

Covariance Matrix Adaptation for the Rapid Illumination of Behavior Space

Matthew C. Fontaine

Viterbi School of Engineering
University of Southern California
Los Angeles, CA
mfontain@usc.edu

Stefanos Nikolaidis

Viterbi School of Engineering
University of Southern California
Los Angeles, CA
nikolaid@usc.edu

ABSTRACT

We focus on the challenge of finding a diverse collection of quality solutions on complex continuous domains. While quality diversity (QD) algorithms like Novelty Search with Local Competition (NSLC) and MAP-Elites are designed to generate a diverse range of solutions, these algorithms require a large number of evaluations for exploration of continuous spaces. Meanwhile, variants of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) are among the best-performing derivative-free optimizers in single-objective continuous domains. This paper proposes a new QD algorithm called Covariance Matrix Adaptation MAP-Elites (CMA-ME). Our new algorithm combines the self-adaptation techniques of CMA-ES with archiving and mapping techniques for maintaining diversity in QD. Results from experiments based on standard continuous optimization benchmarks show that CMA-ME finds better-quality solutions than MAP-Elites; similarly, results on the strategic game Hearthstone show that CMA-ME finds both a higher overall quality and broader diversity of strategies than both CMA-ES and MAP-Elites. Overall, CMA-ME more than doubles the performance of MAP-Elites using standard QD performance metrics. These results suggest that QD algorithms augmented by operators from state-of-the-art optimization algorithms can yield high-performing methods for simultaneously exploring and optimizing continuous search spaces, with significant applications to design, testing, and reinforcement learning among other domains.

KEYWORDS

Quality diversity, illumination algorithms, evolutionary algorithms, Hearthstone, optimization, MAP-Elites

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

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

© 2020 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.

ACM ISBN 978-1-4503-7128-5/20/07...\$15.00
<https://doi.org/10.1145/3377930.3390232>

Julian Togelius

Tandon School of Engineering
New York University
New York City, NY
julian@togelius.com

Amy K. Hoover

Ying Wu College of Computing
New Jersey Institute of Technology
Newark, NJ
ahoover@njit.edu

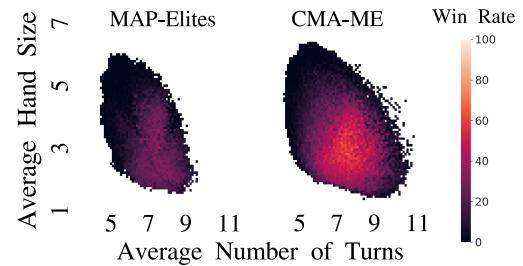


Figure 1: Comparing Hearthstone Archives. Sample archives for both MAP-Elites and CMA-ME from the Hearthstone experiment. Our new method, CMA-ME, both fills more cells in behavior space and finds higher quality policies to play Hearthstone than MAP-Elites. Each grid cell is an elite (high performing policy) and the intensity value represent the win rate across 200 games against difficult opponents.

ACM Reference Format:

Matthew C. Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K. Hoover. 2020. Covariance Matrix Adaptation for the Rapid Illumination of Behavior Space. In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Cancún, Mexico*. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3377930.3390232>

1 INTRODUCTION

We focus on the challenge of finding a diverse collection of quality solutions on complex continuous domains. Consider the example application of generating strategies for a turn-based strategy game (e.g., chess, Go, Hearthstone). What makes these games appealing to human players is not the presence of an optimal strategy, but the variety of fundamentally different viable strategies. For example, “aggressive” strategies aim to end the game early through high-risk high-reward play, while “controlling” strategies delay the end of the game by postponing the targeting of the game objectives [11]. These example strategies vary in one aspect of their *behavior*, measured by the number of turns the game lasts.

Finding fundamentally different strategies requires navigating a continuous domain, such as the parameter space (weights) of a

neural network that maps states to actions, while simultaneously exploring a diverse range of behaviors. Quality Diversity algorithms, such as Novelty Search with Local Competition (NSLC) [34] and Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [8], drive a divergent search for multiple good solutions, rather than a convergent search towards a single optimum, through sophisticated archiving and mapping techniques. Solutions are binned in an archive based on their behavior and compete only with others exhibiting similar behaviors. Such stratified competition results in the discovery of potentially sub-optimal solutions called *stepping stones*, which have been shown in some domains to be critical for escaping local optima [19, 33]. However, these algorithms typically require a large number of evaluations to achieve good results.

Meanwhile, when seeking a single global optimum, variants of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [23, 27] are among the best-performing derivative-free optimizers, with the capability to rapidly navigate continuous spaces. However, the self-adaptation and cumulation techniques driving CMA-ES have yet to successfully power a QD algorithm.

We propose a new hybrid algorithm called Covariance Matrix Adaptation MAP-Elites (CMA-ME), which rapidly navigates and optimizes a continuous space with CMA-ES seeded by solutions stored in the MAP-Elites archive. The hybrid algorithm employs CMA-ES's ability to efficiently navigate continuous search spaces by maintaining a mixture of normal distributions (candidate solutions) dynamically augmented by objective function feedback. The key insight of CMA-ME is to *leverage the selection and adaptation rules of CMA-ES to optimize good solutions, while also efficiently exploring new areas of the search space*.

Building on the underlying structure of MAP-Elites, CMA-ME maintains a population of modified CMA-ES instances called *emitters*, which like CMA-ES operate based on a sampling mean, covariance matrix, and adaptation rules that specify how the covariance matrix and mean are updated for each underlying normal distribution. The population of emitters can therefore be thought of as a Gaussian mixture where each normal distribution focuses on improving a different area of behavior space. This paper explores three types of candidate emitters with different selection and adaptation rules for balancing quality and diversity, called the *random direction*, *improvement*, and *optimizing* emitters. Solutions generated by the emitters are saved in a single unified archive based on their corresponding behaviors.

We evaluate CMA-ME through two experiments: a toy domain designed to highlight current limitations of QD in continuous spaces and a practical turn-based strategy game domain, Hearthstone, which mirrors a common application of QD: finding diverse agent policies.¹ Hearthstone is an unsolved, partially observable game that poses significant challenges to current AI methods [28]. Overall, the results of both experiments suggest CMA-ME is a competitive alternative to MAP-Elites for exploring continuous domains (Fig. 1). The potential for improving QD's growing number of applications is significant as our approach greatly reduces the computation time required to generate a diverse collection of high-quality solutions.

