret and Clune, 2015)* to automatically generate a dataset for representations using only a performance function and a diversity space. Quality diversity (QD) algorithms *(Cully and Demiris, 2017)* like MAP-Elites are a good fit for representation discovery: creating archives of diverse high-performing solutions is precisely their purpose. Using the MAP-Elites archive as a source of example solutions, we can capture the genetic distribution of the highest performing solutions, or elites, by training a VAE and obtaining a latent representation. As the VAE is only trained on elites, this learned representation, or Data-Driven Encoding (DDE), has a strong bias towards solutions with high fitness; and because the elites have varying phenotypes, the DDE is able to express a range of solutions. Though the elites vary along a phenotypic continuum, they commonly have many genotypic similarities *(Vassiliades and Mouret, 2018)*, making it more likely to find a well-structured latent space.

Nonetheless, MAP-Elites will struggle to find high-performing solutions without an adequate representation. Fortunately, the archive is produced by MAP-Elites in an iterative, any-time fashion, so there is no "end state" to wait for before a DDE can be trained — a DDE

can be trained *during optimization*. The DDE can then be used to enhance optimization. By improving the quality of the archive the DDE improves the quality of its own source data, establishing a virtuous cycle of archive and encoding improvement.

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

In this paper, we address this challenge by mixing solutions generated with the DDE with solutions obtained using standard evolutionary operators. Our algorithm applies classic operators, such as Gaussian mutation, to create candidates which could not be captured by the current DDE. At the same time we leverage the DDE to generalize common patterns across the map and create new solutions that are likely to be high-performing. To avoid introducing new hyperparameters, we tune this exploration/exploitation trade-off optimally using a multi-armed bandit algorithm *(Garivier and Moulines, 2011)*.

This new algorithm, DDE-Elites, reframes optimization as a search for representations (Figure 6.1). Integrating MAP-Elites with a VAE makes it possible to apply quality diversity to high-dimensional search spaces, and to find effective representations for future uses. We envision application to domains that have straightforward but expansive low-level representations, for instance: joints positions at 20Hz for a walking robot ($12 \times 100 = 1200$ joint positions for a 5-second gait of a robot with 12 degrees of freedom), 3D shapes in which each voxel is encoded individually (1000-dimensional for a $10 \times 10 \times 10$ grid), images encoded in the pixel-space, etc.

Ideally, the generated DDE will capture the main regularities of the domain. In robot locomotion, this could correspond to periodic functions, since we already know that a 36-dimensional controller based on periodic functions can produce the numerous joint commands required every second to effectively drive a 12-joint walking robot in many different ways *(Cully et al., 2015)*. In many domains the space of possible solutions can be vast, while the inherent dimensionality of interesting solutions is still compact. By purposefully seeking out a space of solutions, rather than the solutions themselves, we can solve high-dimensional problems in a lower dimensional space.

6.2 BACKGROUND

6.2.1 *Optimization of Representations*

In his 30 year perspective on adaptation in evolutionary algorithms, Kenneth De Jong identified representation adaptation as "perhaps the most difficult and least understood area of EA design." *(De Jong, 2007)*

Despite the difficulty of creating adaptive encodings, the potential rewards have lured researchers for decades. Directly evolving genotypes to increase in complexity has a tradition going back to the eighties *(Goldberg et al., 1989; Altenberg, 1994)*. The strategy of optimizing a solution at low complexity and then adding degrees of freedom has proved effective on problems from optimal control *(Gaier and Asteroth, 2014a)*, to aerodynamic design *(Olhofer et al., 2001)*, to neural networks *(Stanley and Miikkulainen, 2002)*. Evolving the genome's structure is particularly important when the structure itself is the solution, such as in genetic programming *(Koza, 1990)* or neural architecture search *(Elsken et al., 2019; Gaier and Ha, 2019; Miikkulainen et al., 2019)*.

Recent approaches toward representation evolution have focused on genotype-phenotype mappings *(Bongard and Pfeifer, 2003)*. Neural networks, which map between inputs and outputs, are a natural choice for such 'meta-representations'. These mappings can evolve with the genome *(Simões et al., 2014; Scott and Bassett, 2015)*, or fix the genome and evolve only the mapping *(Stanley, 2007; Stanley et al., 2009)*.

Supervised methods have been previously applied to learn encodings. These approaches require a set of example solutions for training. Where large, well-curated data sets are available this strategy has proven effective at creating representations well suited to optimization *(Bontrager et al., 2018a,b; Volz et al., 2018; Giacomello et al., 2019; Fontaine et al., 2020; Schrum et al., 2020a,b)*, but where a corpus of solutions does not exist it must be created. In *(Moreno et al., 2018; Scott and De Jong, 2018)* these solutions were collected by saving the champion solutions found after repeatedly running an optimizer on the problem, with the hope that the learned representation would then be effective in similar classes of problems, but a principled method of collecting differing solutions was absent.

6.2.2 *MAP-Elites*

MAP-Elites *(Mouret and Clune, 2015)* is a QD algorithm which uses a niching approach to produce high-performing solutions which span a continuum of user-defined phenotypic dimensions. These phenotypic dimensions, or behavior descriptors, describe *the way* the problem is solved, and are often orthogonal to performance. MAP-Elites has been used in such diverse cases as optimizing the distance traveled by a walking robot using different legs *(Cully et al., 2015)*, the drag

of aerodynamic designs with varied volumes and curvatures *(Gaier et al., 2017a)*, and the win rate of decks composed of different cards in deck-building games *(Fontaine et al., 2019b)*.

MAP-Elites is a steady-state evolutionary algorithm which maintains a population in a discretized grid or 'archive'. This grid divides the continuous space of possible behaviors into bins, or 'niches' with each bin holding a single individual, or 'elite'. These elites act as parents, and are mutated to form new individuals. These child individuals are evaluated and assigned a niche based on their behavior. If the niche is empty the child is placed inside; if the niche is already occupied, the individual with higher fitness is stored in the niche and the other discarded. By repeating this process, increasingly optimal solutions which cover the range of phenotype space are found. The MAP-Elites algorithm is summarized in Algorithm 1 in Section 2.1.

Though phenotypically diverse the elites are often genotypically similar, existing in an "elite hypervolume", a high performing region of genotype space *(Vassiliades and Mouret, 2018)*. Just as in nature, where species as diverse as fruit flies and humans share nearly 60 percent of their genome *(Adams et al., 2000)*, the "recipe" for high performance is often composed of many of the same ingredients.

This insight was leveraged in *(Vassiliades and Mouret, 2018)* to create a new variation operator which considers the correlation among elites. Genes which vary little across the elites, and so are likely common factors that produce high performance, are also subject to the smallest amount of perturbation — lowering the chance their children stray from the elite hypervolume. Biasing mutation in this way ensures that exploration is focused on factors which induce phenotypic variation without drifting into regions of poor performance.

6.2.3 *Variational Autoencoders*

Autoencoders (AEs) *(Hinton and Salakhutdinov, 2006)* are neural networks designed to perform dimensionality reduction. AEs are composed of two components: an encoder, which maps the input to a lower dimensional latent space; and a decoder, which maps the latent space back to the original space. The decoder is trained to reconstruct the input through this lower dimensional latent "bottleneck". The encoder component can be viewed as a generalization of Principal Component Analysis *(Wold et al., 1987)*, with the latent space approximating principal components.

Though the AE is able to represent the data at a lower dimensionality, and reproduce it with minimal loss, it can still be a poor representation for optimization. An important quality of representations is 'locality', that a small change in the genotype induces a small change in the phenotype *(Rothlauf, 2006)*. When AEs are trained only to minimize reconstruction error they may overfit the distribution of

the training data and create an irregular latent space. The low-locality of such latent spaces limits their usefulnesses in optimization: nearby points in latent space may decode to very different solutions, meaning even a small mutation could have a large effect.

Variational autoencoders (VAEs) *(Kingma and Welling, 2014)* are AEs whose training is regularized to ensure a high-locality latent space. The architecture is broadly the same: an encoder and decoder mediated by a bottleneck, but rather than encoding the input as a single point it is encoded as a normal distribution in the latent space. When training the model a point from this input distribution is sampled, decoded, and the reconstruction error computed. By encoding the input as a normal distribution we induce the distributions produced by the encoder to be closer to normal. VAEs are trained by minimizing two terms: (1) the reconstruction error, and (2) the Kullback-Leibler (KL) divergence *(Kullback and Leibler, 1951)* of the latent space to a unit Gaussian distribution, giving the loss function:

$$loss = \|x - \hat{x}\|^2 + KL\left[N\left(\mu_x, \sigma\right), N(0,1)\right] \tag{6.1}$$

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

6.3 DDE-ELITES

Every representation biases optimization in some way, improving optimization by limiting the range of solutions that can be expressed to those which are valid or high-performing *(Rothlauf, 2006)*. But finding a balance between expressivity and bias is an arduous task requiring considerable domain expertise. Our method, DDE-Elites, automates the process of representation design and learns new encodings in tandem with search — allowing optimization and representation learning to improve each other in a self-reinforcing cycle.

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

DDE-Elites is a variant of the MAP-Elites algorithm. The core component of competition within a niched archive is maintained, but novel methods of producing child solutions are introduced. Child solutions

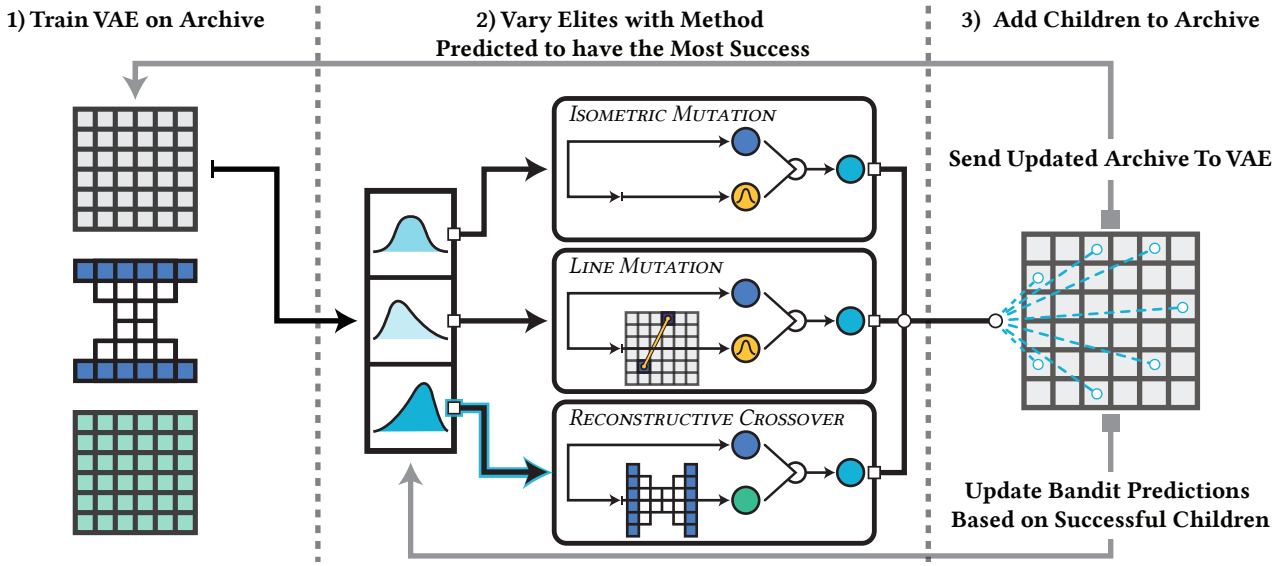

Figure 6.2: *DDE-Elites Algorithm*
(1) A VAE is trained on the archive, and used to create a 'reconstructive crossover' operator which creates new solutions by averaging the parameters of an individual with its own reconstruction; (2) the mix of exploitative and explorative variation operators predicted to have the most success is chosen by the multi-armed bandit algorithm UCB1 and used to create new solutions; (3) the new solutions are added to the archive and the success rate of the applied variation operator is updated.

are created using an encoding learned from the archive. This encoding is refined as the archive improves, which in turn improves the optimization process. DDE-Elites optimizes an archive of varied solutions by reframing optimization as a search for the best representation, rather than the best solution.

The DDE-Elites algorithm proceeds as follows (see Figure 6.2 and Algorithm 3): (1) a DDE and *reconstructive crossover* operator is created by training a VAE on the archive; (2) the probability of using each variation operator is determined by the UCB1 bandit algorithm; (3) MAP-Elites is run with the chosen variation operator probabilities. The success rate of the variation operators to create solutions is used to update the bandit and the improved archive is used to create a new DDE and reconstructive crossover operator.

