Dynamic mutation in MAP-Elites for robotic repertoire generation

Jørgen Nordmoen[1], Eivind Samuelsen[1], Kai Olav Ellefsen[1] and Kyrre Glette[1,2]

[1] Department of Informatics, University of Oslo, P.O Box 1080 Blindern, 0316 Oslo, Norway
[2] RITMO, University of Oslo, P.O Box 1017 Blindern, 0315 Oslo, Norway
jorgehn@ifi.uio.no

Abstract

One of the core functions in most Evolutionary Algorithms is mutation. In complex search spaces, which are common in Evolutionary Robotics, mutation is often used both for optimizing existing solutions, described as exploitation, and for spanning the search space, called exploration. This presents a difficult challenge for researchers as mutation parameters must be selected with care in order to balance the two, often contradictory, effects. Strategies that vary mutation during the search often try to estimate these effects in order to modify the mutation parameters. In this regard MAP-Elites, a Quality Diversity algorithm, presents an interesting opportunity. Because factors related to exploration and exploitation are readily available during the search, optimization based on these factors could be utilized to improve the search. In this paper we study how online adaptation of mutation rate, dynamic mutation, affects MAP-Elites in order to gain insight into how the search process is affected by the mutation rate. Our study compares fixed and dynamic mutation parameters for two different complex gait controllers. The results show that dynamic mutation combines favorably with MAP-Elites and that there is a strong relation between mutation parameters and exploration that can be utilized.

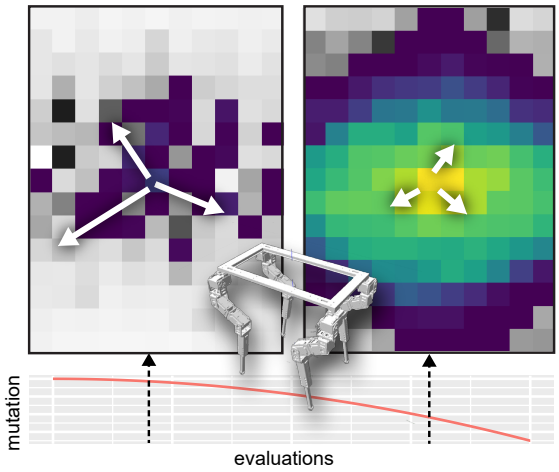


Figure 1: By changing the mutation rate, the characteristics of the search can be altered as the search progresses. This can allow for greater exploration when few solutions are discovered and more exploitation when many solutions are present.

Introduction

Dynamic mutation has long been a staple of Evolutionary Strategies (ES) where varying the degree of mutation can aid in both exploration and exploitation (Eiben and Smith, 2007). ES can be used when the search space is unknown or difficult to predict in order to minimize the number of search parameters to optimize (Eiben et al., 1999). These properties are often described as *self-adaptation* and can be invaluable help in difficult problems. However, self-adaptation is often an additional step that researchers must implement without knowing what results to expect. It is therefore interesting to implement self-adaptive methods in new problem or algorithm domains to gain a shared understanding of expected trade-offs.

A recent trend in Evolutionary Robotics (ER) is the use of Quality Diversity (QD) algorithms to evolve both high performing solutions and a large diversity in behavior (Pugh et al., 2016). A growing body of work has shown that QD algorithms are able to solve problems that have previously been difficult to solve (Lehman and Stanley, 2008) and also that retaining inferior solutions to enhance diversity can lead to better results (Mouret and Clune, 2015). Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) is one such QD algorithm that has been used to great effect (Cully et al., 2015). The simplicity of the algorithm coupled with the insight that different non-optimal behaviors can be interesting in their own right have given rise to a new way of thinking about controller development for robots (Duarte et al., 2017).

One reason to combine dynamic mutation and MAP-Elites is to reduce the computational complexity often associated with QD algorithms (Gaier et al., 2017). This complexity limits the number of repetitions available for finding a good mutation rate and could preclude some solutions from developing. By employing a parameter free dynamic mutation the solutions can adapt to the problem at hand re-

quiring less computation than repeating the experiment with varying mutation rates.

It is also interesting to combine dynamic mutation and MAP-Elites because of the structured design of MAP-Elites. Two readily available measurements in MAP-Elites are *coverage* and *precision*, which define the current proportion of discovered solutions and the fitness of these solutions. These measurements are analogous to exploration and exploitation, and previous results (Samuelsen and Glette, 2018) indicate that there could be a relation between coverage, precision and mutation rate.

In this paper, we analyse how dynamic mutation affects MAP-Elites trying to answer two key questions, *1)* can dynamic mutation achieve the same performance as a hand-tuned mutation rate without the need for a manual search for mutation parameters and *2)* is there a relation between the notion of coverage and mutation rate that can be utilized during the search process. The analysis is a contribution to further the understanding of QD algorithms and could be used to guide new development within the field.

Background

Evolutionary Strategies

ES has for a long time been one of the core methods in Evolutionary Algorithms (EAs) (Bäck et al., 2013). It can be viewed as a specialization of an EA where the building blocks are continuously adapted to gain additional advantages (Bäck and Schwefel, 1993). ES has been very successful thanks to innovations such as CMA-ES (Hansen et al., 2003), which have seen widespread use in many different fields (Gregory et al., 2011; Bergstra et al., 2011).