¹Code is available for both the continuous optimization benchmark and Hearthstone domains [16, 17]

2 BACKGROUND

This section outlines previous advancements in quality diversity (QD) including one of the first QD algorithms, MAP-Elites, and background in CMA-ES to provide context for the CMA-ME algorithm proposed in this paper.

2.1 Quality Diversity (QD)

QD algorithms are often applied in domains where a diversity of good but meaningfully different solutions is valued. For example QD algorithms can build large repertoires of robot behaviors [7, 9, 10] or a diversity of locomotive gaits to help robots quickly respond to damage [8]. By interacting with AI agents, QD can also produce a diversity of generated video game levels [1, 22, 31].

While traditional evolutionary algorithms speciate based on encoding and fitness, a key feature of the precursor to QD (e.g., Novelty Search (NS) [32]) is speciation through behavioral diversity [32]. Rather than optimizing for performance relative to a fitness function, searching directly for behavioral diversity promotes the discovery of sub-optimal solutions relative to the objective function. Called *stepping stones*, these solutions mitigate premature convergence to local optima. To promote intra-niche competition [40] objectives were reintroduced in the QD algorithms Novelty Search with Local Competition (NSLC) [34] and MAP-Elites [8].

While NSLC and MAP-Elites share many key features necessary for maintaining a diversity of quality solutions, a core difference between the two is whether the archive is dynamically or statically generated. NSLC dynamically creates behavior niches by growing an archive of sufficiently novel solutions while MAP-Elites (detailed in the next section) maintains a static mapping of behavior. For CMA-ME we choose MAP-Elites as our diversity mechanism to directly compare benefits inherited from CMA-ES. Though we make this design choice for the CMA-ME algorithm solely for comparability, the same principles can be applied to the NSLC archive.

2.2 MAP-Elites

While one core difference between two early QD algorithms NSLC [34] and MAP-Elites [8] is whether behaviors are dynamically or statically mapped, another is the number of behavioral dimensions among which solutions typically vary. Rather than defining a single distance measure to characterize and differentiate behaviors, MAP-Elites often searches along at least two measures called behavior characteristics (BCs) that induce a Cartesian space (called a *behavior space*). This behavior space is then tessellated into uniformly spaced grid cells, where the goal of the algorithm is to 1) maximize the number of grid cells containing solutions and 2) maximize the quality of the best solution within each grid cell. Modifications and improvements to MAP-Elites often focus on the tessellation of behavior space [18, 43, 45].

However, this paper proposes improvements to MAP-Elites based on the generation of solutions rather than the tessellation of behavior space. At the start of the MAP-Elites algorithm, the archive (map) is initialized randomly by solutions sampled uniformly from the search space. Each cell of the map contains at most one solution (i.e., an *elite*), which is the highest performing solution in that behavioral niche. New solutions are generated by taking an elite (selected uniformly at random) and perturbing it with Gaussian

noise. MAP-Elites computes a behavior vector for each new solution and assigns the new solution to a cell in the map. The solution replaces the elite in its respective cell if the new solution has higher fitness, or the new solution simply fills the cell if the cell is empty.

2.3 CMA-ES

Evolution strategies (ES) are a family of evolutionary algorithms that specialize in optimizing continuous spaces by sampling a population of solutions, called a generation, and gradually moving the population toward areas of highest fitness. One canonical type of ES is the $(\mu/\mu, \lambda)$ -ES, where a population of λ sample solutions is generated, then the fittest μ solutions are selected to generate new samples in the next generation. The $(\mu/\mu, \lambda)$ -ES recombines the μ best samples through a weighted average into one mean that represents the center of the population distribution of the next generation. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a particular type of this canonical ES, which is one of the most competitive derivative-free optimizers for single-objective optimization of continuous spaces [25].

CMA-ES models the sampling distribution of the population as a multivariate normal distribution $\mathcal{N}(m, C)$ where m is the distribution mean and C is its covariance matrix. The main mechanisms steering CMA-ES are the selection and ranking of the μ fittest solutions, which update the next generation's next sampling distribution, $\mathcal{N}(m, C)$. CMA-ES maintains a history of aggregate changes to m called an evolution path, which provides benefits to search that are similar to momentum in stochastic gradient descent.

2.4 Related Work

It is important to note that QD methods differ both from diversity maintenance methods and from multi-objective optimization algorithms, in that QD methods search for solutions that exhibit different *behaviors*, which in our strategy game example would be number of turns, rather than searching for diversity in parameter space (neural network weights that induce a strategy). For example, consider the case of two different sets of network weights exhibiting similar play styles. A diversity maintenance method, such as niching or speciation would consider them different species, while QD would treat them as similar with respect to the exhibited behavior, forcing intra-niche competition. Several versions of CMA-ES exist that incorporate niching or speciation [39, 42].

Multi-objective search could also be applied to our strategy game. By treating a behavior characteristic (game length) as an additional objective, we aim to maximize or minimize the average game length in addition to maximizing win rate. However, without any insight into our strategy game, it is unclear whether we should maximize or minimize game length. Quality diversity algorithms differ from multi-objective search by seeking solutions across the whole spectrum of this measure, rather than only at the extremes.

Several previous works explored incorporating ideas from a simplified ES. For example Conti et al. [5] introduced novelty seeking to a $(\mu/\mu, \lambda)$ -ES. However, their ES does not leverage adaptation and perturbs solutions through static multivariate Gaussian noise. Nordmoen et al. [38] dynamically mutate solutions in MAP-Elites, but globally adapt σ (mutation power) for all search space variables

rather than adapting the covariances between search space variables. Vassiliades and Mouret [46] exploited correlations between elites in MAP-Elites, proposing a variation operator that accelerates the MAP-Elites algorithm. Their approach exploits covariances between elites rather than drawing insights from CMA-ES to model successful evolution steps as a covariance matrix.

3 APPROACH: THE CMA-ME ALGORITHM

Through advanced mechanisms like step-size adaptation and evolution paths, CMA-ES can quickly converge to a single optimum. The potential for CMA-ES to refine the best solutions discovered by MAP-Elites is clear. However, the key insight making CMA-ME possible is that by repurposing these mechanisms from CMA-ES we can improve the *exploration* capabilities of MAP-Elites. Notably, CMA-ES can efficiently both *expand* and contract the search distribution to explore larger regions of the search space and stretch the normal distribution within the search space to find hard to reach nooks and crannies within the behavior space. In CMA-ME we create a population of modified CMA-ES instances called *emitters* that perform search with feedback gained from interacting with the archive.