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

Algorithm 3 DDE-Elites

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

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

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

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

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

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

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

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

$$\mathbf{x}_i^{(t+1)} = \frac{1}{2} * (\mathbf{x}_i^{(t)} + VAE.Decode(VAE.Encode(\mathbf{x}_i^{(t)}))) \tag{6.2}$$

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

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

In addition to isometric Gaussian mutation commonly used in MAP-Elites, we apply the line mutation operator proposed in *(Vassiliades and Mouret, 2018)*. Line mutation imposes a directional component on the Gaussian perturbations. During mutation the parent genome is compared to a random genome from the archive. The variance of

mutation in each dimension is then scaled by the difference in each gene:

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \sigma_1 \mathcal{N}(0, \mathbf{I}) + \sigma_2 \left(\mathbf{x}_j^{(t)} - \mathbf{x}_i^{(t)} \right) \mathcal{N}(0, 1) \tag{6.3}$$

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

In many cases certain parameter values will be correlated to high fitness, regardless of the individual's place in behavior space. The line operator is a simple way of exploiting this similarity, but in contrast to reconstructive crossover does not limit expressivity – allowing it to be used as a method of exploring new solutions. Though both the reconstructive crossover and line mutation operators take advantage of the similarities between high performing individuals, their differing approaches allow them to be effectively combined as explorative and exploitative operators.

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

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

Faced with a trade-off between exploration and exploitation we frame the choice of operators as a multi-armed bandit problem *(Auer et al., 2002)*. Multi-armed bandits imagine sets of actions as levers on a slot machine, each with their own probability of reward. The goal of a bandit algorithm is to balance exploration, trying new actions, and exploitation, repeating actions that yield good rewards. Bandit approaches are straightforward to implement and have been previously used successfully to select genetic operators *(DaCosta et al., 2008)*.

We define a set of possible actions as usage ratios between reconstructive crossover, line mutation, and isometric mutation. The ratio of $[\frac{1}{4}, \frac{3}{4}, 0]$, for example, would have solutions created by reconstructive crossover with a probability of $\frac{1}{4}$, line mutation with a probability of $\frac{3}{4}$, and never with isometric mutation. Each action is used to create a batch of child solutions and a reward is assigned in proportion to

the number of children who earned a place in the archive. At each generation a new action is chosen, and the reward earned for that action recorded.

Actions are chosen based on UCB1 *(Auer et al., 2002)*, a simple and effective bandit algorithm which minimizes regret. Actions with the greatest potential reward are chosen, calculated as:

$$Q(a) + \sqrt{(2 \log t)/(N_t(a))} \tag{6.4}$$

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

Our archive is in constant flux, and so the true reward of each mix of operators changes from generation to generation. To handle the non-stationary nature of the problem we use a sliding window *(Garivier and Moulines, 2011)*, basing our predictions only on the most recent generations.

6.4 EXPERIMENTS

PLANAR ARM INVERSE KINEMATICS We demonstrate the effectiveness of DDEs and DDE-Elites on in the inverse kinematics (IK) problem of a 2D robot arm[1], a common benchmark for in QD and robotics behavior exploration *(Baranes and Oudeyer, 2013; Benureau and Oudeyer, 2016; Cully and Demiris, 2017; Vassiliades and Mouret, 2018)*. Given target coordinates, a configuration of joint angles should be found to place the end effector at the target. To solve this task, a discretized behavior space is defined over the x,y plane and MAP-Elites finds a configuration of joint angles which places the end effector in each bin. The location of the end effector is derived for an arm with n joints with angles y with using the forward kinematics equation:

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

There are many solutions to this IK problem, but solutions with lower joint variance are preferred to allow for smoother transitions between configurations. We define fitness as the negative joint variance: $-\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu)^2$ where $(\mu = \sum_{i=1}^{n} y_i)$.

To summarize: the phenotype is the angle of each joint, the behavior is the x,y coordinates of the end effector, and the fitness the negative variance of the joint angles. The difficulty of the problem can be easily scaled up by increasing the number of joints in the arm: we solve

1 see Figure 6.5 for a visualization of this domain

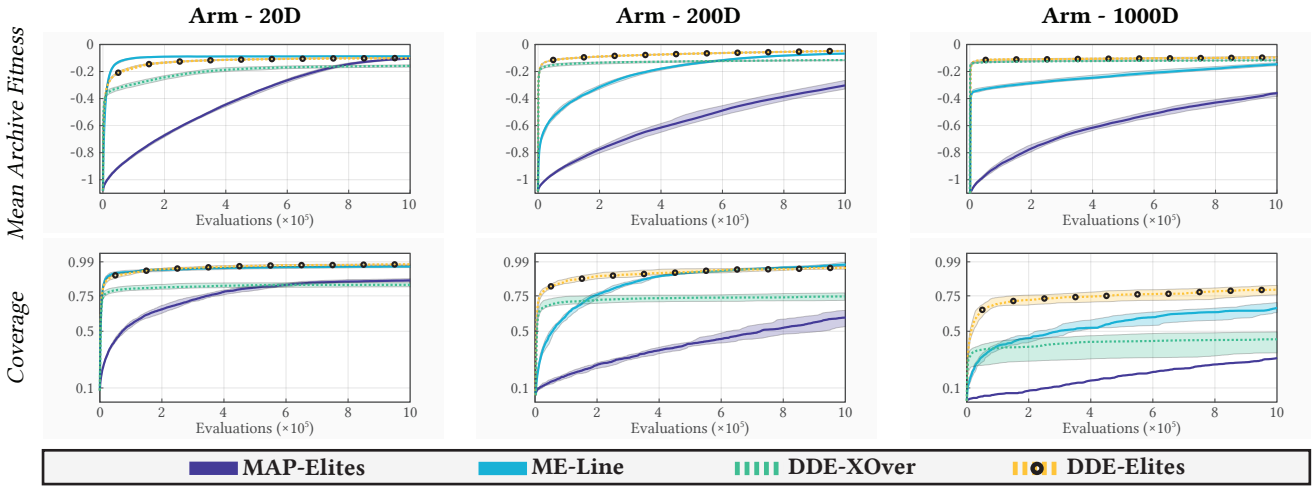

Figure 6.3: *Archive Illumination*
Archive illumination performance of MAP-Elites with different variation operators: standard isometric mutation (*MAP-Elites*), line mutation (*ME-Line*), reconstructive crossover (*DDE-XOver*) and DDE-Elites, which uses the UCB1 bandit algorithm to choose between the three at every generation. We measure fitness as the mean fitness of all solutions in the archive; coverage as the fraction of behavior space bins which contain solutions. Results over 20 replicates with lines indicating medians and quartile bounds shaded. The median of DDE-Elites, our approach, is additionally noted with black dots. All final results are significantly different ($p < 0.01$ Mann-Whitney U) in fitness and coverage. Progress is shown in evaluations (0 to 1 million); a batch size of 100 evaluations per generation was used, so this scale corresponds to generations from 0 to 10,000.

this task with 20, 200, and 1000 joints. When a DDE is used, 10 latent dimensions are used for the 20D arm, and 32 dimensions for the 200 and 1000D arms. The same archive structure is used for all domains. A unit circle is divided into 1950 bins, with each bin defined by a Voronoi cell *(Vassiliades et al., 2017)* with centers placed in a ring formation[2].

6.4.1 *Archive Illumination*

We first demonstrate the ability of DDE-Elites to scale up illumination to high-dimensional problems. The performance of DDE-Elites is compared to three algorithmic variants: the canonical MAP-Elites algorithm using isometric mutation (*MAP-Elites*); MAP-Elites using line, or directional, mutation (*ME-Line*); and MAP-Elites using the reconstructive crossover (*DDE-XOver*). Our proposed approach *DDE-Elites* uses all operators at a ratio determined by the UCB1 bandit algorithm. These treatments are summarized in Table 6.1.

2 See supplementary material for a visualization of this structure

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

Table 6.1: Algorithm variants. DDE-Elites is our approach.

These variants are compared based on the quality of the archive at each generation (Figure 6.3). Archives are judged based on two metrics: (1) coverage, the number of bins filled, and (2) performance, the mean fitness of solutions.[3]

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

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

The line mutation operator is clearly able to leverage the similarities in high performing solutions across the archive — in every case performing far better than the isometric mutation operator. The mechanism for doing this, adjusting the range of parameter mutations, does not appear to scale well enough to handle very high dimensional problems. The reconstructive crossover operator is able to rapidly find high-performing solutions even in high-dimensional spaces, but is

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

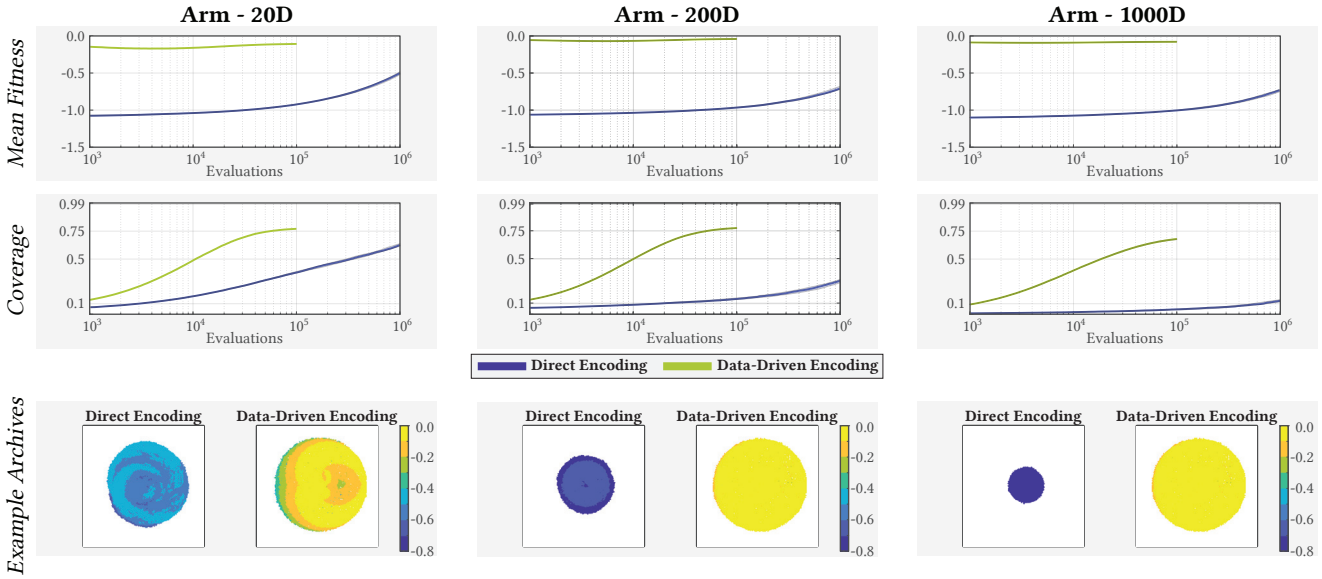

Figure 6.4: *Illumination with a Data-Driven Encoding*
Top: Performance of MAP-Elites algorithm when run with direct or a data-driven encoding. The data-driven encoding was trained on an archive with 1950 bins structured in a ring; the archive used here is 10,000 bins arranged as centroidal Voronoi tessellations covering a square. When using the direct encoding, MAP-Elites was given one order of magnitude more evaluations (note logarithmic scale of evaluations). Fitness is measured as the mean fitness of all solutions in the archive, coverage as the fraction of behavior space bins which contain solutions. Results over 20 replicates with lines indicating medians and quartile bounds shaded. *Bottom:* Example final archives using each representation, with discovered bins colored by fitness.

poor at exploring. Search with reconstructive crossover is confined to the distribution of genes that already exist in the archive, if used exclusively that distribution of genes is limited to the initial population. By combining these operators — expanding the range of genes in the archive with mutation, and spreading high performing genes with reconstructive crossover — DDE-Elites is able to create high-performing archives even in high-dimensional problems.

6.4.2 *Illumination with DDE*

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

does more than guide search, but learns a representation, we search the space again, using the found DDE in place of the direct encoding.

We run the standard MAP-Elites algorithm, with isometric mutation only, using a learned DDE[4] acting as our genome. In the 20D arm this DDE has 10 parameters, in the 200D and 1000D arms the DDE has 32 parameters. No previous solutions are maintained, *only the trained DDE*. For reference we compare to the MAP-Elites algorithm using the direct encoding. An order of magnitude fewer evaluations were budgeted when using the DDE. To ensure we are not simply recreating memorized points, we replace the original archive configuration of 1950 bins arranged in a ring with an archive of 10,000 bins created using centroidal Voronoi tessellations as in *(Vassiliades et al., 2017)*[5]

In every case the DDE far outperforms the direct encoding, reaching the same levels of fitness and coverage with several orders of magnitude fewer evaluations (Figure 6.4). The DDE can express the same range of solutions as were found in the original archive, and finds them rapidly. The core $\approx$ 7500 reachable bins of the archive filled in less than 100,000 evaluations — fewer than 15 evaluations per bin. The found solutions are also invariably high performing.

