Lexicase Selection Outperforms Previous Strategies for Incremental Evolution of Virtual Creature Controllers

Jared M Moore[1] and Adam Stanton[2]

[1]School of Computing and Information Systems, Grand Valley State University, Allendale, MI.
[2]School of Computing and Mathematics, Keele University, ST5 5BG, UK.
moorejar@gvsu.edu a.stanton@keele.ac.uk

Abstract

Evolving robust behaviors for robots has proven to be a challenging problem. Determining how to optimize behavior for a specific instance, while also realizing behaviors that generalize to variations on the problem often requires highly customized algorithms and problem-specific tuning of the evolutionary platform. Algorithms that can realize robust, generalized behavior without this customization are therefore highly desirable. In this paper, we examine the Lexicase selection algorithm as a possible general algorithm for a wall crossing robot task. Previous work has resulted in specialized strategies to evolve robust behaviors for this task. Here, we show that Lexicase selection is not only competitive with these strategies but after parameter tuning, actually exceeds the performance of the specialized algorithms.

Introduction

Evolutionary robotics (ER) seeks to evolve *robust* behaviors for animats in a variety of task domains. Typically, a general task encompasses many similar, but unique subtasks. Behaviors are often evolved for these subtasks, but fail to identify the general aspects of a problem (Pinville et al., 2011). Evolved controllers typically have difficulty expressing a general behavior in response to such a challenge, instead effectively handling a specific subset of tasks encountered during evolution while failing on related problems (Moore et al., 2013). As such, a capacity for generalized behaviors, indicating suitability to such a class of problems rather than specific instances, is a desirable outcome. For example, beyond just simple locomotion, a legged robot is considered more robust when it is able to walk on a variety of surfaces, each providing a different level of traction, despite these variations being instances of the same underlying problem.

Eliciting generalized control typically requires the algorithm to expose agents to a high number of different environments. In the ER domain, the increased computational or temporal overhead makes such a strategy infeasible for even moderately-sized problems. One solution is to divide and conquer: breaking down the problem and working incrementally towards success in specific components (Bongard, 2008). However, it is often unclear how an evolutionary regimen can be structured to train a controller for a task whilst also encouraging the behavior to generalize to more than one specific instance. The common response is to apply a considerable amount of domain knowledge and uniquely tailored evolutionary algorithms to specific problems. This can increase the time to develop a solution, discourages reusability of an algorithm to different problems, and prevents the application of common techniques to improve performance.

In this paper, we expand a previous investigation by Stanton and Channon (2013) on generalized controller evolution by implementing the Lexicase selection algorithm originally proposed by Spector (2012). The particular problem addressed in this work is the evolution of a neurocontroller for a fully-articulated quadrupedal animat in a 3D environment, supported by simulated rigid-body physics. The animat's task is to move towards a target location, an objective that includes the traversal of a wall situated halfway between the agent and the destination. Since the height of the wall can vary, the general wall traversal task (crossing walls of any height between a defined minimum and maximum) represents a class of sub-problems, the traversal of walls of specific heights. As in Stanton and Channon (2013), our goal is to apply evolutionary methods to discover animats that solve this general task, without incurring the computational overhead of an exhaustive search across the generalised problem. This earlier work demonstrated that general behaviors can be evolved for the wall-traversal task using specific strategies to present sub-problems over evolutionary time. However, the evolutionary strategies employed in these solutions were highly customized to the specific environment.

Building on the previous investigation, we add the Lexicase selection algorithm as an alternate evolutionary strategy. Lexicase selection has the potential to be a general evolutionary algorithm. Should it be competitive with these problem-optimized algorithms, it would be preferable to employ a general evolutionary algorithm capable of being applied to other problems with minimal change. Four separate treatments are conducted, each with a different strategy for limiting the number of environments an agent is exposed to

during evolution. The algorithms are compared based on how well they evolve robust individuals capable of traversing a variety of wall heights.

Our results show that while the previously investigated approaches do indeed evolve effective individuals for the generalized task, Lexicase selection is superior. We argue that a general algorithm, in this case Lexicase, is preferable to an algorithm constructed to fit the exact problem. First, Lexicase selection, without any special parameter tuning, achieves similar fitnesses to algorithms specifically structured to address the problem. Second, changing the number of environments individuals are evaluated on each generation impacts the performance of Lexicase. Finally, tuning the different parameters of the algorithm produces significantly better results than non-Lexicase algorithms.