At a high-level, CMA-ME is a scheduling algorithm for the population of emitters. Solutions are generated in search space in a round-robin fashion, where each emitter generates the same number of solutions (see Alg. 1). The solutions are generated in the same way for all emitters by sampling from the distribution $\mathcal{N}(m, C)$ (see `generate_solution` in Alg. 1). The procedure `return_solution` is specific to each type of emitter used by CMA-ME and is responsible for adapting the sampling distribution and maintaining the sampled population.

Algorithm 1: Covariance Matrix Adaptation MAP-Elites

```

CMA-ME (evaluate,  $n$ )
  input : An evaluation function evaluate which computes
         a behavior characterization and fitness, and a
         desired number of solutions  $n$ .
  result : Generate  $n$  solutions storing elites in a map  $M$ .
  Initialize population of emitters  $E$ 
  for  $i \leftarrow 1$  to  $n$  do
    Select emitter  $e$  from  $E$  which has generated the least
    solutions out of all emitters in  $E$ 
     $x_i \leftarrow \text{generate\_solution}(e)$ 
     $\beta_i, \text{fitness} \leftarrow \text{evaluate}(x_i)$ 
     $\text{return\_solution}(e, x_i, \beta_i, \text{fitness})$ 
  end

```

3.1 CMA-ME Emitters

To understand how emitters differ from CMA-ES instances, consider that the covariance matrix in CMA-ES models a distribution of possible search directions within the search space. The distribution captures the most likely direction of the next evolution step where fitness increase will be observed. Unlike estimation of multivariate normal algorithms (EMNAs) that increase the likelihood of reobserving the previous best individuals (see Figure 3 of Hansen [23]),

CMA-ES increases the likelihood of successful future evolution steps (steps that increase fitness). Emitters differ from CMA-ES by adjusting the ranking rules that form the covariance matrix update to maximize the likelihood that future steps in a given direction result in archive improvements. However, there are many ways to rank, leading to many possible types of emitters.

We propose three types of emitters: optimizing, random direction, and improvement. Like CMA-ES, described in Section 2.3, each emitter maintains a sampling mean m , a covariance matrix C , and a parameter set P that contains additional CMA-ES related parameters (e.g., evolution path). However, while CMA-ES restarts its search based on the best current solution, emitters are differentiated by their rules for restarting and adapting the sampling distribution, as well as for selecting and ranking solutions.

We explore *optimizing emitters* to answer the question: are restarts alone enough to promote good exploration in CMA-ME as they are in multi-modal methods? An optimizing emitter is almost identical to CMA-ES, differing only in that restart means are chosen from the location of an elite rather than the fittest solution discovered so far. The random direction and improvement emitters are described below in more detail.

To intuitively understand the *random direction* emitters, imagine trying to solve the classic QD maze domain in the dark [32, 33]. Starting from an initial position in the behavior space, random direction emitters travel in a given direction until hitting a wall, at which point they restart search and move in a new direction. While good solutions to the maze and other low-dimensional behavior spaces can be found by random walks, the black-box nature of the forward mapping from search space to behavior makes the inverse mapping equally opaque. Random direction emitters are designed to estimate the inverse mapping of this correspondence problem.

When a random direction emitter restarts, it emulates a step in a random walk by selecting a random direction or bias vector v_β to move toward in behavior space. To build a covariance matrix such that it biases in direction v_β at each generation, solutions in search space are mapped to behavior space (β_i). The mean (m_β) of all λ solutions is calculated in behavior space and each direction with respect to the mean calculated. Only solutions that improve the archive are then ranked by their projection value against the line $m_\beta + v_\beta t$. If none of these solutions improve the archive, the emitter restarts from a randomly chosen elite with a new bias vector v_β .

Interestingly because random emitters choose the bias vector or direction in behavior space to move toward at the beginning of a restart, it necessarily ignores areas of high fitness that it finds along the way. Instead, while exploring *improvement emitters* exploit the areas of high fitness by ranking solutions based on the improvement or change in fitness within each niche. When determining the amount of improvement, solutions filling empty cells are prioritized over those replacing existing solutions in their niche by ranking them higher in the covariance matrix update. Rather than exploring a fixed direction for the duration of the run, the advantage of improvement emitters is that they fluidly adjust their goals based on where progress is currently being made.

Algorithm 2 shows the implementation of `return_solution` from algorithm 1 for an improvement emitter. Each solution x_i that has been generated by the emitter maps to a behavior β_i and a cell

$M[\beta_i]$ in the map. If the cell is empty (line 2), or if x_i has higher fitness than the existing solution in the cell (line 6), x_i is added to the new generation's *parents* and the map is updated. The process repeats until the generation of x_i s reaches size λ (line 9), where we adapt the emitter: If we have found parents that improved the map, we rank them (line 11) by concatenating two groups: first the group of parents that discovered new cells in the map, sorted by their fitness, and second the group of parents that improved existing cells, sorted by the increase in fitness over the previous solution that occupied that cell. If we have not found any solutions that improve the map, we restart the emitter (line 15).

Algorithm 2: An improvement emitter's `return_solution`.

```

return_solution ( $e, x_i, \beta_i, fitness$ )
  input : An improvement emitter  $e$ , evaluated solution  $x_i$ ,
         behavior vector  $\beta_i$ , and fitness.
  result : The shared archive  $M$  is updated by solution  $x_i$ . If
          $\lambda$  individuals have been generated since the last
         adaptation, adapt the sampling distribution
          $N(m, C)$  of  $e$  towards the behavioral regions of
         largest improvement.

1   Unpack the parents, sampling mean  $m$ , covariance matrix
     $C$ , and parameter set  $P$  from  $e$ .
2   if  $M[\beta_i]$  is empty then
3      $\Delta_i \leftarrow fitness$ 
4     Flag that  $x_i$  discovered a new cell
5     Add  $x_i$  to parents
6   else if  $x_i$  improves  $M[\beta_i]$  then
7      $\Delta_i \leftarrow fitness - M[\beta_i].fitness$ 
8     Add  $x_i$  to parents
9   end
10  if sampled population is size  $\lambda$  then
11    if parents  $\neq \emptyset$  then
12      Sort parents by (newCell,  $\Delta_i$ )
13      Update  $m$ ,  $C$ , and  $P$  by parents
14      parents  $\leftarrow \emptyset$ 
15    else
16      | Restart from random elite in  $M$ 
17    end
18  end

