Behavioral search drivers and the role of elitism in soft robotics

William La Cava and Jason Moore
University of Pennsylvania, Philadelphia, PA 19104
lacava@upenn.edu

Abstract

Behavioral search drivers allow more information about the behavior of individuals in an environment to be used during selection. In this paper, we examine several selection methods based on de-aggregating the motion of soft robots into behavior vectors used to drive search. We adapt three behavioral search drivers to this task: ϵ-lexicase selection, discovery of objectives by clustering, and novelty search. These methods are compared to age-fitness pareto optimization and random search. We analyze how these search drivers affect the diversity and quality of soft robots that are tasked with moving as far of a distance as possible. Perhaps the most surprising finding is that random search with elitism is competitive with previously published methods. Overall, we find that elitism plays an important role in the ability to find high fitness solutions, and that lexicase selection and discovery of objectives by clustering with elitism tend to produce the most fit solutions.

Introduction

Living organisms interact with their environments in myriad ways that affect their chances of selection and survival. Factors that influence their outcomes vary with space, time, and as a function of their own behavior and morphology. Despite this complexity, it is very common in evolutionary computation (EC) to assign scalar fitness values to individuals to determine their selection/survival probabilities. Such a simplification is welcomed, of course, if it is an adequate surrogate for producing the types of results and/or solutions a researcher is concerned with producing, but the assumptions are important to note.

By collapsing all of a candidate's behavior into a single value, scalar fitness assignments may lose information regarding the differential performance of individuals for specific scenarios. For example, imagine an individual exhibits behavior in a particular part of the environment that is perferable to the behavior of the rest of the population in similar scenarios. If this behavior is not reflected in the average of the individual's behavior over all scenarios, this potentially useful semantic/behavioral "building block" can be lost during evolution. The evolutionary dynamics of scalar fitnesses were central to the unexpected findings in an early analysis of genetic algorithms (GAs) (Mitchell et al., 1994), which found that aggregate fitness scores could mask the ability of GAs to leverage building blocks, even in the ideal conditions created by the royal road functions.

Others have pointed out the shortcoming of scalar fitness values (Spector, 2012; Krawiec and Liskowski, 2015) and attempts to address it have led the emergence of so-called *behavioral search drivers*, also known as semantic methods. Although motivations for their development vary, a central motivation for most behavioral search drivers is their ability to leverage more information about each individual during the selection process. Many of these techniques have developed in the genetic programming (GP) community. In this paper, we analyze a handful of recent methods for their ability to evolve soft robotic creatures. The use of behavioral search drivers may be well-suited to the soft robotics domain for the following reasons: 1) robots must interact with heterogenous environments, where challenges to suvival may vary spatially; 2) robots have complex morphologies that expand and contract with time, leading to complex behaviors that may contain important subsets of information useful to determining their survival.

Below, we briefly review previous work in evolving soft robots. In the Methods section, we describe the five algorithms compared in our study. In our experiment, we benchmark the ability of these search drivers to evolve locomotion in soft robots. We then perform a detailed analysis of the resulting creatures.

Previous Work

To simulate soft-tissue robots, Hiller and Lipson (2014) developed the Voxelyze and VoxCad simulation tools, in which objects are composed of 3D "pixels", i.e. voxels, that have various material properties. The physics simulator provides a 3-dimensional physical world with static and dynamic multibody physics. The initial attempts to evolve robots using simple representations yielded mixed, but promising, results (Hiller and Lipson, 2012). Following these initial experiments, Cheney et al. (2013) proposed a new representation scheme that allowed greater evolvability. Robots were constructed from a pre-defined palette of four materials representing bone, muscles and ligaments. The authors proposed a developmental solution to the construction of more regular and symmetrical morphologies by sampling a compositional pattern producing network (CPPN) (Stanley, 2007) to construct forms. Subsequent work has moved towards a multi-objective approach, using Pareto optimization to incorporate complexity, energy usage, and age into guide the evolutionary process (Cheney et al., 2015; Kriegman et al., 2017). The results have produced seemingly life-like gaits and behaviors[1] reminiscent of early work by Sims (1994).

Several studies have built off of the results of Cheney et al. (2013). Different environmental settings have been studied, including aquatic environments (Corucci et al., 2016), different gravitational environments (Methenitis et al., 2015), and constricted spaces (Cheney et al., 2015). New approaches to encoding the robots using a developmental lifespan have shown promise (Kriegman et al., 2017). Of particular relevance to this paper has been the proposal to use novelty search to guide evolution. Methenitis et al. (2015) found that novelty search, especially when paired with an elitist strategy, could find morphologies that travelled farther distances than those found with using distance travelled as fitness, although the differences were less dramatic than those afforded by the indirect encoding.