Such improvement cannot be explained away by the decrease in dimensionality of the search. In both low and high dimensional cases the bias toward high performance is also apparent: the mean fitness curve is nearly flat at the optima, indicating that when new solutions are added to the map they are already near optimal. The contrast with the direct encoding is stark: the direct encoding requires considerable effort to find even poor solutions, the DDE finds only good solutions. DDE-Elites not only produces better solutions, but learns a representation of the underlying elite hypervolume – independent structure of the archive used originally.

6.4.3 *Optimization with Learned Encodings*

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

We illustrate this generality by again solving the arm inverse kinematics problem with the black-box optimizer CMA-ES *(Hansen, 2008)*. A set of target positions for the end effector is defined (Figure 6.5, left), and CMA-ES used to find a joint configuration which reaches each target. In one case optimization is performed using the DDE; in the other the direct encoding is used. The DDE is not only biased toward

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

5 These bins cover even unreachable regions of the descriptor space, and so the maximum coverage is limited to $\approx$ 75% of the bins.

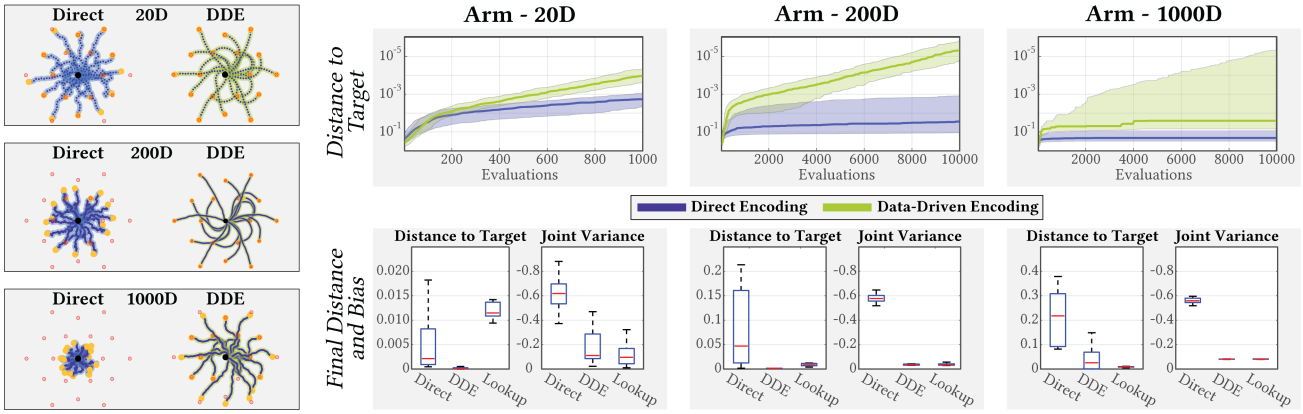

Figure 6.5: *Optimization with Direct and Data-Driven Encodings*
CMA-ES is given a set budget to find a solution with a target behavior, and searches with either a direct encoding or a DDE.
Left: Example solutions for target matching with the direct and data driven encodings. End effectors in yellow, targets in red.
Top: Optimization over time of median distance (dotted line) to the 18 targets over 20 replicates (quartiles shaded).
Bottom: The final solutions found by CMA-ES using the direct and data-driven encoding, with the nearest solutions to the targets in the archive the DDE was trained on (*Lookup*) shown for comparison. Solutions are judged by the final distance to the targets, and the joint variance. Joint variance was not optimized by CMA-ES, but optimized by DDE-Elites during the creation of the DDE, biasing the solutions produced. All differences are significant (Mann-Whitney U, $p < 0.001$) excluding the distance to target of the *DDE* and *Lookup* results in the 1000D case ($p > 0.2$).

high performance, but presents a much lower dimensional problem to be solved — in the 1000D case two orders of magnitude fewer dimensions. As a baseline we compare the final configurations to the accuracy of simply choosing the nearest point in the map that was used to train the DDE. If the DDE can surpass this "lookup" approach it is evidence that the DDE is an encoding useful for interpolating to new solutions, not only a method of encoding those already known.

When optimizing with the DDE, CMA-ES quickly finds solutions to the target hitting problems with a precision never matched with the direct encoding (Figure 6.5, top). The DDE is doing more than recalling memorized configurations: the distance of the closest solution in the archive the DDE was trained on is also surpassed. Moreover, a bias for *how* the problem is solved is built into the representation (Figure 6.5, bottom). As the DDE was trained only on solutions with high fitness, that is low joint variance, this same property is found in the solutions found by CMA-ES with the DDE — even without searching for them. With the DDE CMA-ES not only finds solutions to the IK problem, with the built-in priors of the DDE it finds the solutions we want.

6.5 DISCUSSION

Learning representations by combining quality diversity (here, MAP-Elites) and generative models (here, a VAE) opens promising research avenues for domains in which optimizations of the same cost function are launched continuously. This is, for example, the case of Model Predictive Control *(Mayne et al., 2000)*, in which the sequence of actions for the next seconds is optimized at every time-step of the control loop, or the case of shape optimization in interactive design tools *(Bendsøe and Sigmund, 1995; Hoyer et al., 2019)*, in which each modification by the user requires a novel optimization.

In preliminary experiments, we searched for an encoding to describe action sequences for a walking robot. The results show that using MAP-Elites to generate a diversity of sequences, then using a VAE to learn a representation leads to an encoding that can accelerate future optimizations by several orders of magnitude. Nevertheless, using the representation during optimization, as described in this paper, did not accelerate the quality diversity optimization as much as in the high-dimensional arm used here. One hypothesis is that the regularities in action sequences are harder to recognize than in the arm experiments, especially at the beginning of the process. For instance, it might help to use an auto-encoder that is especially designed for sequences *(Vaswani et al., 2017; Co-Reyes et al., 2018)*.

For other tasks, appropriate generative models could be explored, for example convolutional models for tasks with spatial correlations *(Salimans et al., 2015)*. In addition, though the latent spaces created by VAEs are easier to navigate than those created by normal autoencoders, even better models offer the opportunities for further improvements. Much work has been done to create VAEs which have even better organized latent spaces *(Higgins et al., 2017; Burgess et al., 2018; Chen et al., 2018; Kim and Mnih, 2018)*, ideally with each dimension responsible for a single phenotypic feature such as the lighting or color of an image.

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

bin. To best utilize the bandit, other methods of judging performance in QD algorithms should be explored.

Beyond performance advantages, for both the current and future optimizations, these "disentangled" representations offer even more interesting opportunities. Reducing the dimensionality of the search space into meaningful components would allow rapid model-based optimization of single solutions *(Shahriari et al., 2015)*, or entire archives *(Gaier et al., 2018a)*. Engineers could interactively explore and understand such encodings, laying bare the underlying properties responsible for performance and variation — and so from encodings receive, rather than provide, insight and domain knowledge.

7 DISCUSSION

In this manuscript we introduced methods to integrate machine learning and quality diversity approaches that extend the capabilities of both. These contributions fall into two broad categories:

- Accelerating quality diversity through predictive modeling and active learning.
- Creating bespoke generative models and representations by evolving the datasets which inform them.

In this section we discuss broadly the approaches we introduced: their current limitations, promising remedies, and future directions for research they present.

7.1 ACCELERATING QD THROUGH PREDICTIVE MODELING

The most dramatic example of the the performance gains offered by model-based optimization were those provided by SAIL — showing that a diversity of high performing solutions can be found in roughly the same amount of evaluations as a single solution. Improving data-efficiency by multiple orders of magnitude does more than just save compute, it radically expands the range of domains to which QD can be applied. Where the goal is to create collections of alternative solutions, SAIL provides archives of nearly continuous resolution with minimal compute. Such detailed results not only allows expensive design domains to be tackled with QD, but also provides tools for rapid exploration and analysis of representations and descriptors. However, the model-based nature of SAIL prevents it from being applied as universally as MAP-Elites.

Modeling Limitations

All surrogate-assisted algorithms inherit the weaknesses of the models on which they depend. Some difficulties, such as predicting the performance of neural networks with evolved topologies, we were able to address in the course of our research. Other limitations of the Gaussian process models used by Bayesian Optimization-inspired algorithms in this manuscript were not directly addressed here, but are areas of active research. As a non-parametric method the complexity of GP models increases with the number of samples, putting a computational limit on number of samples SAIL or SA-NEAT can

make use of. Training of GP models is $\mathcal{O}(N^3)$ and prediction $\mathcal{O}(N^2)$, though many sparse approximations have been developed *(Smola and Bartlett, 2001; Csató and Opper, 2002; Seeger et al., 2003)*.

One strategy to deal with large numbers of samples is the combination of global and "local" GPs, models which only use nearby solutions rather than all solutions to create a prediction *(Snelson and Ghahramani, 2007)*. Local models such as these could be useful in creating better performing models for SA-NEAT, whose fractured search space is a poor fit for global models. The locality of local GPs is based on the same distance as the predictions, the distance in parameter space. QDs provide another notion of locality, locality in the descriptor space. Depending on the nature of the descriptor, this could be an even better way of demarcating which solutions are best at informing predictions than the parameter space: modeling performance of a robot controller producing forward motion by comparing to it to the set of controllers that also produce forward motion is likely more useful than those with the most similar oscillator values.

Distance measures also break down in higher dimensional spaces, causing kernel-based methods like GPs to struggle *(Srinivas et al., 2010; Bull, 2011)*. With the standard formulation of a GP, optimizing a representation with hundreds of parameters is just not feasible. Existing methods for using GPs in high dimensional space rely on dimensionality reduction techniques, by using learned *(Djolonga et al., 2013)* or random *(Wang et al., 2016)* projections to lower dimensions. Here again, we may be able to leverage the low-dimensional descriptor spaces as an alternative notion of the locality needed to make predictions.

Modeling of Descriptors

This impressive results of SAIL come with an important caveat: the descriptor must be able to be derived *without* evaluation. Many interesting characteristics fall into this class of descriptor, from design features in games and engineering to genotypic characteristics like age or modularity.

Many descriptors, which are typically characteristics of how the solutions is solved, can only be found be by evaluating the solution. The impressive results of QD in game playing and robotics depend on exploring a behavior space which cannot be easily derived from the genome — it may be easy to count the number of enemies in a Mario level, but much harder to predict how a neural network will guide Mario through that level. When characteristics are more than phenotypic, but behaviors that depend on the environment, even small changes can make a big difference. Such functions are difficult to model with the techniques we have introduced here: GP models make smoothness assumptions about the nature of the functions they model, but non-linearities are not at all uncommon in behavioral

interactions. Adding to the difficulty, in many cases the environment is also dynamic or variable — meaning the same genome might not always translate to the same behavior.

Yet the exploration of behavioral descriptors is one of QDs most promising capabilities, so any progress in predicting behavioral descriptors would pay off handsomely. In some cases standard techniques may be sufficient to model behaviors, but in others new approaches will have to be developed. Predicted descriptors may need to incorporate uncertainty, and so be expressed not as points but as distributions. The structured grid and elitism of MAP-Elites is not well suited to such a fuzzy descriptor, but though SAIL was presented using MAP-Elites to illuminate the descriptor space, the paradigm could be easily adapted to NSLC-like unstructured collections such as those in *(Lehman and Stanley, 2011b; Cully and Demiris, 2017)*.

7.2 CREATING BESPOKE REPRESENTATIONS

By reframing the QD optimization process as a search for a high quality representation rather than for a high quality archive we were able to apply QD to much higher dimensional problems. Not only were we able to solve higher dimensional problems orders of magnitude faster, but the resulting data-driven encoding is useful for other tasks in the same domain. We can use this representation for rapid optimization for other tasks, or efficiently 'up sampling' archives by using the DDE to illuminate high resolutions or differently structured maps.

Dimensionality Reduction for SAIL in High Dimensions

It is not unrealistic then that the dimensionality reduction performed by DDE-Elites could also be useful for modeling. SAIL and DDE-Elites could be combined, by basing performance predictions on the encoding learned by the DDE. When only using a precomputed DDE this would be a straightforward way to accelerate illumination, or Bayesian optimization of single solutions. If the illumination processes of SAIL and DDE-Elites were integrated the reconstruction error gives another ad-hoc measure of model uncertainty. Solutions which we know do not compress well would be rewarded by any UCB acquisition function, driving exploration and improvement of both generative and predictive models.

Disentangled Models

Surrogate-assisted optimization would be even more effective if the solution representation was not only low-dimensional, but meaningful. Creating disentangled representations *(Higgins et al., 2017; Burgess et al., 2018; Chen et al., 2018; Kim and Mnih, 2018)*, whose parameters

closely corresponded to features would allow models to better predict performance — just as knowing the curvature of a wing is more informative than knowing the location of a single spline control point. Parameters that correspond to cohesive properties of a solution may be key to predicting its behavior without evaluation. Beyond possible modeling benefits, interpretable features would be of great use in design contexts, allowing users to explore and tweak designs at higher levels of abstraction.