```

4 TOY DOMAIN

Previous work on applying QD to Hearthstone deck search [18] observed highly distorted behavior spaces. However, to our knowledge, no previous work involving standard benchmark domains observes distortions in behavior space. The goal of the toy domain is to create the simplest domain with high degrees of behavior space distortions. Surprisingly, a linear projection from a high-dimensional search space to a low-dimensional behavior space highly distorts the distribution of solutions in behavior space. This section details experiments in our toy domain designed to measure the effects of distortions on MAP-Elites. We hypothesize the most likely benefit of CMA-ME is the ability to overcome distortions in

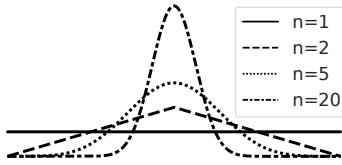


Figure 2: A Bates distribution demonstrating the narrowing property of behavior spaces formed by a linear projection.

behavior space and this experiment measures the benefits of an adaptable QD algorithm over using a fixed mutation power.

4.1 Distorted Behavior Spaces

Previous work in QD measures the effects of different types of behavior characteristics (BCs) on QD performance, such as BC alignment with the objective [15, 40, 41]. We highlight a different limiting effect on performance: the distortion caused by dimensionality reduction from search space to behavior space. While any dimensionality reduction can increase the difficulty of covering behavior space, we demonstrate that exploring the behavior space formed from a simple linear projection from a high-dimensional search space results in significant difficulty in standard MAP-Elites.

Specifically in the case of linear projection, each BC depends on every parameter of the search space. In the case when all projection coefficients are equal, each parameter contributes equally to the corresponding BC. By being equally dependent on each parameter, a QD algorithm needs to navigate every parameter to reach extremes in the behavior space instead of only a subset of the parameters.

This is shown by uniformly sampling from the search space and projecting the samples to behavior vectors, where each component is the sum of n uniform random variables. When divided by n (to normalize the BC to the range $[0, 1]$), the sampling results in the Bates distribution shown in Fig. 2. As the dimensions of the search space grow, the distribution of the behavior space narrows making it harder to find behaviors in the tails of the distribution.

We hypothesize that the adaptation mechanisms of CMA-ME will better cover this behavior space when compared to MAP-Elites, since CMA-ME can adapt each parameter with a separate variance, rather than with a fixed global mutation rate. Additionally, the final goal of this experiment is to explore the performance of these algorithms in a distributed setting, therefore we choose parameters that allow for parallelization of the evaluation.

4.2 Experiments

To show the benefit of covariance matrix adaptation in CMA-ME, we compare the performance of MAP-Elites, CMA-ES, and CMA-ME when performing dimensionality reduction from the search space to behavior space. We additionally compare against the recently proposed line-mutation operator for MAP-Elites [46], which we call ME (line) [17]. As QD algorithms require a solution quality measure, we include two functions from the continuous black-box optimization set of benchmarks [24, 25] as objectives.

Our two objective functions are of the form $f : \mathbb{R}^n \rightarrow \mathbb{R}$: a sphere shown in Eq. 1 and the Rastrigin function shown in Eq. 2.

The optimal fitness of 0 in these functions is obtained by $x_i = 0$. To avoid having the optimal value at the center of the search space, we offset the fitness function so that, without loss of generality, the optimal location is $x_i = 5.12 \cdot 0.4 = 2.048$ (note that $[-5.12, 5.12]$ is the typical valid domain of the Rastrigin function).

$$\text{sphere}(x) = \sum_{i=1}^n x_i^2 \quad (1)$$

$$\text{rastrigin}(x) = 10n + \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i)] \quad (2)$$

Behavior characteristics are formed by a linear projection from \mathbb{R}^n to \mathbb{R}^2 , and the behavior space is bounded through a *clip* function (Eq. 3) that restricts the contribution of each component x_i to the range $[-5.12, 5.12]$ (the typical domain of the constrained Rastrigin function). To ensure that the behavior space is equally dependent on each component of the search space (i.e., \mathbb{R}^n), we assign equal weights to each component. The function $p : \mathbb{R}^n \rightarrow \mathbb{R}^2$ formalizes the projection from the search space \mathbb{R}^n to the behavior space \mathbb{R}^2 (see Eq. 4), by computing the sum of the first half of components from \mathbb{R}^n and the sum of the second half of components from \mathbb{R}^n .

$$\text{clip}(x_i) = \begin{cases} x_i & \text{if } -5.12 \leq x_i \leq 5.12 \\ 5.12/x_i & \text{otherwise} \end{cases} \quad (3)$$

$$p(x) = \left(\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \text{clip}(x_i), \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^n \text{clip}(x_i) \right) \quad (4)$$

We compare MAP-Elites, ME (line) CMA-ES, and CMA-ME on the toy domain, running CMA-ME three times, once for each emitter type. We run each algorithm for 2.5M evaluations. We set $\lambda = 500$ for CMA-ES, whereas a single run of CMA-ME deploys 15 emitters of the same type with $\lambda = 37$ ². The map resolution is 500×500 .

Algorithms are compared by the QD-score metric proposed by previous work [41], which in MAP-Elites and CMA-ME is the sum of fitness values of all elites in the map. To compute the QD-score of CMA-ES, solutions are assigned a grid location on what their BC would have been and populate a pseudo-archive. Since QD-score assumes maximizing test functions with non-negative values, fitness is normalized to the range $[0, 100]$, where 100 is optimal. Since we can analytically compute the boundaries of the space from the linear projection, we also show the percentage of possible cells occupied by each algorithm.

MAP-Elites perturbs solutions with Gaussian noise scaled by a factor σ named mutation power. Previous work [13, 37] shows that varying σ can greatly affect both the precision and coverage of MAP-Elites. To account for this and obtain the best performance of MAP-Elites on the toy domain, we ran a grid search to measure MAP-Elites performance across 101 values of σ uniformly distributed across $[0.1, 1.1]$, and we selected the value with the best coverage, $\sigma = 0.5$, for all the experiments. Since CMA-ES and CMA-ME adapt their sampling distribution, we did not tune any parameters, but we set the initial value of their mutation power also to $\sigma = 0.5$.