Background

Evolutionary robotics (Nolfi and Floreano, 2000; Sims, 1994; Floreano et al., 2008; Doncieux et al., 2015) applies concepts derived from biology to the design of robotic systems. It has demonstrated an ability to address challenging problems in many task domains including: gaits (Clune et al., 2009), object manipulation (Bongard, 2008), biological study (Crespi et al., 2013; Doorly et al., 2009), and the optimization of morphology (Auerbach and Bongard, 2010; Bongard, 2010; Cheney et al., 2013). Often, these tasks have a single performance objective, or weighted sum, to assess the fitness of each individual. However, as task complexity increases, distilling performance in a single measure becomes increasingly difficult.

Lexicase selection was introduced by Spector (2012) for modal problems in genetic programming (GP). It falls in the domain of many-objective problems, where more than five objectives are used to evaluate potential solutions. Rather than consider all objectives in a many-objective optimization problem, Lexicase selection instead considers objectives individually. Per each selection event, one or many objectives may be used to compare individuals. Individuals are compared on all objectives over the course of evolution, but not necessarily every generation. Further details of the Lexicase algorithm are presented in the next section. Helmuth et al. (2014) have shown that Lexicase is effective at solving challenging problems in GP. Although it was initially introduced for GP, Lexicase has the potential to be a very effective algorithm in evolutionary robotics where objectives relating to performance (e.g. distance traveled, speed, stability, multiple tasks) and efficiency (e.g. energy expended, trajectory, passive elements) can result in many-objective problems. Moore and McKinley (2016) applied Lexicase selection in a multi-objective evolutionary robotics task. There, NSGA-II (Deb et al., 2002) slightly outperformed Lexicase selection, likely due to the relatively low number of objectives (3). Lexicase still outperformed a traditional Genetic Algorithm (GA). In this study, we attempt to evolve general Artificial Neural Network (ANN) controllers for quadrupedal animats capable of climbing over obstacles of 100 different heights. Each wall height is treated as a separate objective.

Methods

Robot Platform and Simulation Environment Figure 1 shows the quadrupedal animat used in this study. The robot has a cuboid torso and four legs placed at the corners. Each leg is divided into an upper and lower segment. The hip is a 2-degree of freedom (DOF) joint while the knee is a 1 DOF joint; see Table 1 for specific parameter values.

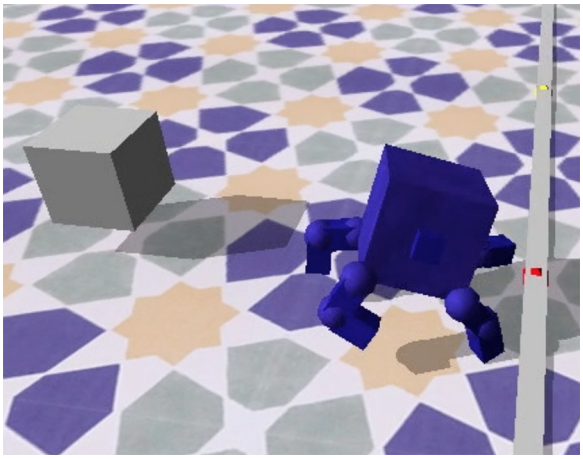

Figure 1: The quadrupedal animat and simulation environment in this study. The animat is tasked with crossing a wall (image right) and moving towards a target, represented by the box (image left).

Head dimension	$0.2 \times 0.2 \times 0.2$
Leg section dimension	$0.075 \times 0.05 \times 0.05$
Head mass	2.0
Leg section mass	0.5
Hip joint axis 1	vertical, range $[-\frac{\pi}{4}, \frac{\pi}{4}]$
Hip joint axis 2	horizontal, range $[0, \frac{\pi}{2}]$
Knee joint axis 1	horizontal, range $[0, \frac{\pi}{2}]$
Maximum torque	0.125

Table 1: Physical parameters of robot.

The robot's joints are actuated using a Proportional-Derivative (PD) control mechanism (Reil and Husbands, 2002) that takes a target angle as input and applies a torque to the joint according to Equation 1, where T is the applied torque, k_s and k_d are the spring and damper constants, θ_d is the desired angle, θ the current angle and $\dot{\theta}$ the angle change since the last timestep. As in previous work, $k_s = k_d = 0.5$ in this study.

$$T = k_s \times (\theta_d - \theta) - k_d \dot{\theta} \quad (1)$$

Simulations are conducted in the Open Dynamics Engine (ODE) (Smith, 2013), a real-time rigid body physics engine. ODE version 0.13.1 was used. ODE computes the interaction between the different rigid components of the robot as well as interaction with the environment; Table 2 lists key parameters used.