In contrast to previous studies, the central focus of our paper is to analyze selection methods used to evolve soft robots. We introduce two behavioral selection methods that were developed for GP: discovery of objectives by clustering (Krawiec and Liskowski, 2015) and ϵ-lexicase selection (La Cava et al., 2016). These two methods differ from novelty search in that they treat each behavioral observation as a *test* that directly reflects the ability of a candidate solution to solve a portion of the overall problem at hand. Nevertheless, we consider novelty search a behavioral search driver as well, due to its use of a set of behaviors to determine preference for selection.

The behavioral search drivers we consider here are selection methods, although semantic variation methods are also actively researched in GP; see Vanneschi et al. (2014) for a review. The semantic selection methods we study have been compared for classification tasks in Liskowski et al. (2015) and symbolic regression tasks in Liskowski and Krawiec (2017). The regression and classification problem domains are quite different, in that the "tests", i.e. behaviors, can be mapped directly to samples from the training set. In the following section, we introduce a behavioral definition to de-aggregate robot behavior in its environment in a way that is reflective of progress towards the objective.

[1]https://youtu.be/EXuR_soDnFo

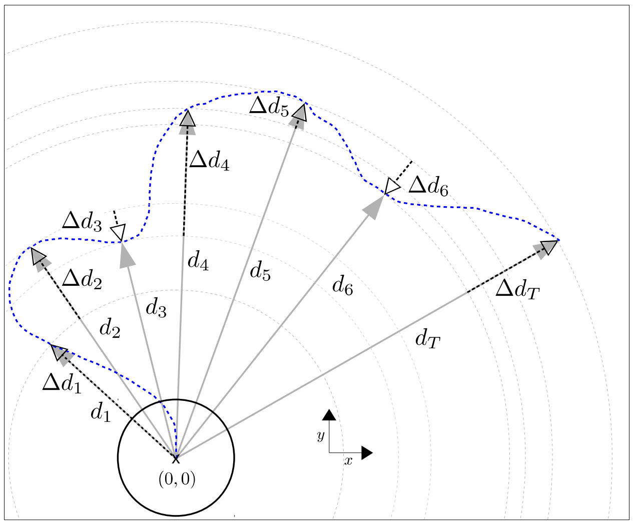

Figure 1: De-aggregated behavior definition. The dotted blue line represents an example robot trajectory. d_t (gray arrow) is the robot's distance from the origin at time step t. Δd_t (black dotted arrow) is the change in distance from the origin at time step t.

Methods

In the following sections we describe how the behavior of each robot is defined. We then describe the different selection methods tested and offer some insight into their mechanics.

Aside from the methods for selection and the behavior definitions, we follow the soft robot implementations used previously. The genotype of the robots are defined using a CPPN representation, evolved using framework known as neuro-evolution of augmenting topologies (NEAT) (Stanley, 2007). The CPPN encoding produces three outputs that are interpreted to decide 1) whether or not to place a voxel in a given location, 2) whether the voxel will be soft or stiff, and 3) whether the voxel will be active (muscle) or passive. Prior work has highlighted the improvements in evolvability and regularity of the morphologies brought about by this encoding (Cheney et al., 2015; Methenitis et al., 2015). Our implementation of these methods is built off of an open-source implementation[2], and is available here[3].

Behavior Definition

In accordance with previous work, we focus on the task of robot exploration by rewarding robots for travelling far from their initial location during simulation. The position of our robot's center of mass is defined by the tuple $(x_t,\ y_t,\ z_t)$, with $(x_0 = 0, y_0 = 0, z_0 = 0)$ corresponding to its start-

[2]https://github.com/skriegman/evosoro
[3]http://github.com/lacava/evosoro

ing position. At time step t, the body lengths travelled by a robot, d_t is defined as the square root of the norm of its planar motion:

$$d_t = \sqrt{(x_t^2 + y_t^2)}/L \tag{1}$$

Here L is the lattice dimension of the robot. Previous works (Kriegman et al., 2017; Cheney et al., 2015) have used d_t in addition to other fitness penalties. Complexity is penalized based on the number of voxels, v. Energy expenditure is penalized based on the number of active voxels, denoted e. Finally, age (a) has been used an objective in an age-fitness Pareto optimization scheme (Schmidt and Lipson, 2011). In combination with random restarts each generation, the age variable generally improves diversity and prevents premature convergence.

In order to apply behavioral search drivers, we require a set of behaviors that are reflective of preference for selection. We will refer to this set of behaviors as $\mathcal{F} = \{f_1(n), \ldots, f_m(n), n \in \mathcal{N}\}$, where $f_i(n)$ is the ith behavior of robot n in population $\mathcal{N}$.