²Both the Toy and Hearthstone domains use the same λ and emitter count.

Table 1: Sphere Function Results

| Algorithm | $n = 20$ | | | $n = 100$ | | |
|--------------|-------------|----------------|-------------------|-------------|----------------|-------------------|
| | Max Fitness | Cells Occupied | QD-Score | Max Fitness | Cells Occupied | QD-Score |
| CMA-ES | 100 | 3.46 % | 731,613 | 100 | 3.74 % | 725,013 |
| MAP-Elites | 99.596 | 56.22 % | 11,386,641 | 96.153 | 26.97 % | 5,578,919 |
| ME (line) | 99.920 | 68.99 % | 13,714,405 | 98.021 | 31.75 % | 6,691,582 |
| CMA-ME (opt) | 100 | 12.53 % | 2,573,157 | 100 | 2.70 % | 654,649 |
| CMA-ME (rd) | 98.092 | 90.32 % | 13,651,537 | 96.731 | 77.12 % | 13,465,879 |
| CMA-ME (imp) | 99.932 | 87.75 % | 16,875,583 | 99.597 | 61.98 % | 12,542,848 |

Table 2: Rastrigin Function Results

| Algorithm | $n = 20$ | | | $n = 100$ | | |
|--------------|---------------|----------------|-------------------|---------------|----------------|-------------------|
| | Max Fitness | Cells Occupied | QD-Score | Max Fitness | Cells Occupied | QD-Score |
| CMA-ES | 99.982 | 4.17 % | 818,090 | 99.886 | 3.64 % | 660,037 |
| MAP-Elites | 90.673 | 55.70 % | 9,340,327 | 81.089 | 26.51 % | 4,388,839 |
| ME (line) | 92.700 | 58.25 % | 9,891,199 | 84.855 | 27.72 % | 4,835,294 |
| CMA-ME (opt) | 99.559 | 8.63 % | 1,865,910 | 98.159 | 3.23 % | 676,999 |
| CMA-ME (rd) | 91.084 | 87.74 % | 10,229,537 | 90.801 | 74.13 % | 10,130,091 |
| CMA-ME (imp) | 96.358 | 83.42 % | 14,156,185 | 86.876 | 60.72 % | 9,804,991 |

4.3 Results

We label CMA-ME (opt), CMA-ME (rd), and CMA-ME (imp) for the optimizing, random direction, and improvement emitters, respectively. Tables 1 and 2 show the results of the sphere and rastrigin function experiments. CMA-ES outperforms all other algorithms in obtaining the optimal fitness. As predicted, covering the behavior space becomes harder as the dimensions of the search space grow. CMA-ME (rd) and CMA-ME (imp) obtain the highest QD-Scores and fill the largest number of unique cells in the behavior space. This is because of the ability of the CMA-ME emitters to efficiently discover new cells. Notably, CMA-ME (opt) fails to keep up with CMA-ES, performing worse in maximum fitness. We find this result consistent with the literature on multi-modal CMA-ES [26], since CMA-ES moves with a single large population that has global optimization properties, while CMA-ME (opt) has a collection of smaller populations with similar behavior.

Fig. 3(c) and Fig. 3(d) show the distribution of elites by their fitness. CMA-ME (rd) achieves a high QD-score by retaining a large number of relatively low quality solutions, while CMA-ME (imp) fills slightly less cells but of higher quality. In the Hearthstone experiment (Section 5) we use improvement emitters, since we are more interested in elites of *high quality* than the number of cells filled in the resulting collection.

5 HEARTHSTONE DOMAIN

We aim to discover a collection of high quality strategies for playing the Hearthstone game against a set of fixed opponents. We wish to explore *how differently* the game can be played rather than just *how well* the game can be played. We selected this domain, since its behavior space has distortions similar to those described in section 4 and observed in previous work [18], which make exploration of the behavior space particularly challenging.

Hearthstone [14] is a two-player, turn-taking adversarial online collectable card game that is an increasingly popular domain for evaluating both classical AI techniques and modern deep reinforcement learning approaches due to the many unique challenges it poses (e.g., large branching factor, partial observability, stochastic actions, and difficulty with planning under uncertainty) [28]. In Hearthstone players construct a deck of exactly thirty cards that players place on a board shared with their opponent. The game's objective is to reduce your opponent's health to zero by attacking your opponent directly using cards in play. Decks are constrained by one of nine possible hero classes, where each hero class can access different cards and abilities.

Rather than manipulating the reward function of individual agents in a QD system [2, 36] (like the QD approach in AlphaStar [47]), generating the best gameplay strategy or deck [4, 20, 44], or searching for different decks with a fixed strategy [18], our experiments search for a diversity of strategies for playing an expert-level human-constructed deck.

5.1 Experiments

This section details the Hearthstone simulator, agent policy and deck, and our opponents' policies and decks.

SabberStone Simulator: SabberStone [12] is a Hearthstone simulator that replicates the rules of Hearthstone and uses the card definitions publicly provided by Blizzard. In addition to simulating the game, SabberStone includes a turn-local game tree search that searches possible action sequences that can be taken at a given turn. To implement different strategies, users can create a heuristic scoring function that evaluates the state of the game. Included with the simulator are standard “aggro and control” card-game strategies which we use in our opponent agents.

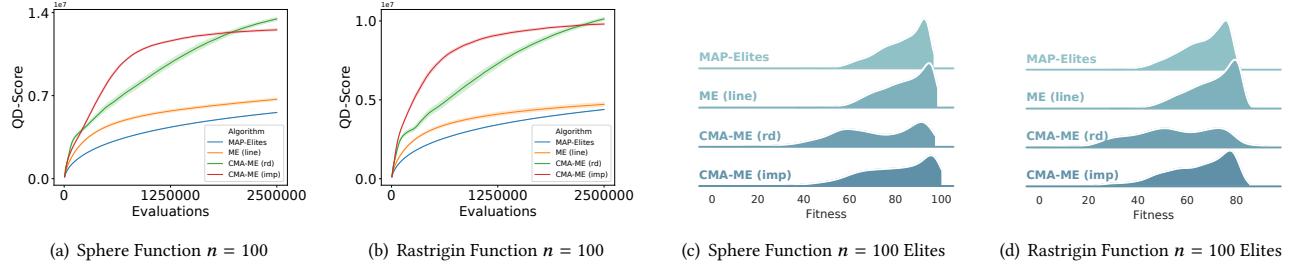


Figure 3: Toy Domain Results. (a-b) Improvement in QD-Score over evaluations. (c-d) The distribution of elites scaled by the number of occupied cells to show the relative makeup of elites within each archive.