Timestep	0.02 seconds
Gravity	-1.2
Friction model	Pyramid approximation, $\mu = 2.0$
Global ERP	0.2
Global CFM	5.0×10^{-5}
Wall dimension	$0.05 \times 5.0 \times h$
Wall position	$x = 1; y = 0$
Target position	$x = 2; y = 0$
Start location	$x = 0; y = 0$
Simulation time	20 seconds

Table 2: ODE simulation parameters.

Task Environment Animats are evaluated on their ability to cross a wall and move towards a target position. Wall heights vary, from 0.01 up to a maximum value of 1.0 in 0.01 increments. Fitness values for animats are negative values, with a maximum fitness of 0.0 corresponding to an animat reaching the target. For analysis, we divide fitness values into the following bins: (1) reached objective ($>= 0.0$), (2) crossed wall ($>= -0.2$), (3) stuck on wall ($>= -0.6$), (4) reached wall ($>= -1.0$), and (5) did not reach wall (< -1.0).

Controller The controller is a feed-forward ANN with fixed, fully-connected topology. Inputs comprise a number of signals from the environment as well as spontaneous activity generated by sinusoidal functions. These inputs are shown in Table 3. The ANN has 12 hidden nodes and 12 outputs mapping to desired angles for each joint at the next timestep within the ranges described in Table 1. $\tanh$ function is used for the transfer function in the hidden layer, and the logistic function for outputs. The ANN is updated in lock-step with the dynamics simulation, and inputs are propagated completely through the network at each update.

Lexicase Selection Algorithm 1 presents the Lexicase selection algorithm with modifications made for this study acting as the selection operator in our genetic algorithm. Lexicase differs from traditional selection methods by evaluating individuals against a number of objectives per selection event. Starting with one objective, individuals are compared based on their performance (lines 3-19). Only in the case of ties are additional objectives considered. If multiple individuals have the same performance, those individuals are then compared on the next objective in the random ordering. If at the end of the selection process, there are multiple individuals still left the algorithm selects an individual at random from the subset (line 21).

1	$\sin(2\pi t)$
2	$\cos(2\pi t)$
3	balance: $\arccos(\mathbf{H}[10])$
4	$(\lvert\overrightarrow{H_l}\rvert - \lvert\overrightarrow{H_r}\rvert) \div H_{width}$
5–12	hip joint angles
13–16	knee joint angles

Table 3: ANN controller inputs, where $\mathbf{H}$ is the ODE rotation matrix of the robot's head, $\lvert\overrightarrow{H_x}\rvert$ is the distance from each side of the robot's head to the target and H_{width} is the width of the head.

Algorithm 1 Lexicase Selection Pseudocode. Adapted from Spector (2012) and Moore and McKinley (2016)

```
1:  subset ← GetSubsetFromPopulation(population, 4)
2:  obj_order ← Shuffle(fitness_objectives)
3:  for obj in obj_order do
4:      r_sub ← RankSubset(subset, obj)
5:      tie_index ← 0
6:      for i in 1 to length(r_sub) do
7:          if r_sub[i][obj] ≥ threshold * r_sub[0][obj]
    then
8:              tie ← True
9:              tie_index ← i
10:         end if
11:     end for
12:     if tie is True then
13:         subset ← r_sub[0 : tie_index]
14:     else
15:         tie ← False
16:         subset ← r_sub[0]
17:         break
18:     end if
19: end for
20: if tie is True then
21:     return RandomChoice(subset)
22: else
23:     return subset[0]
24: end if
```

Two individuals with similar performance do not necessarily have the exact same distances travelled in the evolutionary robotics domain. In this study, we adopt a modification to the Lexicase algorithm proposed in Moore and

McKinley (2016). Individuals are considered "tied" if they are within a threshold of performance (5%) on an objective when compared to the best individual in that objective (lines 7-10). This serves to relax performance requirements, selecting individuals that complete the task, but are not necessarily "optimal" in one objective. We parameterize this threshold as *fuzz_factor* varying the range of equivalent performance based on the treatments described in Results.

Evolutionary Algorithm A generational EA was employed in this study. A genome specifies a set of floating-point weights for the ANN controllers. Populations comprise 50 individuals, each randomly initialized based on a starting seed corresponding to the replicate number. For non-Lexicase strategies (see below) fitness-based selection is used where the population is ranked and the lower half is replaced with mutated copies of the upper half, recombined with a random individual from the upper half using single-point crossover. The mutation rate used was $\frac{2}{N}$, where N is the length of the genome. For Lexicase strategies, a new population is created each generation using Lexicase selection to choose parents, and creating children using the same recombination parameters. Morphologies of the animats remain fixed throughout evolution. The number of generations varies among treatments and is noted in each treatment's description following this section. An animat is evaluated based on their Euclidean distance from a target at the end of a simulation. High fitnesses are only attained if the animat climbs over a wall of varying height.