Whereas previous work has defined behavior according to many different measures (Methenitis et al., 2015), we adopt a more simplistic approach and focus just on the planar motion of the robot at each time step. Assume our simulation consists of T timesteps $0 \ldots t \ldots T$. Then the distance covered by a robot at each time step can be defined as follows:

$$\Delta d_{t+1} = d_{t+1} - d_t \tag{2}$$

This behavior definition is shown in Figure 1. In addition to distances travelled at each time step, we include the aggregate scores for energy, number of voxels and age as behaviors. For the purposes of our algorithm definitions, we will assume all behaviors are *minimized*, leading to the following behavior definition:

$$\mathcal{F} = \{-\Delta d_1, -\Delta d_t, \ldots, -\Delta d_T\} \cup \{v\} \cup \{e\} \cup \{a\} \tag{3}$$

This behavioral definition is used for all the behavioral search drivers.

Search Drivers

We describe the five search drivers compared in this study in this section. For every experiment, these algorithms are applied to the set of individuals consisting of the population and its offspring, represented by $\mathcal{N}$. The number of selections each generation, ns, is therefore half the size of $\mathcal{N}$.

Pareto Optimization As a starting point, we compare to the age-fitness pareto optimization (AFP) used in recent work by Kriegman et al. (2017) and Cheney et al. (2015). This implementation of AFP uses the Pareto optimization strategy shown in Algorithm 1. The objectives are set to $\{d_t, a, e, v\}$. Each individual is ranked based on the number of individuals in the population that dominate it, and the population is sorted by this ranking, with lower being better. ns individuals are chosen to survive based on their ranking. In the event of ties, equivalently ranked individuals are sorted first by v, then e, then a, and finally d_t, and added until the population is filled. This strategy imparts a preference relation among the objectives for breaking ties.

Algorithm 1 : Pareto Optimization

```
Input: population $\mathcal{N}$, objectives $\mathcal{O}$, number of selections $ns$
  ◊ calculating ranking base on dominance, $\prec$
  for $n \in \mathcal{N}$:
      $\texttt{rank}(n) = \sum_{m \in \mathcal{N}} \mathbb{I}[m \prec n]$
  $\mathcal{N} \leftarrow \texttt{sort}(\mathcal{N})$ by rank
  $\mathcal{P} \leftarrow \emptyset$                                    ◊ parents
  $\texttt{front} \leftarrow 0$
  while $|\mathcal{P}| + |\mathcal{N}(n|\texttt{rank}(n) = \texttt{front})| \leq ns$:
      $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{N}(n|\texttt{rank}(n) = \texttt{front})$
      $\texttt{front} \leftarrow \texttt{front} + 1$
  if $|\mathcal{P}| < ns$:
      $\mathcal{S} \leftarrow \mathcal{N}(n|\texttt{rank}(n) = \texttt{front})$
      sort($\mathcal{S}$) by objective preference
      $\mathcal{P} \leftarrow \mathcal{P} \cup$ first $ns$ individuals in $\mathcal{S}$

  return $\mathcal{P}$
```

Novelty Search Novelty search (NOV) is the first of three behavioral search drivers used in this work. The implementation we use is summarized in Algorithm 2. Novelty search selects individuals that are the farthest apart in behavioral space, based on some distance measure. In this case, we use k-nearest neighbors with k=4 to determine the novelty score for each individual. Novelty is calculated as the average distance of an individual to its k nearest neighbors, where distance is defined as the Euclidean distance between the behavior vectors. The behavior vectors are scaled zero mean, unit-variance before computing the distances in order to account for the different scaling of the behaviors. The top ns most novel individuals are selected each generation.

Algorithm 2 : Novelty Search (Lehman and Stanley, 2008)

```
Input: population $\mathcal{N}$, behaviors $\mathcal{F}$, number of selections $ns$
  ◊ calculate $k$ nearest neighbors to each $n \in \mathcal{N}$
  $\mathcal{S}$ = neighbors($\mathcal{F}$, $k$)
  ◊ average distance over neighbors
  for $n \in \mathcal{N}$:
      $\texttt{novelty}(n) = \frac{1}{k} \sum_{s \in S_n} dist(n, s)$
  $\mathcal{P} \leftarrow \texttt{sort}(\mathcal{N})$ descending order of novelty
  return first ns individuals in $\mathcal{P}$
```

ϵ-Lexicase Selection ϵ-lexicase selection (referred to as LEX in this paper) is a version of lexicase selection (Spector, 2012; Helmuth et al., 2014) designed for use on continous-valued behaviors and demonstrated for symbolic regression problems (La Cava et al., 2016, 2018). It is described in Algorithm 3. LEX begins each selection event with the entire population in the selection pool and proceeds as follows.

"Cases", i.e. behaviors, are drawn randomly from $\mathcal{F}$ without replacement. Individuals are filtered from the selection pool if they are not within ϵ of the best performance for that behavior *among the current pool*. Here ϵ is defined as the median absolute deviation of $\mathbf{f}_i$, the values of f_i for all individuals in $\mathcal{N}$. Once the selection pool has winnowed down to one individual or has run out of cases, this process ends, and an individual from the selection pool is chosen at random.