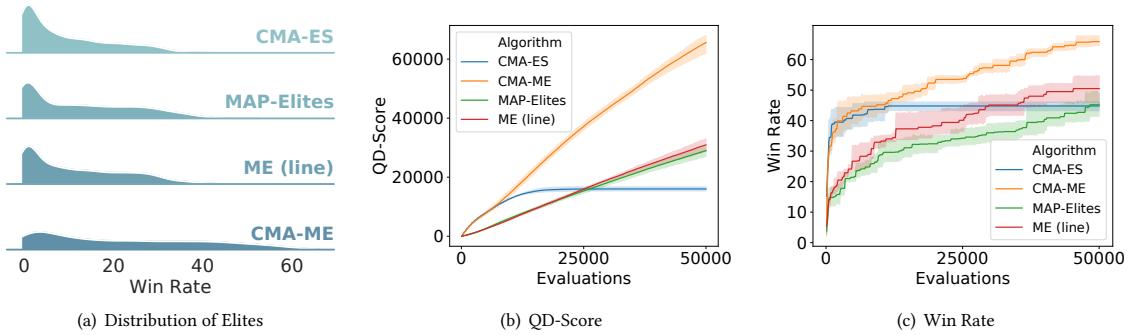


Figure 4: Hearthstone Results (a) The distribution of elites by win rate. Each distribution is scaled by the number of occupied cells to show the relative makeup of elites within each archive. (b) Improvement in QD-Score over evaluations. (c) Improvement in win rate over evaluations.

Our Deck: In Hearthstone there are subsets of cards that can only be used by specific “classes” of players. We selected the class Rogue, where “cheap” cards playable in the beginning of the game can be valuable later on. Successful play with the Rogue class requires long-term planning with sparse rewards. To our knowledge, our work is the first to create a policy to play the Rogue class. We selected the Tempo Rogue archetype from the Hearthstone expansion *Rise of Shadows*, which is a hard deck preferred by advanced players. While many variants of the Tempo Rogue archetype exist, we decided to use the decklist from Hearthstone grandmaster Fei “ETC” Liang who reached the number 1 ranking with his list in May 2019 [35].

Opponents: Fontaine et al. [18] used game tree search to find decks and associated policies. They found six high performing decks for the Paladin, Warlock, and Hunter classes playing aggro and control strategies; we use these as our opponent suite.

Neural Network: We search for the parameters of a neural network that scores an observable game state based on the cards played on the board and card-specific features. The network maps 15 evaluation features defined by Decoster et al. [12] to a scalar score. Cuccu et al. [6] show that a six-neuron neural network trained with a natural evolution strategies (NES) [21], can obtain competitive and sometimes state-of-the-art solutions on the Atari Learning Environment (ALE) [3]. They separate feature extraction from decision-making, using vector quantization for feature extraction

and the six-neuron network for the decision making. Motivated by this work, we use a 26 node fully connected feed-forward neural network with layer sizes [15, 5, 4, 1] (109 parameters).

5.2 Search Parameters and Tuning:

We use the fitness function proposed in [4, 18]: the average health difference between players at the end of the game, as a smooth approximation of win rate. For MAP-Elites and CMA-ME, we characterize behavior by the average hand size per turn and the average number of turns the game lasts. We choose these behavioral characteristics to capture a spectrum of strategies between aggro decks, which try to end the game quickly, and control decks that attempt to extend the game. The hand size dimension measures the ability of a strategy to generate new cards.

To tune MAP-Elites, we ran three experiments with σ values of 0.05, 0.3, and 0.8. MAP-Elites achieved the best coverage and maximum win rate performance with $\sigma = 0.05$ and we used that as our mutation power. Our archive was resolution 100×100 , where we set the range of behavior values using data from the Hearthstone player data corpus [29]. As with the Toy Domain in section 4, CMA-ES and CMA-ME used the same hyperparameters as MAP-Elites. For CMA-ME we ran all experiments using improvement emitters, since their performance had the most desirable performance attributes in the toy domain.

Table 3: Hearthstone Results

| Algorithm | Maximum Overall | | All Solutions | |
|------------|-----------------|---------------|----------------|-----------------|
| | Fitness | Win Rate | Cells Filled | QD-Score |
| CMA-ES | -2.471 | 44.8 % | 17.02 % | 16,024.8 |
| MAP-Elites | -2.859 | 45.2 % | 21.72 % | 25,936.0 |
| ME (line) | -0.252 | 50.5 % | 22.30 % | 28,132.7 |
| CMA-ME | 5.417 | 65.9 % | 29.17 % | 63,295.6 |

5.3 Distributed Evaluation

We ran our experiments on a high-performance cluster with 500 (8 core) CPUs in a distributed setting, with a master search node and 499 worker nodes. Each worker node is responsible for evaluating a single policy at a time and plays 200 games against our opponent suite. A single experimental trial evaluating 50,000 policies takes 12 hours. MAP-Elites and ME (line) are run asynchronously on the master search node, while CMA-ME and CMA-ES synchronize after each generation. We ran each algorithm for 5 trials and generated 50,000 candidate solutions per trial.

5.4 Results

Table 3 shows that CMA-ME outperforms both MAP-Elites and CMA-ES and in maximum fitness, maximum win rate, the number of cells filled, and QD-Score. The distribution of elites for all three algorithms show that CMA-ME finds elites in higher performing parts of the behavior space than both CMA-ES and MAP-Elites (see Fig. 4(a)). The sample archive for CMA-ME and MAP-Elites in Fig. 1 similarly illustrates that CMA-ME better covers the behavior space and finds higher quality policies than MAP-Elites and ME (line).

Fig. 4(b) shows the increase in quality diversity over time, with CMA-ME more than doubling the QD-Score of MAP-Elites. Fig. 4(c) shows the increase in win rate over time. CMA-ME maintains a higher win rate than CMA-ES, MAP-Elites and ME (line) at all stages of evaluation. CMA-ES quickly converges to a single solution but is surpassed by MAP-Elites later in the evaluation.