Treatments Stanton and Channon (2013) found that different strategies of presenting component problems in an incremental evolutionary system affects the generalization of the final solutions. In that work, the authors separated the strategies into *homogeneous* (those that do not revisit earlier components) and *heterogeneous* (those that do). A clear difference was found between the two groups, with heterogeneous clearly outperforming the other group in the given task. In the present work, we apply the commonly-used *random* strategy (the worst-performing heterogeneous strategy), as well as the best strategy from Stanton and Channon (2013) and the trivial *direct* strategy and compare to new strategies that use the Lexicase selection mechanism. We next describe the individual treatments.

Random The Random treatment is used as a baseline for comparison to the other treatments in this study. At each generation, a randomly selected wall height is used to evaluate the performance of the individuals in the population. There is no specific strategy to programmatically introduce wall heights, hence, in one generation the wall may be very short, while the next could be the maximum height.

Direct Single Subtask This treatment presents only one component of the task–in the present work, the wall at its maximum height.

Oscillating Strategies These strategies were amongst the best treatments from Stanton and Channon (2013). Species experience a cyclical change in the wall task, where the maximum height reached is a sinusoidal function of evolutionary time and the period is a hyperparameter chosen empirically. In the 'max' version of this strategy, the oscillation achieves maximum amplitude in the first period. In the 'increasing' version, the amplitude of the oscillation steadily increases over evolutionary time, and in the 'adaptive' version, the amplitude increases according to whether average population fitness has increased in this generation compared to the previous generation. After initial parameter sweeps we found that in this implementation of the task, cyclic strategies with period 100 generations demonstrated the highest fitness and these treatments are presented in the results section for comparison.

Lexicase We consider the Lexicase selection algorithm in three ways. First, we implement Lexicase using rule-of-thumb parameters and compare against other strategies by keeping the total number of evaluations (and thus computation time) constant. Agents are evaluated in 5 randomly selected environments per generation from a set of 100 total wall heights and the algorithm proceeds for 1000 generations, totalling 250,000 evaluations ($1000\times50\times5$). During the selection process, individuals in the subset are compared to each other using the Lexicase algorithm described previously. A parent is selected from a subset of 5 randomly selected individuals.

In the second Lexicase configuration, we examine how changing the number of environments affects the performance of the Lexicase solution. Specifically, we examine 1, 2, 5, 10, and 20 environments. Again we hold computation time constant, so consequently the number of generations changes in each case. We also examine additional tie thresholds of 10% and 15%, comparing each across the four different environment sizes. Changing the threshold alters how likely two individuals are to be tied in one environment, hence changing how many environments in which they are compared, per selection event.

Finally, we explore the potential of Lexicase whilst holding the number of evolutionary events (the creation of new individuals) constant but relaxing the requirement for equivalent computation time. For this, the best tie parameter is chosen from the earlier study and then each environment size is explored for 5000 generations (250k evolutionary events but 250k$\times n$ evaluations, where n is the number of environments).

Results

Evolved Behaviors All treatments evolve at least some wall crossing behaviors, although the level of success varies. Figure 2 illustrates a few of the evolved behaviors of animats completing the task. A video of select evolved individuals can be seen at `https://youtu.be/Q8z0Ktj2n1M`.

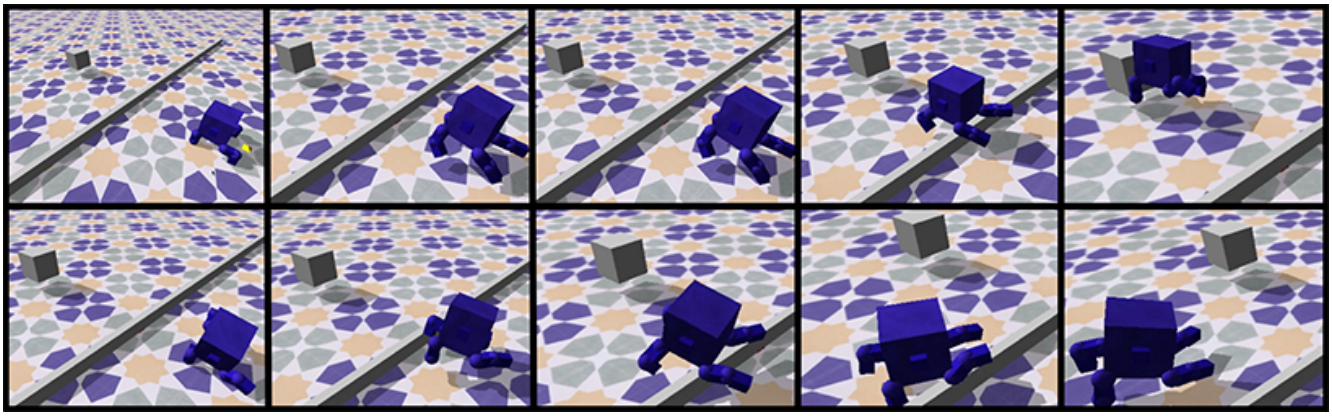

Figure 2: Two evolved animats crossing walls of varying heights. (*Top*) Perfect behavior results in crossing the wall and reaching the box on the other side. (*Bottom*) Some individuals cross the wall but in doing so do not have time to reach the box, these instead end up with high, but not perfect fitnesses.