Due to its design, LEX promotes individuals that are good at particular behaviors and particular subsets of behaviors. Cases impart outsized selection pressure based on how hard they are, where difficulty corresponds to the dispersion of objective performance. As a result LEX tends to promote diversity. A distinct variant of lexicase selection was recently for proposed for evolving robotic controllers (Moore and McKinley, 2016).

Algorithm 3 : ϵ-Lexicase Selection (La Cava et al., 2016)

```
Input: population $\mathcal{N}$, behaviors $\mathcal{F}$, number of selections $ns$
  for $f_i \in \mathcal{F}$:                          ◊ get $\epsilon$ for each $f_i$
      $\epsilon_i \leftarrow$ med_abs_dev($\mathbf{f}_i$)
  $\mathcal{P} \leftarrow \emptyset$                              ◊ parents
  do $ns$ times:
      $\mathcal{P} \leftarrow \mathcal{P} \cup$ GetParent($\mathcal{N}, \mathcal{F}, \epsilon$)    ◊ add selection to $\mathcal{P}$
  return $\mathcal{P}$

  GetParent($\mathcal{N}, \mathcal{F}, \epsilon$) :
    $\mathcal{F}' \leftarrow \mathcal{F}$                             ◊ objectives
    $\mathcal{S} \leftarrow \mathcal{N}$                             ◊ selection pool
    while $|\mathcal{F}'| > 0$ and $|\mathcal{S}| > 1$:
        $f_i \leftarrow$ random choice from $\mathcal{F}'$       ◊ pick random $f_i$
        $f_i^* \leftarrow$ min $f_i(n)$ for $n \in \mathcal{S}$        ◊ best score on $f_i$ in pool

        for $n \in \mathcal{S}$:                         ◊ filter pool
            if $f_i(n) > f_i^* + \epsilon_m$ then
                $\mathcal{S} \leftarrow \mathcal{S} \setminus \{n\}$
        $\mathcal{F}' \leftarrow \mathcal{F}' \setminus \{f_i\}$                     ◊ remove $f_i$
    return random choice from $\mathcal{S}$
```

Discovery of Objectives by Clustering Discovery of objectives by clustering (DOC) attempts to identify stepping stones for the search process based on the current performance of the population. Based on the interaction of robots with their environment, certain sets of behaviors may group together, indicating emergent progress towards certain sub-problems. To capture and preserve these sub-problems during selection, DOC clusters the behavior set $\mathcal{F}$ and averages performance over sets of behaviors to generate a new reduced-order derived objective set, G in Algorithm 4. As in previous work, we use X-MEANS to perform the clustering, resulting in clusters of potentially 2 to 5 derived objectives. These objectives are then used with Pareto Optimization (Algorithm 1) to drive search.

Algorithm 4 : Discovery of Objectives by Clustering (Krawiec and Liskowski, 2015)

```
Input: population $\mathcal{N}$, behaviors $\mathcal{F}$, number of selections $ns$
  $\{C_1 \dots C_k\}$ = cluster($\mathcal{F}$)        ◊ generate $k$ clusters
  $G \leftarrow \{g_1(n) \dots g_k(n), n \in \mathcal{N}\}$      ◊ new behavior set

  ◊ average behavior over clusters
  for $n \in \mathcal{N}$:
      for $j = 1 \dots k$:
          $g_j(n) = \frac{1}{|C_j|} \sum_{i \in C_j} f_i(n)$

  return Pareto Optimization($\mathcal{N}, G, ns$)
```

Random Search We also implement a random selection algorithm as a baseline comparison. Note that only selection is done randomly. Therefore search progresses as a random walk using the variation operators in NEAT.

Elitism For each algorithm, we implement elitist versions as another point of comparison. This choice was motivated by the results of Methenitis et al. (2015), who found that elitism improved the performance of novelty search in this problem domain. Elitism is implemented by checking the population after selection has occured. If the individual from the old population with the highest d_T is not in the new population, it is added by replacing the individual with the lowest d_T. Note that, by its design, the AFP algorithm is already elitist, and therefore no "elite" version is included.

Experiment

The settings for our experiment are shown in Table 1. We conducted 20 trials of each method, focusing on soft robots limited to 7^3 voxels in a 7x7x7 grid. To compare results, we collected the best fitness (body length travelled) found each generation by each algorithm for each trial and the complexity of the corresponding individual. We also compare the algorithms in terms of the median best individual created by that method for each trial.

In addition, we look at the diversity of behaviors found in the final generation's population. We calculate diversity based on the entire time trace of each individual in three dimensions. Let $\mathbf{x}(n) = [x_0, \dots, x_T, y_0, \dots, y_T, z_0, \dots, z_T]$ be the flattened trajectory of individual n. Then the diversity, D, of the population is calculated as

$$D = \frac{1}{N^2} \sum_{n,m \in \mathcal{N}} \frac{1}{2} \left(1 - \frac{cov(\mathbf{x}(n), \mathbf{x}(m))}{\sigma(\mathbf{x}(n))\sigma(\mathbf{x}(m))} \right) \quad (4)$$

D is between 0 and 1.