6 DISCUSSION

Both the toy domain and the Hearthstone domain share a property of the behavior space that challenges exploration with standard MAP-Elites: some areas of the behavior space are hard to reach with random sampling even without the presence of deception. Particularly in the toy domain, discovering solutions in the tails of the behavior space formed by the Bates distribution described in Section 4.1 requires MAP-Elites to adapt its search distribution. While CMA-ES finds *individual* solutions that outperform both QD algorithms, in all of the chosen domains it covers significantly less area of the behavior space than any QD method.

Because CMA-ME covers (e.g., explores) more of the behavior space than MAP-Elites in the Hearthstone domain, even though the mapping is likely non-linear and non-separable, effects similar to those shown with the Bates distribution may be preventing MAP-Elites from discovering policies at the tails of behavior space like the ends of the “average number of turns” dimension (horizontal axis in Fig. 1). However, some behavioral characteristics like the “average hand size” are fairly equally covered by the two algorithms.

Therefore, when selecting an algorithm, if maximal coverage is desired we recommend CMA-ME in part for its ability to efficiently explore a distorted behavior space.

While CMA-ES is often the algorithm of choice for solving problems in continuous optimization, Fig. 4(c) shows in Hearthstone that CMA-ME and MAP-Elites better optimize for fitness (i.e., win rate). Although interestingly in the first half of the search, Fig. 4(b) shows that CMA-ES has a higher QD-score than MAP-Elites, demonstrating its potential for exploration. While in the second half of search this exploration is not sustained as CMA-ES begins to converge, it is possible that in this domain the objective function leads CMA-ES into a *deceptive trap* [33]. The best strategies discovered early are likely aggro, and that by discovering these strategies early in the search, CMA-ES consistently converges to the best aggro strategy instead of learning more complex control strategies.

While there are many possible emitter types, the CMA-ME algorithm proposes and explores three: optimizing, random direction, and improvement emitters. Because of their limited map interaction and ranking purely by fitness, optimizing emitters tend to explore a smaller area of behavior space. Random direction emitters alone can cover more of the behavior space, but solution quality is higher when searching with improvement emitters (Fig. 1). Random emitters additionally require a direction in behavior space, which may be impossible for certain domains [30]. If diversity is more important than quality, we recommend random direction emitters, but alternatively recommend improvement emitters if defining a random direction vector is challenging or if the application prioritizes quality over diversity. We have only explored homogeneous emitter populations; we leave experimenting with heterogeneous populations of emitters as an exciting topic for future work.

7 CONCLUSIONS

We presented a new algorithm called CMA-ME that combines the strengths of CMA-ES and MAP-Elites. Results from both a toy domain and a policy search in Hearthstone show that leveraging strengths from CMA-ES can improve both the coverage and quality of MAP-Elites solutions. Results from Hearthstone additionally show that when compared to standard CMA-ES, the diversity components from MAP-Elites can improve the quality of solutions in continuous search spaces. Overall, CMA-ME improves MAP-Elites by bringing modern optimization methods to quality-diversity problems for the first time. By reconceptualizing optimization problems as quality-diversity problems, results from CMA-ME suggest new opportunities for research and deployment, including any approach that learns policies encoded as neural networks; complementing the objective function with behavioral characteristics can yield not only a useful diversity of behavior, but also better performance on the original objective.

ACKNOWLEDGMENTS

The authors would like to thank Heramb Nemlekar and Mark Nelson for their feedback on a preliminary version of this paper.

REFERENCES

- [1] Alberto Alvarez, Steve Dahlskog, Jose Font, and Julian Togelius. 2019. Empowering Quality Diversity in Dungeon Design with Interactive Constrained MAP-Elites. *IEEE Conference on Games (CoG)*.