An ideal DDE might correspond directly to the descriptor: (x,y) coordinates which correspond directly to the thousand joint positions that place an arm at that location. Such representations could no doubt be created, though may be as interesting as a look up tables. A more intriguing option could be to learn disentangled representations and explore the found parameter values as additional descriptors — an alternate path to evolving and learning both the representation and descriptors.

7.3 COMMON VIEWS OF MODELING IN QD

We introduced methods of accelerating QD algorithms with two kinds of models, generative and predictive. In both cases this acceleration is accomplished by manipulating the trade-off of bias and expressivity. In the generative case this adjustment is explicit: generative models are limited to a subspace based on the solutions they were trained on, so search is restricted to an "elite hypervolume". When optimizing with predictive models this restriction still exists, if only implicitly. Though there are not restrictions about where the search can travel, once a model is established in practice optimization takes place within and around the found hypervolume. Exploring far from this modeled space and search is lost in a sea of uncertainty. To expand this modeled space SAIL rewards solutions for uncertainty, and DDE-Elites explores independently of DDE with standard mutation operators.

Both generative and predictive models are ways of representing the elite hypervolume. In typical applications of MAP-Elites the archive is a sampling of this elite hypervolume — the product of the illumination process is a concrete set of solutions from that volume sampled at a predefined resolution. The model-based illumination methods we have presented produce this set of solutions, but also provide formulations of the boundaries of the elite hypervolume. By defining this space, either by reshaping the representation or the objective function, we allow it to be quickly explored at any resolution. MAP-Elites produces a look-up table; model-based illumination produces a function.

7.4 TOWARDS QD FOR AERODYNAMIC DESIGN

The vision motivating this project from the start was the creation of algorithms that would allow designers to explore a variety of complex and non-intuitive aerodynamic shapes. Such exploration required not only quality diversity algorithms to explore this variety but representations which could create the high-dimensional output needed to define complex three-dimensional shapes.

This vision presented three primary challenges:

1. Enable exploration of a diversity of design options
2. Simultaneous optimization of many of interacting parameters
3. Strictly limited budget imposed by the use of CFD

Significant progress was made in this PhD toward these goals, but much remains to be done. Even as some challenges were overcome, additional opportunities and avenues for investigation arose. We summarize some of these, along with promising lines of attack below.

Data-efficient illumination in high dimensions

SURROGATE-ASSISTED ILLUMINATION OF CPPNS

The initial notion was that complex aerodynamic designs could be evolved with CPPNs *(Stanley, 2007; Clune and Lipson, 2011; Clune et al., 2013)*. By using an indirect encoding we could encode and optimize high resolution deformations or voxel representations. The variable resolution of the output of CPPNs might allow us to evolve designs at varied scales — optimizing first more general forms and then increasing the resolution. It was this indirect encoding approach to evolving high dimensional designs that led us to experimenting with the hereditary kernels of SA-NEAT (Chapter 3).

Unfortunately, preliminary tests with CPPN-NEAT representations for aerodynamics were disappointing. Deformations encoded with CPPNs were easily surpassed by ES techniques. Pixel representations held more promise, with a higher dimensional and more interesting space to explore. We attempted to evolve the back wing of race car, a shape composed of multiple wings. Evolved in 2D as pixels we hoped that CPPNs could evolve a multitude of interacting shapes to create useful airflow. This hope was not realized. We found it difficult to bootstrap complexity in the CPPNs — we created a variety of wedges but little else. Only objective search was used at this point — and the failure reminded us of those from the Picbreeder experiments, motivating the study in Chapter 4 to see if QD could really help us gather the stepping stones needed to find greater complexity.

These stepping stone experiments uncovered other intriguing paths which were then followed, leaving the question of surrogate-assistance

and CPPNs unanswered. Experiments with MAP-Elites and CPPNs showed that QD was effective for finding stepping stones and optimizing CPPNs, but this study was more of a confirmation of assumptions than a new finding. MAP-Elites has been used to optimize CPPNs before *(Mouret and Clune, 2015)*. Whether CPPNs can be optimized by SA-NEAT or SAIL is another topic.

There are reasons to believe that the hereditary kernel of SA-NEAT would work with CPPNs as well — the compatibility distance of NEAT works with CPPNs, so perhaps a kernel based on that distance would as well. But there are also reasons to believe it would be difficult. CPPNs are a highly pleiotropic representation: a single mutation can drastically alter the phenotype in multiple ways. Diverse populations can help keep search on track — there may be many "hopeful monsters" with radically different phenotypes than their parents — but there are also many who make only small changes. The GP models used here make some smoothness assumptions — assumptions that are just not true of the CPPN genotype space.

A more promising direction for CPPNs than the hereditary distance outlined in this thesis are phenotypic distance kernels *(Hagg et al., 2019; Stork et al., 2019)*. As a proxy for distance, these techniques use the difference between different neural networks' outputs when given identical inputs. Though only in its infancy, this approach allows comparison of neural networks with varied topologies while better preserving smoothness — and are applicable beyond networks evolved with NEAT.

SURROGATE-ASSISTED ILLUMINATION OF DDES

The experiments performed in Chapter 4 to better understand QD and stepping stones were themselves a stepping stone that led us to reconsider our approach to high-dimensional optimization with QD. Though the autoencoder was intended as a way of deriving features, we found that we could also use it as a representation. When images in the map were reconstructed using the autoencoder in every single case fitness was improved — hinting that autoencoder captured some aspects of high performance.

This observation led to the creation of the DDE-Elites algorithm in Chapter 6, a pivot from CPPNs to DDEs for high-dimensional illumination. The DDE approach allows us to perform illumination in higher dimensional domains with an order of magnitude fewer evaluations than MAP-Elites, but the absolute number of evaluations is still far too many for applications like aerodynamics.

The obvious extension is a combination of SAIL with DDEs. SAIL provides data-efficient illumination of low-dimensional representations, and the DDE provides the low-dimensional representation of the high dimensional problem. To optimize an already existing DDE

is straightforward with SAIL — the DDE can be optimized like any other representation.

The more ambitious path is to integrate SAIL into the creation of DDEs. DDE-Elites produces solutions which are not created directly from the latent space — but every solution in the map can be encoded by the VAE, and predictions made based on their latent representation. The UCB measure of utility may need to be augmented: we would now be looking not only for points which have high performance, high uncertainty, but also high reconstruction error — indicating a point that will expand the DDE.

Exploration and Analysis

INTERACTIVE EXPLORATION WITH QD

The tools created here for data-efficient QD are only one piece of the puzzle for design exploration. Exploration of a design space is a process that requires movement — the user must be able to examine the results and then decide on the next direction. SAIL gives the capability to see where we are in the design space, but there are many opportunities to help users direct where they are going.

The flexibility of creating prediction maps based only on models enables rapid illumination with different conditions. These conditions could be as simple as increasing the resolution of the map, or as involved as illuminating an entirely different set of features. All manner of prediction maps can be created to view the design space, as understood by the model, through different lenses. With more understanding of the space further model refinement could also be directed by choosing the next points to evaluate: designs in interesting feature regions, high performing designs, or those whose predicted performance seems dubious are all possible candidates.

Prediction maps produced by SAIL can provide a high level view of the search landscape that encompass thousands of designs — that is impressive, but is of questionable real utility. Design exploration is, at its core, an act of human-computer interaction. To extract the most value out of SAIL it must be combined with tools that make the designs and knowledge accessible and actionable. Progress has been in interfaces to explore sets of precomputed designs *(Matejka et al., 2018)* — integration with SAIL would allow these programs to produce new designs on the fly adding another dimension of interactivity.

MODEL AND REPRESENTATION ANALYSIS

The ability for SAIL to illuminate a search space gives it great potential as a tool for better understand our models and our representations. Though we explored some analysis of representations in Chapter 5, these examinations were done by hand. To improve representations many of these decisions could be automated: resizing of parameter

ranges, collapsing parameter which are correlated in high performing solutions, fixing parameters which have little effect on fitness or are always the same for high performing solutions. Such automated efforts could doubly useful to polish DDEs into representations which will be reused.

7.5 NEW DIRECTIONS

REUSING DDES

A recent work *(Hoyer et al., 2019)* recast the traditional method of structural optimization, which acts directly to optimize a volume, to one which optimizes a convolutional neural network whose output corresponds to that volume. This reparameterization allowed optimization to take advantage of the inbuilt priors in CNNs — biases which allowed structures to be optimized on several spatial scales at once. The designs produced with this alternate formulation not only proved stronger, the objective being optimized, but also more easily manufactured, with more solid columns and fewer 'spider web' like structures. That even well studied problems like structural optimization can benefit from alternate representations and formulations in leads us to believe that learned encodings hold promise as more than curiosities — but have real impact in improving real world design optimization. DDEs could be created for common design tasks, like structural optimization, with additional biases encoded for desirable properties like manufacturability.

DDEs created for reuse could be useful building blocks for rapid exploration in design, and could be especially useful in domains like control and robotics where rapid adaptation is even more important. In the original hexapod experiments with MAP-Elites the produced archive of was used as a look-up table of behaviors, a discrete menu of different controllers, designed to deal with a discrete range of damage conditions. DDEs have the potential to act instead as a continuous behavioral representations, allowing fine tuning of controllers to varied damage conditions, environmental perturbations, and reality gaps. The strong priors of DDEs that allow for rapid adaptation could provide a basis for advances in Model Predictive Control *(Mayne et al., 2000)* or meta-learning *(Lemke et al., 2015; Kaushik et al., 2020b)*.

DDES AND DESCRIPTORS

Advanced DDEs could blur the bridge between encodings and descriptors, creating representations which encode descriptors — allowing us to precisely define the characteristics we want the representation to encode. Already VAEs have been designed which are able to use provided class labels as latent dimensions *(Antoran and Miguel, 2019)*, descriptors could be provided in the same way — allowing generation of new samples with a given descriptor.

While creating well disentangled DDEs may present additional difficulties there is reason to be optimistic — there is a large community of machine-learning researchers working on these problems, even if they have not viewed them through this context.

Novelty-search-derived methods like QD are successful because of the paradigm shift to search based on descriptors. These methods have used evolution and a "I know it when I see it" approach to descriptors, measuring the descriptor along with fitness. If representations can be learned which incorporate descriptors these descriptors can be optimized directly — possibly even with gradient based methods. While such a search may have difficulty extrapolating to new descriptors, it could prove effective for interpolating between known descriptors to produce continuous behavioral repertoires for robots or fine grained control over procedural content generation.

DERIVED DESCRIPTORS

In Chapter 4 we introduced a method of deriving descriptors using autoencoders and the solutions within the archive. Automatic derivation of descriptors is not a new idea, and efforts typically have involved dimensionality reduction techniques of some kind. Our approach used the reconstruction error, DeLeNoX *(Liapis et al., 2013)* compresses the phenotype to get a lower dimensional descriptor, and AURORA *(Cully, 2019)* compresses a high dimensional descriptor into a lower dimensional one.

The DDE presents a different avenue. The DDE is a low dimensional representation of the "elite hypervolume", and this encoding could itself be explored with QD — with a selection, or all, latent dimensions acting as additional descriptors. The VAE encoder produces descriptors by compression, and the range of the DDE genotype is explored. Such an approach then explores not only the space of behaviors, but the components of variation within high performing solutions.

Deriving descriptors is, in many applications, a solution in search of a problem. In design users may very well know what descriptors they want to explore — the difficulty is less in intention than in communication. During a generative exploration of a space a user may see what they like, but not be able to define it what makes it interesting. In this case data-driven ways of defining descriptors may be useful. This could be done in an interactive way, with users selecting generated designs with characteristics they like or dislike or in a data driven way. Such an approach could be especially useful for things like aesthetics, where a style might be learned by examples and labels and then a range, such as Brutalist to Bauhaus, can be explored.

8

CONCLUSION

Throughout this thesis, we have presented algorithms designed with the goal of supporting exploration in the design process. Such exploration relies on finding not only optimal solutions, but diverse solutions. Furthermore for algorithms to be useful for exploration they must also be efficient. Design problems often rely on expensive simulations to judge the quality of solutions, so if an algorithm cannot find a diversity of solutions efficiently it cannot be used at all. The proposed algorithms attempt to combine optimality, variety, and efficiency — only by fulfilling all of these objectives can we produce something of use to designers.

In Chapter 3 we presented a new algorithm, SA-NEAT, to accelerate the optimization of neuroevolution algorithm NEAT. The algorithm introduces additional optimization methods to place NEAT into a Bayesian Optimization framework, and showed that by doing so we are able to improve the efficiency of the algorithm by more than three times of benchmark control tasks. The greatest innovation in this algorithm was not the optimization processes, but the kernel which allowed predictions to be performed at all. By leveraging the genealogical information already recorded by NEAT a distance measure was derived which allowed non-parametric models to be used to make predictions. When these models were used to guide evolution that performed as well as using the true fitness value of the solutions. The quality of these models is encouraging for the extension of surrogate-assisted optimization to other approaches that use the NEAT genotype, such as CPPNs *(Stanley, 2007)* or WANNs *(Gaier and Ha, 2019)*.