We take a structured approach to exploring the soft robots that result from our experiments. Based on the trajectories $\mathbf{x}$, we use clustering to identify groups of behaviors that have evolved. We use Hierarchical Density Based Clustering (HDBSCAN) (McInnes and Healy, 2017) to compute

Table 1: Settings for the evolutionary algorithms.

Setting	Value
population size	30
generations	1000
simulation time	5 sec
robot dimensionality	7^3
number of random inds per generation	1

clusters from the final generations of all runs with a minimum cluster size of 10 individuals. We present the resulting clusters with visualizations of the robot morphologies in the following section.

Results

The body lengths travelled by the best individuals found each generation by each method are shown in Figure 2. Note that the dotted lines represent selection methods with elitism. It is interesting to note that elitist algorithms always outperform non-elitist ones, across algorithms. In particular, random search with elitism outperforms all non-elite algorithms, a finding that has not been reported in other problem domains (e.g. (La Cava et al., 2016)). Overall, LEX with elitism produces soft robots that travel the farthest, followed by DOC with elitism. AFP, and RAND with elitism, and NOV with elitism all perform similarly. Among non-elite methods, LEX produces the farthest travelling robots, followed by DOC and NOV. RAND produces by far the worst solutions.

The body lengths travelled by the best individual found in each trial of each search driver is plotted in boxplot form in Figure 4. This figure emphasizes again the importance of elitism in driving the search towards better solutions. The median body lengths travelled for DOC with elitism outperforms other methods, followed by LEX with and without elitism, AFP, and RAND with elitism. We test for statistically significant differences using a Wilcoxon rank-sum test, shown in Table 2. Every search driver significantly outperforms RAND without elitism. However, surprisingly, no algorithm significantly outperforms RAND with elitism. It is worth noting the test between LEX with elitism and RAND with elitism is close to significant ($p = 0.12$), motivating an experiment with more trials. LEX, DOC, AFP with elitism also outperform NOV without elitism ($p <$ 3.73e-2).

Figure 3 shows the number of voxels in the best individual each generation for each method. RAND selection shows a clear bias towards larger solutions, with and without elitism. Both RAND variations show a slight upward trend after the first 100 generations. This suggests that the variation operators in NEAT may be prone to a small amount of bloat. It is also interesting to note the drop in size among best-found individuals across methods. This suggests, among randomly constructed robots, smaller ones tend to have a higher fit-

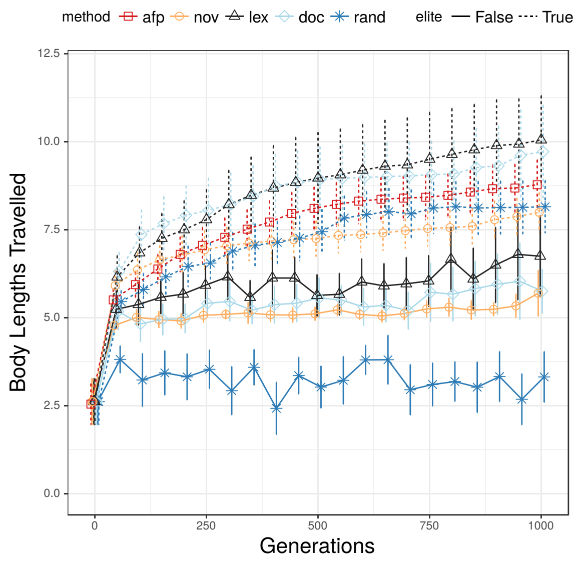

Figure 2: Body lengths travelled by the best individual found each generation for 7^3 dimension robots. Shapes represent different selection methods. The dotted line indicates elitism. Error bars denoted the 95% confidence interval.

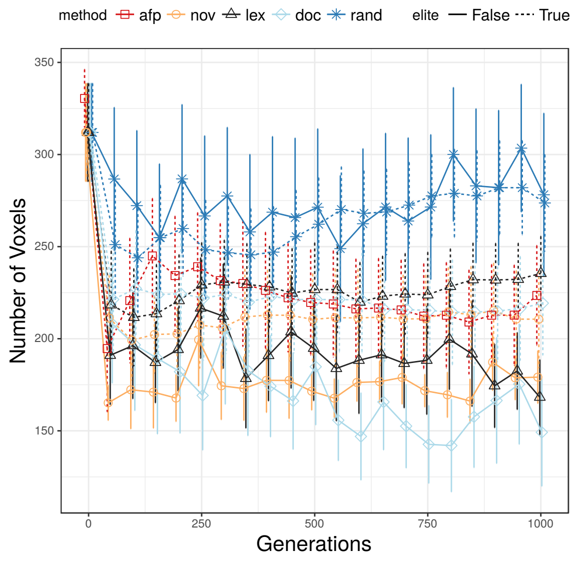

Figure 3: Number of voxels in the best individual found each generation for 7^3 dimension robots. Shapes represent different selection methods. The dotted line indicates elitism. Error bars denoted the 95% confidence interval.