- [2] Kai Arulkumaran, Antoine Cully, and Julian Togelius. 2019. AlphaStar: An Evolutionary Computation Perspective. In *GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Manuel López-Ibáñez (Ed.). ACM, New York, NY, USA.
- [3] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47 (2013), 253–279.
- [4] Aditya Bhatt, Scott Lee, Fernando de Mesentier Silva, Connor W. Watson, Julian Togelius, and Amy K. Hoover. 2018. Exploring the Hearthstone Deck Space. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*. ACM, 18.
- [5] Edoardo Conti, Vashishth Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2018. Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems (NIPS'18)*. Curran Associates Inc., USA, 5032–5043. <http://dl.acm.org/citation.cfm?id=3327345.3327410>
- [6] Giuseppe Cuccu, Julian Togelius, and Philippe Cudré-Mauroux. 2019. Playing atari with six neurons. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 998–1006.
- [7] Antoine Cully. 2019. Autonomous Skill Discovery with Quality-Diversity and Unsupervised Descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)*. ACM, 81–89.
- [8] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015. Robots that can adapt like animals. *Nature* 521, 7553 (2015), 503.
- [9] Antoine Cully and Jean-Baptiste Mouret. 2013. Behavioral Repertoire Learning in Robotics. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*. ACM, 175–182.
- [10] Antoine Cully and Jean-Baptiste Mouret. 2016. Evolving a Behavioral Repertoire for a Walking Robot. *Evolutionary Computation* 24 (2016), 59–88, Issue 1.
- [11] Fernando de Mesentier Silva, Rodrigo Canaan, Scott Lee, Matthew C Fontaine, Julian Togelius, and Amy K Hoover. 2019. Evolving the hearthstone meta. In *2019 IEEE Conference on Games (Cog)*. IEEE, 1–8.
- [12] Cedric Decoster, Jean Seong Bjorn Choe, et al. 2019. *Sabberstone*. <https://github.com/HearthSim/SabberStone> Accessed: 2019-11-01.
- [13] A. E. Eiben, R. Hinterding, and Z. Michalewicz. 1999. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 3, 2 (July 1999), 124–141. <https://doi.org/10.1109/4235.771166>
- [14] Blizzard Entertainment. [n. d.]. Hearthstone. <https://playhearthstone.com/en-us/>. ([n. d.]). Accessed: 2019-11-01.
- [15] Stefano Fioravanti and Giovanni Iacca. 2019. Evaluating MAP-Elites on Constrained Optimization Problems. (2019). arXiv:1902.00703
- [16] Matthew Fontaine. 2019. *EvoStone*. <https://github.com/tehqin/EvoStone> Accessed: 2019-12-01.
- [17] Matthew Fontaine. 2019. *QualDivBenchmark*. <https://github.com/tehqin/QualDivBenchmark> Accessed: 2019-12-01.
- [18] Matthew C. Fontaine, Scott Lee, L. B. Soros, Fernando de Mesentier Silva, Julian Togelius, and Amy K. Hoover. 2019. Mapping Hearthstone Deck Spaces through MAP-Elites with Sliding Boundaries. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)*. ACM, New York, NY, USA, 161–169. <https://doi.org/10.1145/3321707.3321794>
- [19] Adam Gaier, 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*. ACM, 115–116.
- [20] Pablo García-Sánchez, Alberto Tonda, Antonio J Fernández-Leiva, and Carlos Cotta. 2019. Optimizing Hearthstone agents using an evolutionary algorithm. *Knowledge-Based Systems* (2019), 105032.
- [21] Tobias Glasmachers, Tom Schaul, Sun Yi, Daan Wierstra, and Jürgen Schmidhuber. 2010. Exponential natural evolution strategies. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 393–400.
- [22] Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius, and Georgios Yannakakis. 2019. Procedural Content Generation through Quality Diversity. *IEEE Conference on Games (CoG)*, 1–8. <https://doi.org/10.1109/CIG.2019.8848053>
- [23] Nikolaus Hansen. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772* (2016).
- [24] N. Hansen, A. Auger, O. Mersmann, T. Tusař, and D. Brockhoff. 2016. COCO: A platform for comparing continuous optimizers in a black-box setting. (2016).
- [25] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošepán. 2010. Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009. *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10 - Companion Publication*, 1689–1696. <https://doi.org/10.1145/1830761.1830790>
- [26] Nikolaus Hansen and Stefan Kern. 2004. Evaluating the CMA evolution strategy on multimodal test functions. In *International Conference on Parallel Problem Solving from Nature*. Springer, 282–291.
- [27] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation* 9, 2 (2001), 159–195.
- [28] Amy K Hoover, Julian Togelius, Scott Lee, and Fernando de Mesentier Silva. 2019. The Many AI Challenges of Hearthstone. *KI-Künstliche Intelligenz* (2019), 1–11.
- [29] HSReplay. [n. d.]. HSReplay. <https://hsreplay.net/>. ([n. d.]). Accessed: 2019-11-01.
- [30] Ahmed Khalifa, Michael Cerny Green, Gabriella Barros, and Julian Togelius. 2019. Intentional computational level design. In *Proceedings of The Genetic and Evolutionary Computation Conference*. 796–803.
- [31] Ahmed Khalifa, Scott Lee, Andy Nealen, and Julian Togelius. 2018. Talakat: Bullet Hell Generation Through Constrained Map-elites. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18)*. ACM, New York, NY, USA, 1047–1054. <https://doi.org/10.1145/3205455.3205470>
- [32] Joel Lehman and Kenneth O. Stanley. 2008. Exploiting Open-Endedness to Solve Problems through the Search for Novelty. In *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*. 329–336.
- [33] Joel Lehman and Kenneth O. Stanley. 2011. Abandoning Objectives: Evolution through the Search for Novelty Alone. *Evolutionary Computation* (2011).
- [34] Joel Lehman and Kenneth O. Stanley. 2011. 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)*. ACM, 211–218.
- [35] Fei Liang. [n. d.]. Tempo Rogue. <https://www.hearthstonetopdecks.com/decks/tempo-rogue-rise-of-shadows-1-legend-etc/>. ([n. d.]). Accessed: 2019-11-01.
- [36] Andrew Y. Ng, Daisaku Harada, and Stuart J. Russell. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML '99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 278–287. <http://dl.acm.org/citation.cfm?id=645528.657613>
- [37] Jørgen Nordmoen, Kai Olav Ellefsen, and Kyrré Glette. 2018. Combining MAP-Elites and Incremental Evolution to Generate Gaits for a Mammalian Quadruped Robot. 719–733. https://doi.org/10.1007/978-3-319-77538-8_48
- [38] Jørgen Nordmoen, Eivind Samuelsen, Kai Olav Ellefsen, and Kyrré Glette. 2018. Dynamic Mutation in MAP-Elites for Robotic Repertoire Generation. In *Artificial Life Conference Proceedings*. MIT Press, 598–605. https://doi.org/10.1162/isal_a_00110
- [39] Mike Preuss. 2012. Improved Topological Niching for Real-valued Global Optimization. In *Proceedings of the 2012T European Conference on Applications of Evolutionary Computation (EvoApplications'12)*. Springer-Verlag, Berlin, Heidelberg, 386–395. https://doi.org/10.1007/978-3-642-29178-4_39
- [40] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. 2016. Quality Diversity: A New Frontier for Evolutionary Computation. *Frontiers in Robotics and AI* 3 (2016), 40. <https://doi.org/10.3389/frobt.2016.00040>
- [41] Justin K. Pugh, L. B. Soros, Paul A. Szerlip, and Kenneth O. Stanley. 2015. Confronting the Challenge of Quality Diversity. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*. ACM, New York, NY, USA, 967–974. <https://doi.org/10.1145/2739480.2754664>
- [42] Ofer M. Shir and Thomas Bäck. 2006. Niche Radius Adaptation in the CMA-ES Niching Algorithm. In *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN'06)*. Springer-Verlag, Berlin, Heidelberg, 142–151. https://doi.org/10.1007/11844297_15
- [43] Davy Smith, Laurissa Tokarchuk, and Geraint Wiggins. 2016. Rapid phenotypic landscape exploration through hierarchical spatial partitioning. In *International conference on parallel problem solving from nature*. Springer, 911–920.
- [44] Maciej Świechowski, Tomasz Tajmajar, and Andrzej Janusz. 2018. Improving Hearthstone AI by Combining MCTS and Supervised Learning Algorithms. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.
- [45] Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. 2018. Using Centroidal Voronoi Tessellations to Scale Up the Multidimensional Archive of Phenotypic Elites Algorithm. *IEEE Transactions on Evolutionary Computation* 22, 4 (2018), 623–630.
- [46] Vassilis Vassiliades and Jean-Baptiste Mouret. 2018. Discovering the elite hyper-volume by leveraging interspecies correlation. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 149–156.
- [47] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 11 (2019), 350–354.