We tested the capabilities of QD algorithms to collect "stepping stone" to solve complex problems in Chapter 4. A classic illustration of the weakness of objective search, the recreation of a target image using a CPPN, was revisited. These images were found by users through undirected search and serendipity — processes optimization algorithms have difficulty replicating. Such a challenge is also impossible for novelty search: without any direction at all there is no mechanism to match a target. MAP-Elites was able to perform much better than a purely objective based approach. Not only were the found solutions recognizably similar to the target as opposed to formless blobs, tracking the progress of optimization showed that reach these solutions multiple detours to poor performing solutions were required.

In Chapter 5 we introduced the SAIL algorithm, a blend of BO and QD designed for use in expensive design problems. SAIL leverages the insight that BO can not only compensate for expensive but also

numerous evaluations. Though QD is searching for many different kinds of solutions, the objective function is the same throughout, and so a common model can be used. The SAIL algorithm adapts MAP-Elites to follow an active learning approach that learns a model of the objective function across the descriptor space. On a benchmark aerodynamics problem we show that in the same number of evaluations for the black-box optimizer CMA-ES to find a single solution SAIL is able to find an *archive* of solutions with equivalent optimality.

Such increases in data-efficiency do more than save compute, but radically expand the domains and applications of QD. We demonstrate the use of SAIL on a 3D aerodynamics problem possible in days rather than months because of the gains in efficiency. As optimization is based only on a model, maps of the descriptor space can be produced at near continuous resolution with minimal compute. Such maps allow for rapid and detailed understanding of the elite hypervolume found by MAP-Elites and the representation used to find it. By allowing rapid illumination of expensive design spaces and detailed post-hoc analysis SAIL is a promising partner for generative design.

SAIL is limited by the ability to predict fitness and descriptors. In the design cases these descriptors were able to be derived, and the genotypes were low-dimensional. This low-dimensionality is important as though BO is one of the most data-efficient optimization algorithms, it relies on GP models, which are only effective with a small number of parameter and samples.

In Chapter 6 we presented a different approach to improving data-efficiency by making use of generative models. The search for optimal solutions is reframed as a search for representations with a strong bias toward optimality. These learned representations, or data-driven encodings (DDE), were created by training Variational Autoencoders on the high performing solutions created by MAP-Elites. Though other approaches have used generative models trained on preexisting datasets as encodings, we proposed an algorithm, DDE-Elites, which produces the data for its own encoding — during optimization. By using the DDE to improve optimization, and optimization to improve the DDE, we were able to accelerate MAP-Elites dramatically. DDE-Elites solves low-dimensional benchmark problems an order of magnitude faster than MAP-Elites, and high (1000D) benchmark control problems which MAP-Elites cannot solve at all.

The DDE produced during optimization can be reused by any black-box optimization algorithm in the same domain. We showcased this ability in two examples: (1) to rapidly create a new MAP-Elites archive of higher resolution and different form, and (2) to be used by CMA-ES to find single solutions. In both cases qualitative comparisons to a direct encoding are difficult to make — even random initialization of the DDE typically performs better than the direct encoding does after thousands of evaluations. Though creating these DDEs are far more

efficient than MAP-Elites, but still distant from the levels of efficiency found by SAIL — or that are required for very expensive domains.

In Chapter 7 we discuss the limitations, commonalities, and opportunities of SAIL and DDE-Elites. Potential ways of addressing the constraints imposed on SAIL by the use of GP models and derived descriptors are outlined, the most intriguing of which is to combine SAIL and DDE-Elites. Making predictions based on a DDE could combine the best of both algorithms. Both approaches reframe the optimization problem as a modeling problem: SAIL models the objective function around the elite hypervolume; DDE-Elites models the genotype around the elite hypervolume. These paradigm shifts open up several directions for further research. From the detailed exploration of hand-designed and learned representations with SAIL to learning disentangled encodings whose behavior can be tuned with a parameter — unifying descriptor and encoding.

In this manuscript we presented quality diversity algorithms that learn models of the objective, allowing MAP-Elites to find thousands of high performing designs after only evaluating hundreds, and which learn models of high-performing genotypes, allowing the optimization hundreds of parameters while only searching dozens. Extending the applicability of MAP-Elites brings real-world engineering domains into reach for quality diversity. But more important than the successes of these specific techniques is the validation of an underlying intuition about machine learning and quality diversity: QD and ML are far from being opposing paradigms, but complimentary approaches whose deeper integration can unlock powerful capabilities in both.

Kullback, Solomon and Richard A Leibler (1951). "On information and sufficiency." In: JSTOR.

Kushner, Harold J (1964). "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise." In:

Berliner, Hans J (1974). *Chess as problem solving: the development of a tactics analyzer*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.

Hicks, Raymond Morton, Earll M Murman, and Garret N Vanderplaats (1974). "An assessment of airfoil design by numerical optimization." In:

Regal, Philip J (1975). "The evolutionary origin of feathers." In: *The Quarterly Review of Biology* 50.1, pp. 35–66.

Friedman, Jerome H, Jon Louis Bentley, and Raphael Ari Finkel (1977). "An algorithm for finding best matches in logarithmic expected time." In: *ACM Transactions on Mathematical Software (TOMS)* 3.3, pp. 209–226.

Mockus, Jonas, Vytautas Tiesis, and Antanas Zilinskas (1978). "The application of Bayesian methods for seeking the extremum." In: *Towards global optimization* 2.117-129, p. 2.

McKay, Michael D, Richard J Beckman, and William J Conover (1979). "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code." In: *Technometrics* 21.2, pp. 239–245.

Roubik, David W (1980). "Foraging behavior of competing Africanized honeybees and stingless bees." In: *Ecology* 61.4, pp. 836–845.

Goldschmidt, Richard (1982). *The material basis of evolution*. Vol. 28. Yale University Press.

Gould, Stephen Jay and Elisabeth S Vrba (1982). "Exaptation—a missing term in the science of form." In: *Paleobiology* 8.1, pp. 4–15.

Sanfeliu, A and K-S Fu (1983). "A distance measure between attributed relational graphs for pattern recognition." In: *IEEE transactions on systems, man, and cybernetics* 3, pp. 353–362.

Dyck, JAN (1985). "The evolution of feathers." In: *Zoologica Scripta* 14.2, pp. 137–154.

Janzen, Daniel H (1985). "On ecological fitting." In:

Plunkett, Roy J (1986). "The history of polytetrafluoroethylene: discovery and development." In: *High Performance Polymers: Their Origin and Development*. Springer, pp. 261–266.

Sederberg, Thomas W and Scott R Parry (1986). "Free-form deformation of solid geometric models." In: *ACM SIGGRAPH computer graphics* 20.4, pp. 151–160.

Goldberg, David E, Jon Richardson, et al. (1987). "Genetic algorithms with sharing for multimodal function optimization." In: *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*. Hillsdale, NJ: Lawrence Erlbaum, pp. 41–49.

Wold, Svante, Kim Esbensen, and Paul Geladi (1987). "Principal component analysis." In: Elsevier.

Niederreiter, H (1988). "Low-discrepancy and low-dispersion sequences." In: *Journal of Number Theory*. ISSN: 0022314X.

Goldberg, David E, Bradley Korb, Kalyanmoy Deb, et al. (1989). "Messy genetic algorithms: Motivation, analysis, and first results." In:

Brooks, Rodney A (1990). "Elephants don't play chess." In: *Robotics and autonomous systems* 6.1-2, pp. 3–15.

Koza, John R (1990). *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University, Department of Computer Science Stanford, CA.

McCarthy, John (1990). "Chess as the Drosophila of AI." In: *Computers, chess, and cognition*. Springer, pp. 227–237.

Mokyr, Joel (1991). "Evolutionary biology, technological change and economic history." In: *Bulletin of Economic Research* 43.2, pp. 127–149.

Yin, Xiaodong and Noel Germay (1993). "A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization." In: *Artificial neural nets and genetic algorithms*. Springer, pp. 450–457.

Altenberg, Lee (1994). "Evolving better representations through selective genome growth." In: *First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. IEEE.

Horn, Jeffrey, Nicholas Nafpliotis, and David E Goldberg (1994). "A niched Pareto genetic algorithm for multiobjective optimization." In: *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*. Ieee, pp. 82–87.

Lister, Adrian M (1994). "The evolution of the giant deer, Megaloceros giganteus (Blumenbach)." In: *Zoological Journal of the Linnean Society* 112.1-2, pp. 65–100.

Bendsøe, Martin P and Ole Sigmund (1995). *Optimization of structural topology, shape, and material*. Vol. 414. Springer.

Harik, Georges R (1995). "Finding Multimodal Solutions Using Restricted Tournament Selection." In: *ICGA*, pp. 24–31.

Mahfoud, Samir W (1995). "Niching methods for genetic algorithms." PhD thesis. University of Illinois at Urbana-Champaign.

Pétrowski, Alain (1996). "A clearing procedure as a niching method for genetic algorithms." In: *Proceedings of IEEE international conference on evolutionary computation*. IEEE, pp. 798–803.

Thompson, Adrian (1996). "An evolved circuit, intrinsic in silicon, entwined with physics." In: *International Conference on Evolvable Systems*. Springer, pp. 390–405.

Gould, Stephen Jay (1997). "The exaptive excellence of spandrels as a term and prototype." In: *Proceedings of the National Academy of Sciences* 94.20, pp. 10750–10755.

Obayashi, Shigeru and Takanori Tsukahara (1997). "Comparison of optimization algorithms for aerodynamic shape design." In: *AIAA journal* 35.8, pp. 1413–1415.

LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Rubner, Yossi, Carlo Tomasi, and Leonidas J Guibas (1998). "A metric for distributions with applications to image databases." In: *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE, pp. 59–66.

Prum, Richard O (1999). "Development and evolutionary origin of feathers." In: *Journal of Experimental Zoology* 285.4, pp. 291–306.

Samareh, Jamshid A (1999). "A Survey of Shape Parameterization Techniques." In: June, pp. 333–343. ISSN: 0001-1452. DOI: 10.2514/2.1391.

Sobieczky, H (1999). "Parametric airfoils and wings." In: *Recent Development of Aerodynamic Design*.

Adams, Mark D et al. (2000). "The genome sequence of Drosophila melanogaster." In: American Association for the Advancement of Science.

Dietrich, Michael R (2000). "From hopeful monsters to homeotic effects: Richard Goldschmidt's integration of development, evolution, and genetics." In: *American Zoologist* 40.5, pp. 738–747.

Gailing, O and K Bachmann (2000). "The evolutionary reduction of microsporangia in Microseris (Asteraceae): transition genotypes and phenotypes." In: *Plant Biology* 2.04, pp. 455–461.

Jones, Terry D et al. (2000). "Nonavian feathers in a late Triassic archosaur." In: *Science* 288.5474, pp. 2202–2205.

Mayne, David Q et al. (2000). "Constrained model predictive control: Stability and optimality." In: Elsevier.

Barnett, Lionel (2001). "Netcrawling-optimal evolutionary search with neutral networks." In: *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*. Vol. 1. IEEE, pp. 30–37.

Ebner, Marc, Mark Shackleton, and Rob Shipman (2001). "How neutral networks influence evolvability." In: *Complexity* 7.2, pp. 19–33.

Hansen, N and A Ostermeier (2001). "Completely derandomized self-adaptation in evolution strategies." In: *Evolutionary computation*.

Knowles, Joshua D, Richard A Watson, and David W Corne (2001). "Reducing local optima in single-objective problems by multi-objectivization." In: *International conference on evolutionary multi-criterion optimization*. Springer, pp. 269–283.

Moritz, DML and JW Kadereit (2001). "The genetics of evolutionary change in Senecio vulgaris L.: a QTL mapping approach." In: *Plant Biology* 3.05, pp. 544–552.

Olhofer, Markus, Yaochu Jin, and Bernhard Sendhoff (2001). "Adaptive encoding for aerodynamic shape optimization using evolution strategies." In: *2001 Congress on Evolutionary Computation*. IEEE.

Rieppel, Olivier (2001). "Turtles as hopeful monsters." In: *BioEssays* 23.11, pp. 987–991.

Smola, Alex J and Peter L Bartlett (2001). "Sparse greedy Gaussian process regression." In: *Advances in neural information processing systems*, pp. 619–625.

Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer (2002). "Finite-time analysis of the multiarmed bandit problem." In: Springer.

Campbell, Murray, A Joseph Hoane Jr, and Feng-hsiung Hsu (2002). "Deep blue." In: *Artificial intelligence* 134.1-2, pp. 57–83.

Csató, Lehel and Manfred Opper (2002). "Sparse on-line Gaussian processes." In: *Neural computation* 14.3, pp. 641–668.

Deb, Kalyanmoy et al. (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II." In: *IEEE transactions on evolutionary computation* 6.2, pp. 182–197.