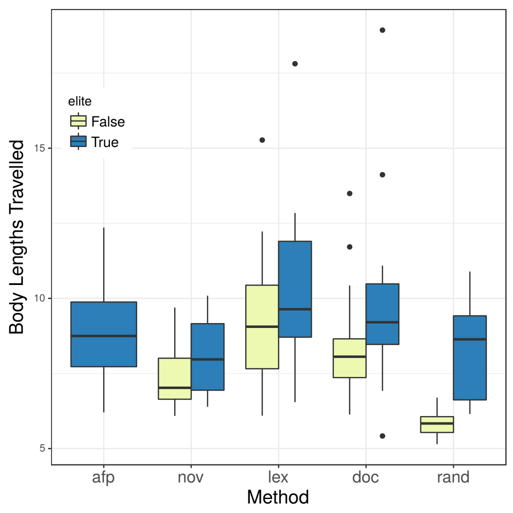

Figure 4: Furthest body length travelled for each trial of each selection method. Colors indicate the use of elitism.

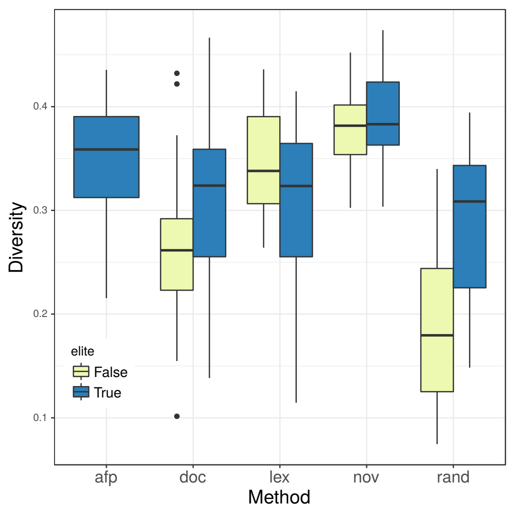

Figure 5: Behavioral diversity in the final generation for each trial of each selection method.

ness. Among the other search drivers, the elitist methods all tend to create larger robots than the non-elite methods, indicated by the dotted lines in Figure 3. Given the large amount of overlap between the standard error bars of the non-RAND search drivers, there appears to be no clear preference for larger or smaller solutions among the algorithms.

Next, we investigate the behavioral diversity of the final generation. In Figure 5, the average diversity (Eqn. 4) for each method is shown, with and without elitism. NOV produces the most diverse populations, which is expected since its objective function explicitly rewards behavioral diversity. It produces significantly more diverse solutions than all other methods except AFP and LEX without elitism ($p < 0.039$). Within families of algorithms, elitism only has a significant effect on the diversity of solutions for RAND, in which case it increases diversity. The fact that elitism increases the diversity of solutions obtained by RAND suggests that solution diversity plays a role in the improvement of solutions brought about by adding elitism.

The behaviors of the final populations are clustered according to the approach described in the experiments section. The HBDSCAN clustering produces 10 clusters of behavior. The trajectories of the robots in these clusters are plotted in Figure 6. The clusters are organized from top to bottom in order of highest average fitness among the robots in the cluster. The gaits of the softbots corresponding to the robots with the highest fitness within each cluster are shown in Figure 7, with the color corresponding to material type. Several behaviors are apparent in the results. The most successful behavior observed is achieved by T-shaped morphologies utilizing two types of muscle. This morphology and variation in stiffness between the muscles allows the robots to canter between front and back points of contact. The most fit individual, as well as the first two clusters in Figure 7, embody this technique. The second to last cluster of robots in Figures 6 and Figures 7 also evolve a cantering gait. As the trajectories of the top and bottom clusters attest, some morphologies are able to exploit the simulation environment to achieve higher fitness by falling forward at the beginning of the run. A video showing the gaits of these robots is available for reference [4].

Discussion and Conclusions

We find that elitism plays a critical role in the discovery of locomoting soft robot morphologies. The use of an elitist strategy with random search is competitive with other selection methods in our experiments. This result suggests the search space is highly non-convex, especially for search driven by scalar fitness. As a result, we are curious to know how other stochastic optimization techniques (e.g. hill climbing) might perform on this task. The results also suggest that behavioral search drivers, namely DOC and LEX, may be promising selection methods for evolving soft robot morphologies, although our experiment lacked adequate power to detect statistically significant differences. In addition, we find that behavioral clustering for exploratory analysis of the resultant robots provides a data-driven way to explore the morphologies.