As the wall height increases, it becomes more challenging for the animats as they have relatively short legs. This sometimes results in wall crossing behavior, but not reaching the target precisely.

Cross Treatment Comparisons Figure 3 plots the performance of the best individuals per replicate across the treatments. We define the best individual for a given replicate as the individual in the last generation of evolution with the highest mean fitness across the 100 environments. Mean fitness is calculated by a final validation step run after the evolutionary process has completed. As shown in the boxplot, Lexicase selection with a fuzz factor of 1.05 and evolved for 1,000 generations (*Lex_5E_1.05F_1000G*) outperforms the direct subtask and random treatments significantly ($p < 0.01$). For this and subsequent significance tests we apply the Mann-Whitney-Wilcoxon Rank-Sum Test. However, for the *Oscillating (Max Amplitude) 100*, *Oscillating (Inc. Amplitude) 100*, and *Oscillating (Adapt Amplitude) 100* treatments this difference is not significant ($p = 0.117$, $p = 0.067$, and $p = 0.681$ respectively). *Lex_5E_1.05F_1000G* is comparable to these treatments, even though it evolves for 1,000 versus 5,000 generations in the other treatments. Furthermore, the fuzz factor and number of environments were chosen by hand.

Best Individual Validation Performance Although the average across replicates is informative from a performance perspective, it does not provide information as to how robust the best individual from each replicate is. Figures 4, 5, and 6 plot the performance of the best individual per replicate across each of the 100 wall heights. Replicates are ordered from lowest (top) to highest performing (bottom) in the figures. Each individual is classified per the categories mentioned in Methods. As shown in the figures,

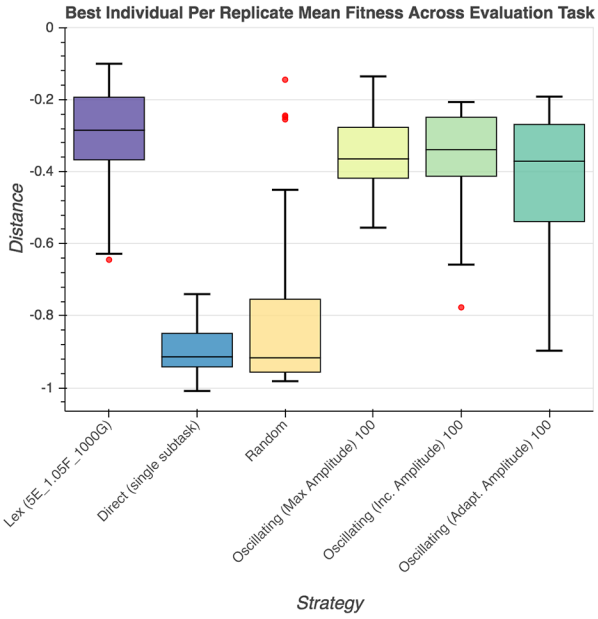

Figure 3: Performance distribution of the mean score of the best individual per replicate across the treatments.

Lex_5E_1.05F_1000G has the highest number of wall traversals, while also having the lowest number of failures. *Oscillating (Max Amplitude) 100* and *Oscillating (Inc. Amplitude) 100* evolve many successful individuals, but performance generally decays rapidly as evidenced by the higher proportion of light and dark blue cells. Validation performance for the *Oscillating (Adapt Amplitude) 100* treatment is similar to the *Oscillating (Max Amplitude) 100* and *Oscillating (Inc. Amplitude) 100* treatments but we have omitted the plot due to space considerations. Tall walls (right side of the figures) tend to be difficult considering the high number of failing individuals there across all three treatments.