Emmerich, Michael et al. (2002). "Metamodel assisted evolution strategies." In: *International Conference on parallel problem solving from nature*. Springer, pp. 361–370.

Li, Jian-Ping et al. (2002). "A species conserving genetic algorithm for multimodal function optimization." In: *Evolutionary computation* 10.3, pp. 207–234.

Norell, Mark et al. (2002). "'Modern'feathers on a non-avian dinosaur." In: *Nature* 416.6876, pp. 36–37.

Stanley, K and R Miikkulainen (2002). "Evolving neural networks through augmenting topologies." In: *Evolutionary computation*.

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

Deb, Kalyanmoy (2003). "Unveiling innovative design principles by means of multiple conflicting objectives." In: *Engineering Optimization* 35.5, pp. 445–470.

Renner, Gábor and Anikó Ekárt (2003). "Genetic algorithms in computer aided design." In: *Computer-Aided Design* 35.8, pp. 709–726.

Seeger, Matthias, Christopher Williams, and Neil Lawrence (2003). "Fast Forward Selection to Speed Up Sparse Gaussian Process Regression." In: *Artificial Intelligence and Statistics 9*.

Ashburn, Ted T and Karl B Thor (2004). "Drug repositioning: identifying and developing new uses for existing drugs." In: *Nature reviews Drug discovery* 3.8, pp. 673–683.

Bell, John E and Patrick R McMullen (2004). "Ant colony optimization techniques for the vehicle routing problem." In: *Advanced engineering informatics* 18.1, pp. 41–48.

Dew, Nicholas, Saras D Sarasvathy, and Sankaran Venkataraman (2004). "The economic implications of exaptation." In: *Journal of Evolutionary Economics* 14.1, pp. 69–84.

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

Donninger, Chrilly and Ulf Lorenz (2004). "The chess monster Hydra." In: *International Conference on Field Programmable Logic and Applications*. Springer, pp. 927–932.

Forseth, Irwin N and Anne F Innis (2004). "Kudzu (Pueraria montana): history, physiology, and ecology combine to make a major ecosystem threat." In: *Critical reviews in plant sciences* 23.5, pp. 401–413.

Jensen, Mikkel T (2004). "Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation." In: *Journal of Mathematical Modelling and Algorithms* 3.4, pp. 323–347.

Phillips, Ben L and Richard Shine (2004). "Adapting to an invasive species: toxic cane toads induce morphological change in Australian snakes." In: *Proceedings of the National Academy of Sciences* 101.49, pp. 17150–17155.

ChessBase (2005a). *Dark horse ZackS wins Freestyle Chess Tournament*. URL: https://en.chessbase.com/post/dark-horse-zacks-wins-freestyle-che-tournament.

ChessBase (2005b). *'Hydra is the Kasparov of computers'*. URL: https://en.chessbase.com/post/-hydra-is-the-kasparov-of-computers-.

ChessBase (2005c). *Hydra misses the quarter-finals of Freestyle tournament*. URL: https://en.chessbase.com/post/hydra-mies-the-quarter-finals-of-freestyle-tournament.

Hasenjäger, M and B Sendhoff (2005). "Three dimensional evolutionary aerodynamic design optimization with CMA-ES." In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation - GECCO '05*.

Jin, Y (2005). "A comprehensive survey of fitness approximation in evolutionary computation." In: *Soft computing*.

Menzel, S, M Olhofer, and B Sendhoff (2005). "Application of free form deformation techniques in evolutionary design optimisation." In: *Proceedings of 6th World Congress on Structural and Multidisciplinary Optimization*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.78.2051{\&}rep=rep1{\&}type=pdf.

Quiñonero-Candela, Joaquin and Carl Edward Rasmussen (2005). "A unifying view of sparse approximate Gaussian process regression." In: *Journal of Machine Learning Research* 6.Dec, pp. 1939–1959.

Stanley, K, B Bryant, and R Miikkulainen (2005). "Evolving neural network agents in the NERO video game." In: *Proceedings of the IEEE*, pp. 182–189.

Wang, Huai et al. (2005). "The origin of the naked grains of maize." In: *Nature* 436.7051, p. 714.

Deb, Kalyanmoy and Aravind Srinivasan (2006). "Innovization: Innovating design principles through optimization." In: *Proceedings Of The 8th Annual Conference On Genetic And Evolutionary Computation - GECCO '06*. ACM, pp. 1629–1636.

Giannakoglou, K.C., D.I. Papadimitriou, and I.C. Kampolis (2006). "Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels." In: *Computer Methods in Applied Mechanics and Engineering*. ISSN: 00457825.

Hinton, G and R Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks." In: *Science*.

Hornby, Gregory et al. (2006). "Automated antenna design with evolutionary algorithms." In: *Space 2006*, p. 7242.

Neuhaus, M, K Riesen, and H Bunke (2006). "Fast suboptimal algorithms for the computation of graph edit distance." In: *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer.

Parkinson, David, Pia Mukherjee, and Andrew R Liddle (2006). "Bayesian model selection analysis of WMAP3." In: *Physical review D* 73.12, p. 123523.

Pfeifer, R and J Bongard (2006). *How the body shapes the way we think: a new view of intelligence*. MIT press.

Rasmussen, C and C Williams (2006). "Gaussian Process for Machine Learning." In: *Gaussian Process for Machine Learning*.

Rothlauf, Franz (2006). "Representations for genetic and evolutionary algorithms." In: *Representations for Genetic and Evolutionary Algorithms*. Springer.

Theißen, Günter (2006). "The proper place of hopeful monsters in evolutionary biology." In: *Theory in Biosciences* 124.3-4, pp. 349–369.

Whiteson, S and P Stone (2006). "Evolutionary function approximation for reinforcement learning." In: *Journal of Machine Learning Research* 7.May, pp. 877–917.

De Jong, Kenneth (2007). "Parameter setting in EAs: a 30 year perspective." In: *Parameter setting in evolutionary algorithms*. Springer.

Greiner, David et al. (2007). "Improving computational mechanics optimum design using helper objectives: an application in frame bar structures." In: *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, pp. 575–589.

Holzer, Dominik, Richard Hough, and Mark Burry (2007). "Parametric design and structural optimisation for early design exploration." In: *International Journal of Architectural Computing* 5.4, pp. 625–643.

Snelson, Edward and Zoubin Ghahramani (2007). "Local and global sparse Gaussian process approximations." In: *Artificial Intelligence and Statistics*, pp. 524–531.

Stanley, K (2007). "Compositional pattern producing networks: A novel abstraction of development." In: *Genetic programming and evolvable machines*.

Zhou, Z, YS Ong, and PB Nair (2007). "Combining global and local surrogate models to accelerate evolutionary optimization." In: *IEEE Transactions on Systems, Man and Cybernetics*.

DaCosta, Luis et al. (2008). "Adaptive operator selection with dynamic multi-armed bandits." In: *10th annual conference on Genetic and evolutionary computation*.

Dumas, L (2008). "CFD-based optimization for automotive aerodynamics." In: *Optimization and Computational Fluid Dynamics*.

Handl, Julia, Simon C Lovell, and Joshua Knowles (2008). "Multiobjectivization by decomposition of scalar cost functions." In: *International Conference on Parallel Problem Solving from Nature*. Springer, pp. 31–40.

Hansen, N (2008). *CMAES Version 3.61.beta*.

Lehman, J and K O Stanley (2008). "Exploiting open-endedness to solve problems through the search for novelty." In: *Proc. of ALIFE*, pp. 329–336.

Mouret, Jean-Baptiste and Stéphane Doncieux (2008). "MENNAG: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars." In: Springer.

Aaltonen, T et al. (2009). "Observation of electroweak single top-quark production." In: *Physical review letters*.

Flager, Forest et al. (2009). "Multidisciplinary process integration and design optimization of a classroom building." In: *Journal of Information Technology in Construction (ITcon)* 14.38, pp. 595–612.

Forrester, A I J and AJ Keane (2009). "Recent advances in surrogate-based optimization." In: *Progress in Aerospace Sciences*.

Hastings, Erin Jonathan, Ratan K Guha, and Kenneth O Stanley (2009). "Automatic content generation in the galactic arms race video game." In: *IEEE Transactions on Computational Intelligence and AI in Games* 1.4, pp. 245–263.

Mouret, Jean-Baptiste and Stéphane Doncieux (2009). "Evolving modular neural-networks through exaptation." In: *2009 IEEE congress on evolutionary computation*. IEEE, pp. 1570–1577.

Osepchuk, John M (2009). "The history of the microwave oven: A critical review." In: *2009 IEEE MTT-S International Microwave Symposium Digest*. IEEE, pp. 1397–1400.

Padulo, M et al. (2009). "Airfoil Design Under Uncertainty with Robust Geometric Parameterization." In: *17th AIAA/ASME/AHS Adaptive Structures Conference*. ISSN: 02734508.

Raiko, T and M Tornio (2009). "Variational Bayesian learning of nonlinear hidden state-space models for model predictive control." In: *Neurocomputing*.

Riesen, K and H Bunke (2009). "Approximate graph edit distance computation by means of bipartite graph matching." In: *Image and Vision computing*.

Stanley, Kenneth O, David B D'Ambrosio, and Jason Gauci (2009). "A hypercube-based encoding for evolving large-scale neural networks." In: MIT Press.

Brochu, E, VM Cora, and N De Freitas (2010). "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning." In: *arXiv preprint arXiv:1012.2599*.

Durr, Peter, Dario Floreano, and Claudio Mattiussi (2010). "Genetic representation and evolvability of modular neural controllers." In: IEEE.

Hamdaoui, M et al. (2010). "Using Multiobjective Evolutionary Algorithms and Data-Mining Methods to Optimize Ornithopters' Kinematics." In: *Journal of aircraft* 47.5, pp. 1504–1516.

Kasparov, Garry (2010). "The chess master and the computer." In: *The New York Review of Books* 57.2, pp. 16–19.

Lian, Yongsheng, Akira Oyama, and Meng-Sing Liou (2010). "Progress in design optimization using evolutionary algorithms for aerodynamic problems." In: *Progress in Aerospace Sciences*.

Quiñonero-Candela, Joaquin, Carl Edward Rasmussen, AnÃbal R Figueiras-Vidal, et al. (2010). "Sparse spectrum Gaussian process regression." In: *Journal of Machine Learning Research* 11.Jun, pp. 1865–1881.

Risi, Sebastian, Charles E Hughes, and Kenneth O Stanley (2010). "Evolving plastic neural networks with novelty search." In: *Adaptive Behavior* 18.6, pp. 470–491.

Srinivas, Niranjan et al. (2010). "Gaussian process optimization in the bandit setting: no regret and experimental design." In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 1015–1022.

Bull, Adam D (2011). "Convergence rates of efficient global optimization algorithms." In: *Journal of Machine Learning Research* 12.Oct, pp. 2879–2904.

Clune, J and H Lipson (2011). "Evolving three-dimensional objects with a generative encoding inspired by developmental biology." In: *ECAL*, pp. 141–148.

Das, Swagatam et al. (2011). "Real-parameter evolutionary multimodal optimization—A survey of the state-of-the-art." In: *Swarm and Evolutionary Computation* 1.2, pp. 71–88.

Garivier, Aurélien and Eric Moulines (2011). "On upper-confidence bound policies for switching bandit problems." In: *International Conference on Algorithmic Learning Theory*. Springer.

Gaspar-Cunha, António, Dirk Loyens, and Ferrie van Hattum (2011). "Aesthetic design using multi-objective evolutionary algorithms." In: *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, pp. 374–388.

Hornby, Gregory S, Jason D Lohn, and Derek S Linden (2011). "Computer-automated evolution of an X-band antenna for NASA's space technology 5 mission." In: *Evolutionary computation* 19.1, pp. 1–23.

Kistemaker, Steijn and Shimon Whiteson (2011). "Critical factors in the performance of novelty search." In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 965–972.

Lehman, J and K O Stanley (2011a). "Abandoning objectives: Evolution through the search for novelty alone." In: *Evolutionary computation* 19.2, pp. 189–223.

Lehman, J and K Stanley (2011b). "Evolving a diversity of virtual creatures through novelty search and local competition." In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation - GECCO '11*.

Mouret, Jean-Baptiste (2011). "Novelty-based multiobjectivization." In: *New horizons in evolutionary robotics*. Springer, pp. 139–154.

Schmidt, Michael and Hod Lipson (2011). "Age-fitness pareto optimization." In: *Genetic programming theory and practice VIII*. Springer, pp. 129–146.

Secretan, Jimmy et al. (2011). "Picbreeder: A case study in collaborative evolutionary exploration of design space." In: *Evolutionary Computation* 19.3, pp. 373–403.

Turrin, Michela, Peter Von Buelow, and Rudi Stouffs (2011). "Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms." In: *Advanced Engineering Informatics* 25.4, pp. 656–675.