The results also leave us with several other questions and directions for future research. A clear next step is to study how non-uniform environments would affect evolution, especially since behavioral search drivers provide a way to rate behavior within local regions of the environment / simulation. Future work could also consider a geodesic distance for fitness, in order to capture the total path of the robot more completely.

[4]http://tiny.cc/e2jbsy

Table 2: Significance test p-values comparing body lengths travelled using the pair-wise Wilcoxon rank-sum test with Holm correction for multiple comparisons. Bold indicates $p < 0.05$.

Method		afp	nov		lex		doc		rand
	Elite	True	False	True	False	True	False	True	False
nov	False	**3.73e-02**							
	True	1	1						
lex	False	1	0.08	1					
	True	1	**2.46e-03**	0.15	1				
doc	False	1	0.94	1	1	0.35			
	True	1	**1.93e-03**	0.18	1	1	0.57		
rand	False	**5.92e-09**	**1.62e-07**	**2.09e-09**	**2.15e-08**	**2.09e-09**	**2.15e-08**	**5.10e-7**	
	True	1	1	1	1	0.12	1	0.93	**8.77e-08**

References

Cheney, N., Bongard, J., and Lipson, H. (2015). Evolving soft robots in tight spaces. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 935–942. ACM.

Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (2013). Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 167–174. ACM.

Corucci, F., Cheney, N., Lipson, H., Laschi, C., and Bongard, J. (2016). Evolving swimming soft-bodied creatures. In *ALIFE XV, The Fifteenth International Conference on the Synthesis and Simulation of Living Systems, Late Breaking Proceedings*, page 6.

Helmuth, T., Spector, L., and Matheson, J. (2014). Solving Uncompromising Problems with Lexicase Selection. *IEEE Transactions on Evolutionary Computation*, PP(99):1–1.

Hiller, J. and Lipson, H. (2012). Automatic design and manufacture of soft robots. *IEEE Trans. on Robotics*.

Hiller, J. and Lipson, H. (2014). Dynamic Simulation of Soft Multimaterial 3d-Printed Objects. *Soft Robotics*, 1(1):88–101.

Krawiec, K. and Liskowski, P. (2015). Automatic derivation of search objectives for test-based genetic programming. In *Genetic Programming*, pages 53–65. Springer.

Kriegman, S., Cheney, N., Corucci, F., and Bongard, J. C. (2017). A minimal developmental model can increase evolvability in soft robots. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 131–138. ACM.

La Cava, W., Helmuth, T., Spector, L., and Moore, J. H. (2018). A probabilistic and multi-objective analysis of lexicase selection and -lexicase selection. *Evolutionary Computation*, pages 1–28.

La Cava, W., Spector, L., and Danai, K. (2016). Epsilon-Lexicase Selection for Regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 741–748, New York, NY, USA. ACM.

Lehman, J. and Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336.

Liskowski, P. and Krawiec, K. (2017). Discovery of Search Objectives in Continuous Domains. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 969–976, New York, NY, USA. ACM.

Liskowski, P., Krawiec, K., Helmuth, T., and Spector, L. (2015). Comparison of Semantic-aware Selection Methods in Genetic Programming. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1301–1307, New York, NY, USA. ACM.

McInnes, L. and Healy, J. (2017). Accelerated hierarchical density based clustering. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, pages 33–42. IEEE.

Methenitis, G., Hennes, D., Izzo, D., and Visser, A. (2015). Novelty search for soft robotic space exploration. In *Proceedings of the 2015 annual conference on Genetic and Evolutionary Computation*, pages 193–200. ACM.

Mitchell, M., Holland, J. H., and Forrest, S. (1994). When will a genetic algorithm outperform hill climbing. In *Advances in neural information processing systems*, pages 51–58.

Moore, J. M. and McKinley, P. K. (2016). A Comparison of Multiobjective Algorithms in Evolving Quadrupedal Gaits. In *From Animals to Animats 14*, Lecture Notes in Computer Science, pages 157–169. Springer, Cham.

Schmidt, M. and Lipson, H. (2011). Age-fitness pareto optimization. In *Genetic Programming Theory and Practice VIII*, pages 129–146. Springer.

Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM.

Spector, L. (2012). Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, pages 401–408.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162.

Vanneschi, L., Castelli, M., and Silva, S. (2014). A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines*, 15(2):195–214.