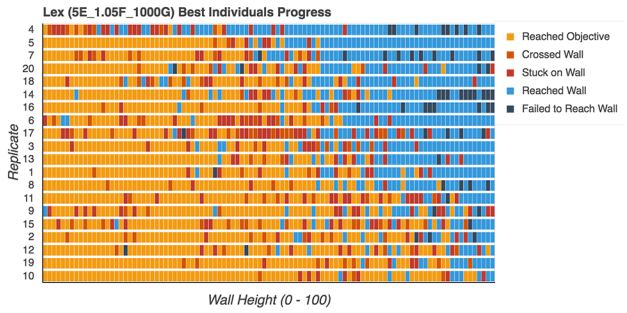

Figure 4: Performance of the best individual per replicate for the Lexicase 1.05 Fuzz Factor, 5 environments, 1,000 generation treatment. Light colors denote crossing the wall and reaching the objective. Darker shades indicate a failure to traverse the wall.

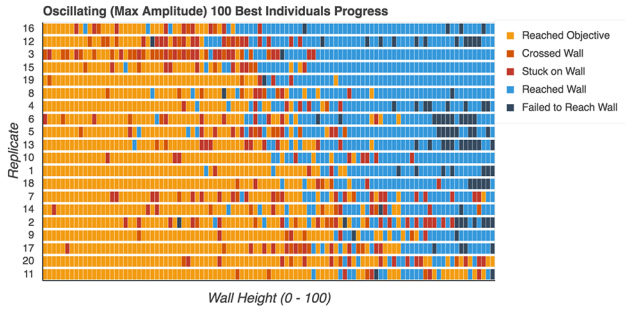

Figure 5: Performance of the best individual per replicate for the Oscillating (Max Amplitude) 100 treatment.

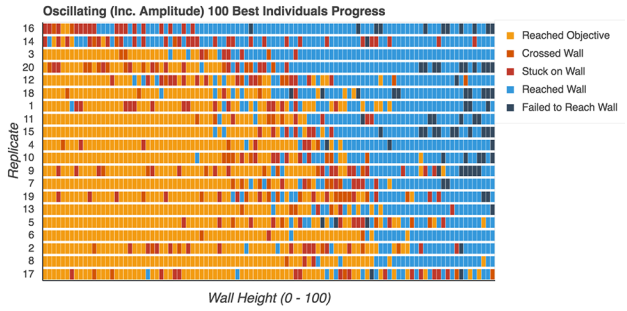

Figure 6: Performance of the best individual per replicate for the Oscillating (Inc. Amplitude) 50 treatment.

Figure 7 shows the average performance across replicates for each treatment broken down into the five categories. The Lexicase treatment has comparable performance to the four oscillating strategies, while the number of perfect and crossing individuals is low in the Direct Subtask and Random treatments. These figures show that, on average, *Lex_5E_1.05F_1000G* , *Oscillating (Max Amplitude) 100*, and *Oscillating (Inc. Amplitude) 100* handle the majority of wall heights in the task. In combination with the heatmaps, it appears that most replicates handle wall heights up to about 0.5, but above that the increasing height is challenging.

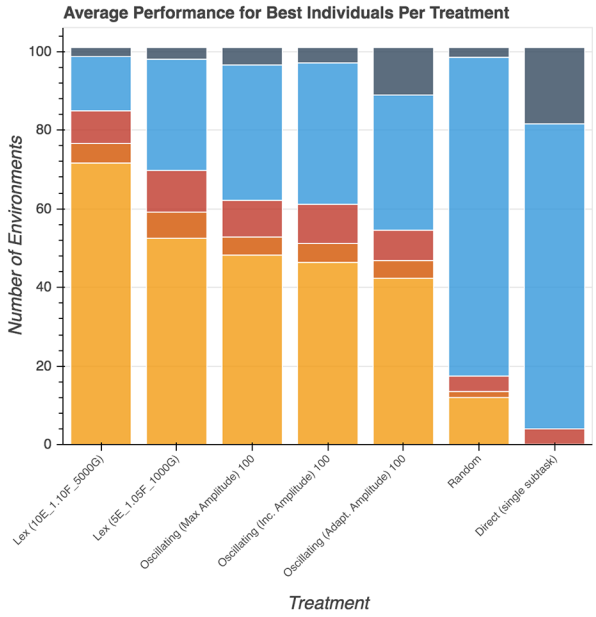

Figure 7: Average number of environments completed at each level for the best individual per replicate across treatments.