Woolley, B G and K O Stanley (2011). "On the deleterious effects of a priori objectives on evolution and representation." In: *Proc. of GECCO*. ACM.

Gerber, David Jason et al. (2012). "Design optioneering: multi-disciplinary design optimization through parameterization, domain integration and automation of a genetic algorithm." In: *Proceedings of the 2012 Symposium on Simulation for Architecture and Urban Design*, pp. 1–8.

Sieger, Daniel, Stefan Menzel, and Mario Botsch (2012). "A comprehensive comparison of shape deformation methods in evolutionary design optimization." In: *Proceedings of the International Conference on Engineering Optimization*. Citeseer.

Snoek, J, H Larochelle, and R P Adams (2012). "Practical bayesian optimization of machine learning algorithms." In: *Proc. of NIPS*, pp. 2951–2959.

Arieff, Allison (2013). "New Forms that Function Better." In: *TECHNOLOGY REVIEW* 116.5, pp. 94–98.

Baranes, Adrien and Pierre-Yves Oudeyer (2013). "Active learning of inverse models with intrinsically motivated goal exploration in robots." In: *Robotics and Autonomous Systems* 61.1, pp. 49–73.

Clune, Jeff, Anthony Chen, and Hod Lipson (2013). "Upload any object and evolve it: Injecting complex geometric patterns into CPPNs for further evolution." In: *2013 IEEE Congress on Evolutionary Computation*. IEEE, pp. 3395–3402.

Cowen, Richard (2013). *History of life*. John Wiley & Sons.

Cully, Antoine and Jean-Baptiste Mouret (2013). "Behavioral repertoire learning in robotics." In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pp. 175–182.

Djolonga, Josip, Andreas Krause, and Volkan Cevher (2013). "High-dimensional gaussian process bandits." In: *Advances in Neural Information Processing Systems*, pp. 1025–1033.

Drela, M (2013). *XFOIL airfoil simulator*. raphael.mit.edu/xfoil/.

Druot, Thierry et al. (2013). "Multi-objective optimization of aircrafts family at conceptual design stage." In: *Inverse Problems, Design and Optimization Symposium. Albi, France*. Vol. 29, pp. 773–779.

Gomes, Jorge and Anders L Christensen (2013). "Generic behaviour similarity measures for evolutionary swarm robotics." In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pp. 199–206.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes." In: *arXiv preprint arXiv:1312.6114*.

Koos, Sylvain, Jean-Baptiste Mouret, and Stéphane Doncieux (2013). "The transferability approach: Crossing the reality gap in evolutionary robotics." In: *IEEE Transactions on Evolutionary Computation*.

Liapis, Antonios et al. (2013). "Transforming Exploratory Creativity with DeLeNoX." In: *Proceedings of the Fourth International Conference on Computational Creativity*, p. 56.

Martínez, Yuliana et al. (2013). "Searching for novel regression functions." In: *2013 IEEE congress on evolutionary computation*. IEEE, pp. 16–23.

Naredo, Enrique and Leonardo Trujillo (2013). "Searching for novel clustering programs." In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pp. 1093–1100.

Tsigkari, Martha et al. (2013). "Integrated design in the simulation process." In: *Proceedings of the Symposium on Simulation for Architecture & Urban Design*. Vol. 28. Society for Computer Simulation International.

Yang, Xin-She and Suash Deb (2013). "Multiobjective cuckoo search for design optimization." In: *Computers & Operations Research* 40.6, pp. 1616–1624.

Bradner, E, F Iorio, and M Davis (2014a). "Parameters tell the design story: Ideation and abstraction in design optimization." In: *Simulation Series*. ISSN: 07359276.

Bradner, Erin, Francesco Iorio, and Mark Davis (2014b). "Parameters tell the design story: ideation and abstraction in design optimization." In: *Proceedings of the symposium on simulation for architecture & urban design*. Vol. 26. Society for Computer Simulation International.

Cheney, Nick et al. (2014). "Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding." In: *ACM SIGEVOlution* 7.1, pp. 11–23.

Deb, Kalyanmoy (2014). "Multi-objective optimization." In: *Search methodologies*. Springer, pp. 403–449.

Gaier, Adam and Alexander Asteroth (2014a). "Evolution of optimal control for energy-efficient transport." In: *IEEE Intelligent Vehicles Symposium Proceedings*.

Gaier, Adam and Alexander Asteroth (2014b). "Evolving look ahead controllers for energy optimal driving and path planning." In: *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*. IEEE, pp. 138–145.

Goodfellow, Ian et al. (2014). "Generative adversarial nets." In: *Advances in neural information processing systems*, pp. 2672–2680.

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes." In: *International Conference on Learning Representation (ICLR)*. Ed. by Yoshua Bengio and Yann LeCun.

Richards, D and M Amos (2014). "Evolving morphologies with CPPN-NEAT and a dynamic substrate." In: *ALIFE 14: Proceedings of the fourteenth international conference on the synthesis and simulation of living systems*, pp. 255–262.

Simões, Luís F et al. (2014). "Self-adaptive genotype-phenotype maps: neural networks as a meta-representation." In: *International Conference on Parallel Problem Solving from Nature*. Springer.

Corucci, Francesco et al. (2015). "Novelty-based evolutionary design of morphing underwater robots." In: *Proceedings of the 2015 annual conference on Genetic and Evolutionary Computation*, pp. 145–152.

Cully, Antoine et al. (2015). "Robots that can adapt like animals." In: *Nature* 521.7553, pp. 503–507.

Doncieux, S et al. (2015a). "Evolutionary robotics: what, why, and where to." In: *Frontiers in Robotics and AI* 2, p. 4.

Doncieux, Stéphane et al. (2015b). "Multi-objective analysis of computational models." In: *arXiv preprint arXiv:1507.06877*.

Ecarlat, Pierre et al. (2015). "Learning a high diversity of object manipulations through an evolutionary-based babbling." In: *Proceedings of the Workshop Learning Object Affordances, IROS*, pp. 1–2.

Gaier, Adam (2015). "Evolutionary Design via Indirect Encoding of Non-Uniform Rational Basis Splines." In: *Proceedings of the Compan-*

ion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 1197–1200.

Gonzalez, Javier et al. (2015). "Bayesian optimization for synthetic gene design." In: *NIPS Workshop on Bayesian Optimization in Academia and Industry*.

Lemke, Christiane, Marcin Budka, and Bogdan Gabrys (2015). "Metalearning: a survey of trends and technologies." In: *Artificial intelligence review* 44.1, pp. 117–130.

Liapis, Antonios, Georgios N Yannakakis, and Julian Togelius (2015). "Constrained novelty search: A study on game content generation." In: *Evolutionary computation* 23.1, pp. 101–129.

Mouret, Jean-Baptiste and Jeff Clune (2015). "Illuminating search spaces by mapping elites." In: *arXiv preprint arXiv:1504.04909*.

Nguyen, A, J Yosinski, and J Clune (2015a). "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436.

Nguyen, Anh Mai, Jason Yosinski, and Jeff Clune (2015b). "Innovation engines: Automated creativity and improved stochastic optimization via deep learning." In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 959–966.

Overbury, Peter and Luc Berthouze (2015). "Using novelty-biased GA to sample diversity in graphs satisfying constraints." In: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 1445–1446.

O'Connor, JK and H Chang (2015). "Hindlimb feathers in paravians: Primarily "wings" or ornaments?" In: *Biology Bulletin* 42.7, pp. 616–621.

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

Schulman, John et al. (2015). "High-dimensional continuous control using generalized advantage estimation." In: *arXiv preprint arXiv:1506.02438*.

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

Shahriari, Bobak et al. (2015). "Taking the human out of the loop: A review of Bayesian optimization." In: *Proceedings of the IEEE*.

Stanley, Kenneth O and Joel Lehman (2015). *Why greatness cannot be planned: The myth of the objective*. Springer.

Andriani, Pierpaolo and Gino Cattani (2016). "Exaptation as source of creativity, innovation, and diversity: Introduction to the special section." In: *Industrial and Corporate Change* 25.1, pp. 115–131.

Benureau, Fabien CY and Pierre-Yves Oudeyer (2016). "Behavioral diversity generation in autonomous exploration through reuse of past experience." In: *Frontiers in Robotics and AI* 3, p. 8.

Brockman, G et al. (2016). *OpenAI Gym*. eprint: arXiv:1606.01540.

Calandra, R et al. (2016). "Bayesian optimization for learning gaits under uncertainty." In: *Annals of Mathematics and Artificial Intelligence* 76.1-2, pp. 5–23.

Chatzilygeroudis, Konstantinos, Antoine Cully, and Jean-Baptiste Mouret (2016). "Towards semi-episodic learning for robot damage recovery." In: *arXiv preprint arXiv:1610.01407*.

Coumans, E and Y Bai (2016–2018). *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org.

Cully, Antoine and J-B Mouret (2016). "Evolving a behavioral repertoire for a walking robot." In: *Evolutionary computation* 24.1, pp. 59–88.

Duarte, Miguel et al. (2016). "EvoRBC: Evolutionary repertoire-based control for robots with arbitrary locomotion complexity." In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 93–100.

Ha, David (2016). "Generating Large Images from Latent Vectors." In: *blog.otoro.net*. URL: http://blog.otoro.net/2016/04/01/generating-large-images-from-latent-vectors/.

Lehman, Joel, Sebastian Risi, and Jeff Clune (2016). "Creative generation of 3D objects with deep learning and innovation engines." In: *Proceedings of the 7th International Conference on Computational Creativity*.

Maynard, Andrew D (2016). "Are we ready for spray-on carbon nanotubes?" In: *Nature nanotechnology* 11.6, pp. 490–491.

Nguyen, Anh, Jason Yosinski, and Jeff Clune (2016). "Understanding innovation engines: Automated creativity and improved stochastic optimization via deep learning." In: *Evolutionary computation* 24.3, pp. 545–572.

Oord, Aaron van den et al. (2016). "Wavenet: A generative model for raw audio." In: *arXiv preprint arXiv:1609.03499*.

Papaspyros, Vaios et al. (2016). "Safety-aware robot damage recovery using constrained bayesian optimization and simulated priors." In: *arXiv preprint arXiv:1611.09419*.

Preen, Richard J and Larry Bull (2016). "Design mining interacting wind turbines." In: *Evolutionary computation* 24.1, pp. 89–111.

Pugh, Justin K, Lisa B Soros, and Kenneth O Stanley (2016). "Quality diversity: A new frontier for evolutionary computation." In: *Frontiers in Robotics and AI* 3, p. 40.

Rasmussen, C and H Nickisch (2016). *Gaussian Process Regression and Classification Toolbox*. http://www.gaussianprocess.org/gpml/code/.

Shahriari, B et al. (2016). "Taking the human out of the loop: A review of bayesian optimization." In: *Proceedings of the IEEE*.

Smith, Davy, Laurissa Tokarchuk, and Geraint Wiggins (2016). "Rapid phenotypic landscape exploration through hierarchical spatial partitioning." In: *International conference on parallel problem solving from nature*. Springer, pp. 911–920.

Wang, Ziyu et al. (2016). "Bayesian optimization in a billion dimensions via random embeddings." In: *Journal of Artificial Intelligence Research* 55, pp. 361–387.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein Generative Adversarial Networks." In: *International Conference on Machine Learning*, pp. 214–223.

Carter, Shan and Michael Nielsen (2017). "Using Artificial Intelligence to Augment Human Intelligence." In: *Distill*. https://distill.pub/2017/aia. DOI: 10.23915/distill.00009.

Chatzilygeroudis, K et al. (2017). "Black-Box Data-efficient Policy Search for Robotics." In: *Proc. of IROS*.

Conti, E et al. (2017). "Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents." In: *arXiv:1712.06560*.

Cully, Antoine and Yiannis Demiris (2017). "Quality and diversity optimization: A unifying modular framework." In: *IEEE Transactions on Evolutionary Computation* 22.2, pp. 245–259.

Gaier, A, A Asteroth, and J-B Mouret (2017a). "Aerodynamic Design Exploration through Surrogate-Assisted Illumination." In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*.

Gaier, A, A Asteroth, and J-B Mouret (2017b). "Data-efficient exploration, optimization, and modeling of diverse designs through surrogate-assisted illumination." In: *Proc. of GECCO*. ACM.

Gulrajani, Ishaan et al. (2017). "Improved training of wasserstein gans." In: *Advances in neural information processing systems*, pp. 5767–5777.

Higgins, Irina et al. (2017). "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." In:

International Human Powered Vehicle Association (2017). *IHPVA Official Speed Records*. (Visited on 11/27/2017).

Jin, Yanghua et al. (2017). "Towards the automatic anime characters creation with generative adversarial networks." In: *arXiv preprint arXiv:1708.05509*.

Kim, Seungsu and Stéphane Doncieux (2017). "Learning highly diverse robot throwing movements through quality diversity search." In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1177–1178.