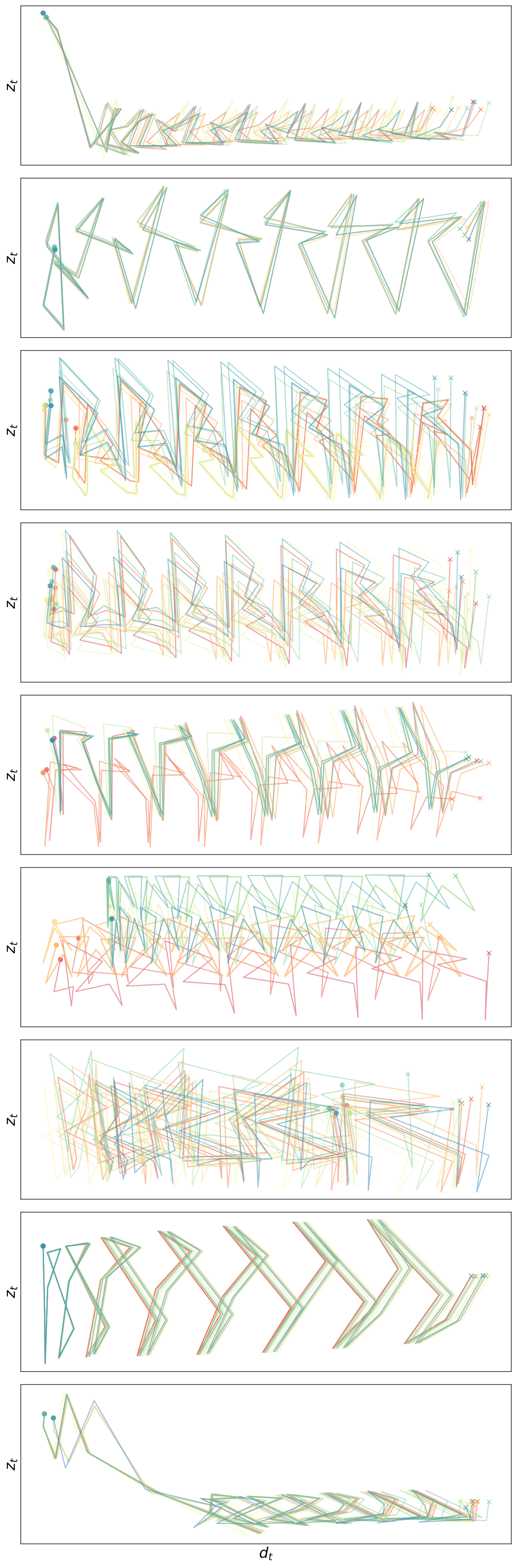

Figure 6: Clustered trajectores of robot behavior.

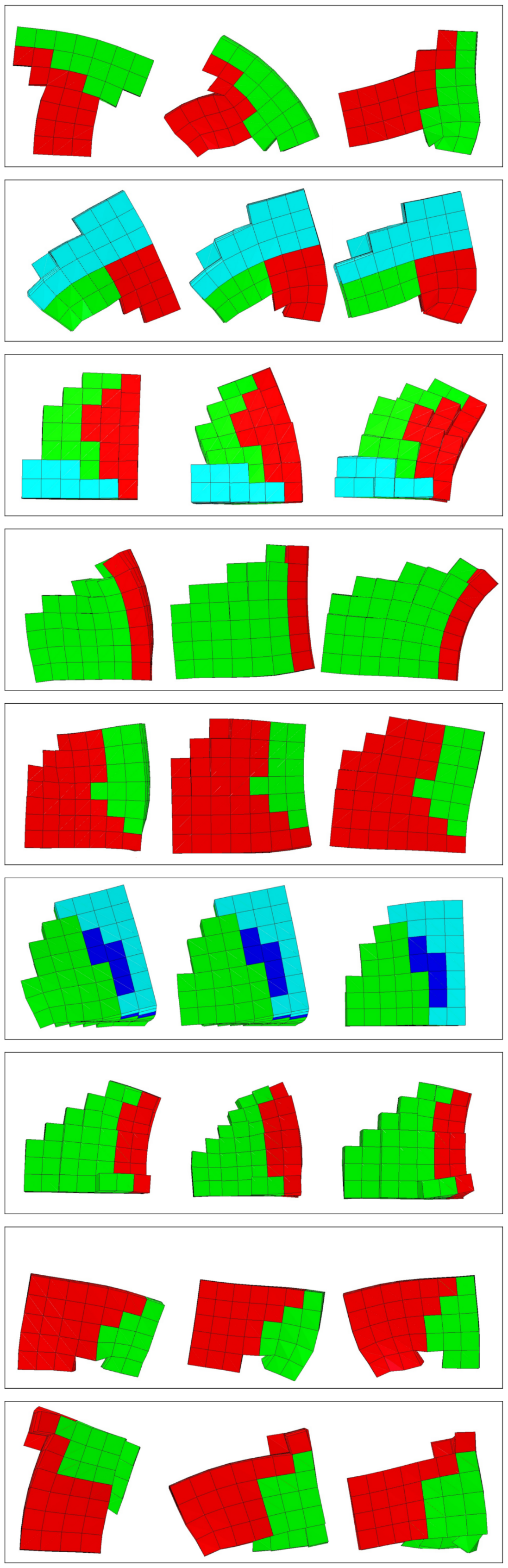

Figure 7: Robot gaits from each cluster in Figure 6.