Lexicase Parameter Optimization After initially testing Lexicase selection with a baseline set of parameters, Fuzz Factor 1.10 and number of environments = 5, we next explore different parameter configurations with three additional Lexicase treatment groups. Figure 8 plots the performance of the four Lexicase groups against each other for a total of 20 treatments. The first two treatment groups employ the same configuration as the 1.05 fuzz factor group, with the number of environments dictating the number of generations in each treatment, maintaining the number of individual simulations at 250,000. Other Lexicase variations with this same approach are: 1 env/5000 gen, 2 env/2500 gen, 10 env/500 gen, and 20 env/250 gen. A fuzz factor of 1.10 produces the highest performing treatment (*Lex_2E_1.10F_2500Gv*) with 2 environments evolved for 2,500 generations among the three Lexicase groups with variable generations (left three in figure). However, this treatment is not significantly different than *Lex_1E_1.05F_5000Gv*, *Lex_2E_1.05F_2500Gv*, *Lex_1E_1.10F_5000Gv*, *Lex_1E_1.15F_5000Gv*, and *Lex_2E_1.15F_2500Gv* ($p < 0.05$).

Performance generally degrades as the number of environments increase beyond 2 in these three treatment groups. This result is contrary to our initial intuition as the addition of more environments during evaluation of an individ-

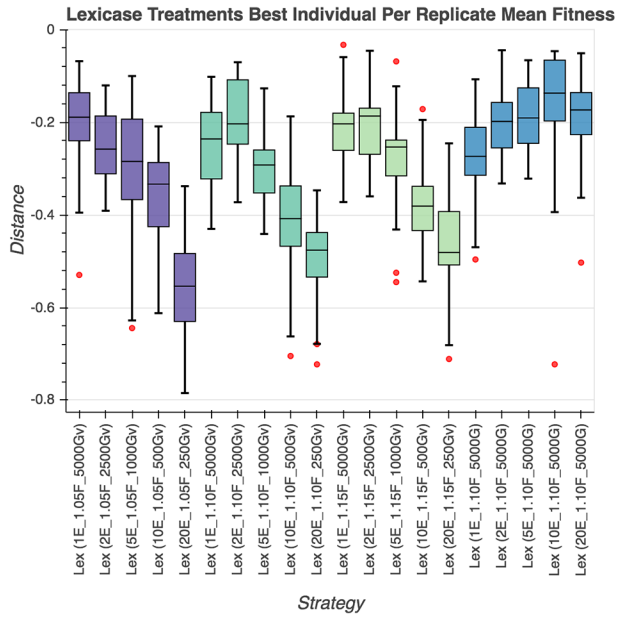

Figure 8: Performance distribution of the mean score of the best individual per replicate across the Lexicase treatments. Treatments are colored based on their parameter configurations.

ual should result in more robust controllers. Changing the fuzz factor results in 2 environment treatments having the highest mean fitness, but performance still degrades in the 5, 10, and 20 environment strategies.

Within each of these three treatment groups, performance generally degrades above 2 environments. We hypothesize that this is due to the reduced evolutionary time given to the 5, 10, and 20 environment treatments which focus on maintaining the number of individual simulations at 250,000. To test this, we conduct a fourth Lexicase group of treatments with a fuzz factor of 1.10 and all five environment configurations evolve for 5,000 generations. Here, we observe that performance increases from 1 environment up to 10 environments, with the 20 environment treatment exhibiting decreased performance. The *Lex_10E_1.10F_5000Gv* treatment is significantly better than all but the *Lex_2E_1.10F_2500* and *Lex_5E_1.10F_5000G* treatments. Furthermore, this treatment results in the highest average performance during validation, with animats crossing the wall in 85 of the 100 environments, compared to 70 environments in the *Lex_5E_1.05F_1000G* treatment.

Figure 9 plots the performance per each replicate during the validation step for the *Lex_10E_1.10F_5000Gv* treatment. When compared to Figures 4, 5, and 6, the individual replicates generally exhibit increased task aptitude and ability to generalize across wall heights. Indeed, this treatment has the highest mean fitness and is significantly different than the *Lex_5E_1.05F_1000G* , *Oscillating (Max Amplitude) 100*, *Oscillating (Inc. Amplitude) 100*, and *Oscillating (Adapt Amplitude) 100* ($p < 0.01$ for all). Three of the replicates are nearly successful in every environment, with only a few failures mixed in. Still, it should be noted that no replicate is completely perfect in any of the treatments conducted in this study.

Figure 9: Performance of the best individual per replicate in the task. Light colors denote crossing the wall and reaching the objective. Darker shades indicate a failure to traverse the wall.

Conclusion

In this paper, we have applied Lexicase selection to a previously examined evolutionary robotics problem. The original evolutionary algorithms were customized to fit the dynamics of the problem, introducing new wall heights in a highly structured manner attempting to realize general behaviors. Whereas, Lexicase selection evaluates individuals on a subset of randomly selected wall heights. Considering the performance of Lexicase selection versus the previously studied treatments, it appears that Lexicase is suitable as a general purpose evolutionary algorithm with the potential to generalize to other problems without the need to devise new and problem-specific EAs for each study.