Nagy, Danil et al. (2017). "Project Discover: An application of generative design for architectural space planning." In: *Proceedings of the Symposium on Simulation for Architecture and Urban Design*. Society for Computer Simulation International, p. 7.

OpenFOAM Foundation (2017). *OpenFOAM Computational Fluid Dynamics toolbox*. www.openfoam.org/.

Salimans, T et al. (2017). "Evolution strategies as a scalable alternative to reinforcement learning." In: *arXiv preprint arXiv:1703.03864*.

Such, F P et al. (2017). "Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning." In: *arXiv:1712.06567*.

Vassiliades, Vassilis, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret (2017). "Using centroidal voronoi tessellations to scale up the multi-dimensional archive of phenotypic elites algorithm." In: *IEEE Transactions on Evolutionary Computation*.

Vaswani, Ashish et al. (2017). "Attention is all you need." In: *Advances in neural information processing systems*, pp. 5998–6008.

World Human Powered Vehicle Association (2017). *Records - Land – World Human Powered Vehicle Association*. (Visited on 11/27/2017).

Zhang, X, J Clune, and K O Stanley (2017). "On the Relationship Between the OpenAI Evolution Strategy and Stochastic Gradient Descent." In: *arXiv:1712.06564*.

Bontrager, Philip et al. (2018a). "Deep interactive evolution." In: *International Conference on Computational Intelligence in Music, Sound, Art and Design*. Springer.

Bontrager, Philip et al. (2018b). "Deepmasterprints: Generating masterprints for dictionary attacks via latent variable evolution." In: *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*. IEEE, pp. 1–9.

Burgess, Christopher P et al. (2018). "Understanding disentangling in beta-VAE." In:

Chatzilygeroudis, Konstantinos, Vassilis Vassiliades, and Jean-Baptiste Mouret (2018). "Reset-free trial-and-error learning for robot damage recovery." In: *Robotics and Autonomous Systems* 100, pp. 236–250.

Chen, Tian Qi et al. (2018). "Isolating sources of disentanglement in variational autoencoders." In: *Advances in Neural Information Processing Systems*.

Co-Reyes, John D et al. (2018). "Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings." In: *Proceedings of the International Conference on Machine Learning (ICML)*.

Conti, Edoardo et al. (2018). "Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents." In: *Advances in neural information processing systems*, pp. 5027–5038.

Cully, Antoine and Yiannis Demiris (2018). "Hierarchical behavioral repertoires with unsupervised descriptors." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 69–76.

Engebråten, Sondre A et al. (2018). "Evolving a Repertoire of Controllers for a Multi-function Swarm." In: *International Conference on the Applications of Evolutionary Computation*. Springer, pp. 734–749.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2018a). "Data-efficient design exploration through surrogate-assisted illumination." In: MIT Press.

Gaier, Adam, Alexander Asteroth, and Jean-Baptiste Mouret (2018b). "Data-efficient neuroevolution with kernel-based surrogate models." In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, pp. 85–92.

Hagg, Alexander, Alexander Asteroth, and Thomas Bäck (2018). "Prototype discovery using quality-diversity." In: *International Conference on Parallel Problem Solving from Nature*. Springer, pp. 500–511.

Khalifa, Ahmed et al. (2018). "Talakat: Bullet hell generation through constrained map-elites." In: *Proceedings of The Genetic and Evolutionary Computation Conference*, pp. 1047–1054.

Kim, Hyunjik and Andriy Mnih (2018). "Disentangling by Factorising." In: *International Conference on Machine Learning*.

Kingma, Durk P and Prafulla Dhariwal (2018). "Glow: Generative flow with invertible 1x1 convolutions." In: *Advances in Neural Information Processing Systems*, pp. 10215–10224.

Lehman, J and K O Stanley (2018). "On the Potential Benefits of Knowing Everything." In: pp. 558–565. DOI: 10.1162/isal_a_00104.

Lehman, Joel et al. (2018). "The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities." In:

Matejka, Justin et al. (2018). "Dream lens: Exploration and visualization of large-scale generative design datasets." In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–12.

Moreno, Matthew Andres, Wolfgang Banzhaf, and Charles Ofria (2018). "Learning an evolvable genotype-phenotype mapping." In: *Genetic and Evolutionary Computation Conference*. ACM.

Nguyen, Su, Mengjie Zhang, and Kay Chen Tan (2018). "Adaptive charting genetic programming for dynamic flexible job shop scheduling." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1159–1166.

Nordmoen, Jørgen, Kai Olav Ellefsen, and Kyrre Glette (2018a). "Combining MAP-elites and incremental evolution to generate gaits for a mammalian quadruped robot." In: *International Conference on the Applications of Evolutionary Computation*. Springer, pp. 719–733.

Nordmoen, Jørgen et al. (2018b). "Dynamic mutation in MAP-Elites for robotic repertoire generation." In: *Artificial Life Conference Proceedings*. MIT Press, pp. 598–605.

Pautrat, Rémi, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret (2018). "Bayesian optimization with automatic prior selection for data-efficient direct policy search." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7571–7578.

Rieffel, John and Jean-Baptiste Mouret (2018). "Adaptive and resilient soft tensegrity robots." In: *Soft robotics* 5.3, pp. 318–329.

Scott, Eric O and Kenneth A De Jong (2018). "Toward learning neural network encodings for continuous optimization problems." In: *Genetic and Evolutionary Computation Conference Companion*. ACM.

Segre, Paolo S and Amanda I Banet (2018). "The origin of avian flight: finding common ground." In: *Biological Journal of the Linnean Society* 125.2, pp. 452–454.

Stork, Jörg, Agoston E Eiben, and Thomas Bartz-Beielstein (2018). "A new taxonomy of continuous global optimization algorithms." In: *arXiv preprint arXiv:1808.08818*.

Urquhart, Neil and Emma Hart (2018). "Optimisation and illumination of a real-world workforce scheduling and routing application (WSRP) via Map-Elites." In: *International Conference on Parallel Problem Solving from Nature*. Springer, pp. 488–499.

Vassiliades, Vassiiis and Jean-Baptiste Mouret (2018). "Discovering the elite hypervolume by leveraging interspecies correlation." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 149–156.

Volz, Vanessa et al. (2018). "Evolving mario levels in the latent space of a deep convolutional generative adversarial network." In: *Genetic and Evolutionary Computation Conference*. ACM.

Alvarez, Alberto et al. (2019). "Empowering quality diversity in dungeon design with interactive constrained MAP-Elites." In: *2019 IEEE Conference on Games (CoG)*. IEEE, pp. 1–8.

Antoran, Javier and Antonio Miguel (2019). "Disentangling and Learning Robust Representations with Natural Clustering." In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, pp. 694–699.

Bruneton, J-P et al. (2019). "Exploration and Exploitation in Symbolic Regression using Quality-Diversity and Evolutionary Strategies Algorithms." In: *arXiv preprint arXiv:1906.03959*.

Canaan, Rodrigo et al. (2019). "Diverse agents for ad-hoc cooperation in hanabi." In: *2019 IEEE Conference on Games (CoG)*. IEEE, pp. 1–8.

Cazenille, Leo, Nicolas Bredeche, and Nathanael Aubert-Kato (2019). "Using map-elites to optimize self-assembling behaviors in a swarm of bio-micro-robots." In: *BioRxiv*, p. 845594.

Cully, Antoine (2019). "Autonomous skill discovery with quality-diversity and unsupervised descriptors." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 81–89.

Dolson, Emily, Alexander Lalejini, and Charles Ofria (2019). "Exploring genetic programming systems with MAP-Elites." In: *Genetic Programming Theory and Practice XVI*. Springer, pp. 1–16.

Doncieux, Stephane, Alban Laflaquière, and Alexandre Coninx (2019). "Novelty search: a theoretical perspective." In: *Genetic and Evolutionary Computation Conference, GECCO'19*, pp. 99–106. DOI: 10.1145/3321707.3321752.

Ecoffet, Adrien et al. (2019). "Go-explore: a new approach for hard-exploration problems." In: *arXiv preprint arXiv:1901.10995*.

Ekern, Erlend (2019). *This Picture Does Not Exist*. URL: https://thispicturedoesnotexist.com/ (visited on 12/30/2019).

Elsken, Thomas, Jan Hendrik Metzen, and Frank Hutter (2019). "Neural Architecture Search: A Survey." In:

Fontaine, Matthew C et al. (2019a). "Covariance Matrix Adaptation for the Rapid Illumination of Behavior Space." In:

Fontaine, Matthew C et al. (2019b). "Mapping hearthstone deck spaces through MAP-elites with sliding boundaries." In: *Proceedings of The Genetic and Evolutionary Computation Conference*.

Gaier, Adam and David Ha (2019). "Weight agnostic neural networks." In: *Advances in Neural Information Processing Systems*.

Giacomello, Edoardo, Pier Luca Lanzi, and Daniele Loiacono (2019). "Searching the Latent Space of a Generative Adversarial Network to Generate DOOM Levels." In: *2019 IEEE Conference on Games (CoG)*. IEEE, pp. 1–8.

Gravina, Daniele et al. (2019). "Procedural content generation through quality diversity." In: *2019 IEEE Conference on Games (CoG)*. IEEE, pp. 1–8.

Hagg, Alexander et al. (2019). "Prediction of neural network performance by phenotypic modeling." In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1576–1582.

Hoyer, Stephan, Jascha Sohl-Dickstein, and Sam Greydanus (2019). "Neural reparameterization improves structural optimization." In:

Jegorova, Marija, Stéphane Doncieux, and Timothy M Hospedales (2019). "Behavioral Repertoire via Generative Adversarial Policy Networks." In: *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*. IEEE, pp. 320–326.

Justesen, Niels et al. (2019). "Learning a Behavioral Repertoire from Demonstrations." In: *arXiv preprint arXiv:1907.03046*.

Karras, Tero, Samuli Laine, and Timo Aila (2019a). "A style-based generator architecture for generative adversarial networks." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410.

Karras, Tero et al. (2019b). "Analyzing and improving the image quality of stylegan." In: *arXiv preprint arXiv:1912.04958*.

Khalifa, Ahmed et al. (2019). "Intentional computational level design." In: *Proceedings of The Genetic and Evolutionary Computation Conference*, pp. 796–803.

Mesentier Silva, Fernando de et al. (2019). "Evolving the hearthstone meta." In: *2019 IEEE Conference on Games (CoG)*. IEEE, pp. 1–8.

Miikkulainen, Risto et al. (2019). "Evolving deep neural networks." In: *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier.

Stork, Jörg et al. (2019). "Surrogate models for enhancing the efficiency of neuroevolution in reinforcement learning." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 934–942.

Zaltron, Nicola, Luisa Zurlo, and Sebastian Risi (2019). "CG-GAN: An Interactive Evolutionary GAN-based Approach for Facial Composite Generation." In: *arXiv preprint arXiv:1912.05020*.

Charity, Megan et al. (2020). "Mech-Elites: Illuminating the Mechanic Space of GVGAI." In: *arXiv preprint arXiv:2002.04733*.

Colas, Cédric et al. (2020). *Scaling MAP-Elites to Deep Neuroevolution*. arXiv: 2003.01825 [cs.NE].

Fontaine, Matthew C et al. (2020). "Illuminating Mario Scenes in the Latent Space of a Generative Adversarial Network." In: *arXiv preprint arXiv:2007.05674*.

Generative Design at Airbus: Customer Stories (2020). URL: https://www.autodesk.com/customer-stories/airbus#.

Kaushik, Rituraj, Pierre Desreumaux, and Jean-Baptiste Mouret (2020a). "Adaptive Prior Selection for Repertoire-based Online Adaptation in Robotics." In: *Frontiers in Robotics and AI* 6, p. 151.

Kaushik, Rituraj, Timothée Anne, and Jean-Baptiste Mouret (2020b). "Fast Online Adaptation in Robotics through Meta-Learning Embeddings of Simulated Priors." In: *arXiv preprint arXiv:2003.04663*.

Le, Vivian (2020). *Their Dark Materials*. URL: https://99percentinvisible.org/episode/their-dark-materials/.

Schrum, Jacob, Vanessa Volz, and Sebastian Risi (2020a). "CPPN2GAN: Combining Compositional Pattern Producing Networks and GANs for Large-scale Pattern Generation." In: *arXiv preprint arXiv:2004.01703*.

Schrum, Jacob et al. (2020b). "Interactive Evolution and Exploration Within Latent Level-Design Space of Generative Adversarial Networks." In: *arXiv preprint arXiv:2004.00151*.

Urquhart, Neil, Emma Hart, and William Hutcheson (2020). "Using MAP-Elites to support policy making around Workforce Scheduling and Routing." In: *at-Automatisierungstechnik* 68.2, pp. 110–117.

Wang, Philip (2020). *This Person Does Not Exist*. URL: https://www.thispersondoesnotexist.com/.

Wikipedia (2020). *Irish elk — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Irish%20elk&oldid=956283220. [Online; accessed 19-May-2020].