Our initial Lexicase treatment (1.05 fuzz, 5 environments) was competitive with the evolutionary strategies proposed in Stanton and Channon (2013). Adjusting the fuzz factor and number of environments further increased performance, ultimately resulting in the most effective evolutionary algorithm explored in this study. Moreover, the Lexicase strategies employed here involved tuning algorithm parameters, rather than exploring specialized algorithms for this specific problem. Adding additional environments does help initially, but there appears to be a plateau in performance as indicated by 5, 10, and 20 environment results for a fuzz factor of 1.15. Here, we examined gradations of 1, 2, 5, 10, and 20 environments, but it is possible that this parameter can be tuned further.

After consideration of the first Lexicase results, we conclude that the lack of opportunity for genetic change drives

the decreasing performance. Changing the number of generations to 5,000 across the different number of environments in Lexicase results in increased performance for 2 environments and above. Although there are now more individual environment evaluations and a corresponding increase in computational time, we consider this acceptable for an offline evolutionary algorithm. The wall clock time of these runs is only longer than the initial treatments (approximately 22 hours) presented in Figure 3 by approximately 20 hours. For an offline algorithm, this increase is acceptable as a deliverable will not be impacted significantly by this increase in computational time.

Future work will examine further parameter tuning of the Lexicase algorithm including the number of environments, fuzz factor, and number of generations. Alternate evolutionary robotics problems will be explored to assess the suitability of Lexicase to other problems in this domain.

Acknowledgements

This work was partly supported by the Evolutionary Systems Research Group in the Faculty of Natural Sciences, Keele University.

References

Auerbach, J. E. and Bongard, J. C. (2010). Dynamic resolution in the co-evolution of morphology and control. In *Proceedings of the Twelfth International Conference on Artificial Life*, pages 451–458, Odense, Denmark.

Bongard, J. C. (2008). Behavior chaining: Incremental behavioral integration for evolutionary robotics. In *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 64–71, Winchester, United Kingdom.

Bongard, J. C. (2010). The utility of evolving simulated robot morphology increases with task complexity for object manipulation. *Artificial Life*, 16(3):201–223.

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, Amsterdam, The Netherlands. ACM.

Clune, J., Beckmann, B. E., Ofria, C., and Pennock, R. T. (2009). Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2764–2771, Trondheim, Norway.

Crespi, A., Karakasiliotis, K., Guignard, A., and Ijspeert, A. (2013). Salamandra robotica ii: An amphibious robot to study salamander-like swimming and walking gaits. *IEEE Transactions on Robotics*, 29(2):308–320.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182 –197.

Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. G. (2015). Evolutionary robotics: What, why, and where to. *Frontiers in Robotics and AI*, 2(4).

Doorly, N., Irving, K., McArthur, G., Combie, K., Engel, V., Sakhtah, H., Stickles, E., Rosenblum, H., Gutierrez, A., Root, R., Liew, C. W., and Long, J. (2009). Biomimetic evolutionary analysis: Robotically-simulated vertebrates in a predator-prey ecology. In *Proceedings of the IEEE Symposium on Artificial Life*, pages 147–154, Nashville, Tennessee, USA.

Floreano, D., Husbands, P., and Nolfi, S. (2008). Evolutionary Robotics. In *Handbook of Robotics*. Springer Verlag, Berlin.

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

Moore, J. M., Clark, A. J., and McKinley, P. K. (2013). Evolution of station keeping as a response to flows in an aquatic robot. In *Proceedings of the 2013 ACM Genetic and Evolutionary Computing Conference*, pages 239–246, Amsterdam, Netherlands. ACM.

Moore, J. M. and McKinley, P. K. (2016). *A Comparison of Multiobjective Algorithms in Evolving Quadrupedal Gaits*, pages 157–169. Springer International Publishing, Aberystwyth, UK.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence and Technology of Self-Organizing Machines*. The MIT Press.

Pinville, T., Koos, S., Mouret, J.-B., and Doncieux, S. (2011). How to promote generalisation in evolutionary robotics: The progab approach. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pages 259–266, Dublin, Ireland. ACM.

Reil, T. and Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, 6(2):159–168.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. *Artificial Life*, 1(4):353–372.

Smith, R. (2013). Open Dynamics Engine, http://www.ode.org/.

Spector, L. (2012). Assessment of problem modality by differential performance of Lexicase selection in genetic programming: A preliminary report. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 401–408, Philadelphia, Pennsylvania, USA. ACM.

Stanton, A. and Channon, A. (2013). Heterogeneous complexification strategies robustly outperform homogeneous strategies for incremental evolution. In *Proceedings of the 12th European Conference on Artificial Life*, pages 973–980, Taormina, Italy.