At the core of ES is the notion that mutation parameters are simply another parameter in the genome (Eiben and Smith, 2007). The mutation parameter is mutated along with the rest of the genome, following a different scheme than the rest, and is then used as the basis to mutate the other parameters in the genome. This means that the mutation operator is able to adapt to the problem and also means fewer parameters to pre-define.

Quality Diversity and MAP-Elites

QD algorithms are a relatively recent addition to EAs (Pugh et al., 2016). The main contribution of QD algorithms is the insight that both fitness and diversity should be valued. This manifests in the generation of a repertoire of solutions instead of one or a few globally best solutions. Instead of optimizing for the best solution these algorithms aim to *illuminate* the search space to discover many different behaviors to solve the problem (Mouret and Clune, 2015). To search for these solutions QD algorithms define *behavior characteristics*, which are intended to capture the overarching desired behavior traits of the solutions. In addition a *quality metric* is defined to capture the fitness of a given behavior characteristic. These two definitions are combined to allow for a variety of behaviors while at the same time optimizing the quality of each individual behavior.

MAP-Elites is a QD algorithm where the behavior space is divided into a discrete N-dimensional grid (Mouret and Clune, 2015). Each dimension in the grid represents a behavior characteristic and each cell contains a potential solution with a given quality, or fitness. At each iteration a solution is selected at random before being mutated, the new solution is then evaluated to determine the behavior and quality metrics. Once these metrics are determined the solution is placed in the grid, if the grid already contains a solution for the given behavior the quality of the two solutions are compared and the higher quality solution is retained.

Several works (Pugh et al., 2016; Auerbach et al., 2016; Cully and Demiris, 2017) have undertaken studies into how QD algorithms work. These works have shown that QD algorithms work well when behavior characteristics are highly related to the desired goal and that different QD algorithms excel with different behavior characteristics.

Behavior Repertoire

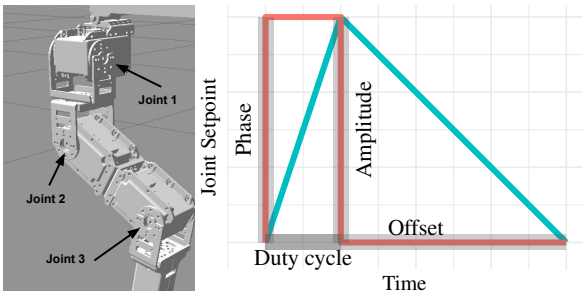
Behavioral repertoire learning is closely related to QD where the intention is to generate a set of useful behaviors that can later be selected among to solve a task (Cully and Mouret, 2013). From early on behavior repertoire learning has been combined with QD (Cully and Mouret, 2016) where the QD algorithm is responsible for generating the diversity of behaviors desired. Recently this combination has been extended to not only create a repertoire of behaviors, but also to create abstract controllers on top of the repertoire (Duarte et al., 2017). The main idea of the work presented is to utilize QD to generate a repertoire of low-level movement primitives, which can later be used by high-level controllers without needing to understand how to generate the movement. This allows high-level controllers to be created for specific tasks or environment and have these controllers work on a range of robots.

Methods

To study how dynamic mutation interacts with MAP-Elites we evolve a behavior repertoire for a mammalian quadruped (Nygaard et al., 2016, 2018). This task represents a difficult real-world world problem and is a realistic test to ensure that the methods presented are useful when evolving solutions to complex problems. The main source of complexity stems from the difficult morphology resulting in difficult fitness gradients (Fukuoka et al., 2003; Nordmoen et al., 2018).

The simulations were performed using Robot Operating System (ROS)[1], Gazebo[2] and Open Dynamics Engine

[1] `https://www.ros.org/`
[2] `http://gazebosim.org/`



(a) Joint configuration within a leg.

(b) The two different splines used for all joints. The red spline is used for joint 3 while the slower rising blue spline is used for both joint 1 and 2.

Figure 2: Joint configuration and spline controller.

(ODE)[3], utilizing the SFERESv2 framework (Mouret and Doncieux, 2010), all code is published online[4].

The rest of this section will describe the gait controller and the evolutionary setup before detailing the experiments performed.

Gait Controller

The controller used for the experiments is an open-loop *ensemble looping first-order spline*. Each joint of the four legs, shown in Figure 2a, is given a spline controller and the parameters that control the splines are *amplitude*, *phase*, *duty cycle* and *offset*, depicted in Figure 2b. Amplitude represents the movement of the joint. Phase represents the time-offset for when the joint should start moving, in Figure 2b the phase is set to 0.0. Duty cycle represents the duration in which the joint moves forward. Finally, offset is the default value of the joint at rest. These parameters can either be completely independent, coordinated intra-leg and/or intra-body. This gives us a total of *48* parameters that can then be reduced by coordinating between legs and/or between joints in a leg. E.g. the phase parameter can be reduced from 12 parameters to 4 by coordinating intra-body so that each leg has an individual phase parameter duplicated for joint 1, 2 and 3.

MAP-Elites

To generate a behavior repertoire we simulate each individual controller for 10 seconds, continuously monitoring the behavior to calculate behavior characteristics. The simulation is set up to abort if an individual falls over before the full simulation time. This allows us to speed up simulation and utilize 'time-before-fall' as surrogate for stability. However, we do not consider solutions that fell successful controllers and remove them after the evolutionary process is done, before calculating any metrics used in the results. This is done to avoid the likely wrong estimates of the behavior characteristics of falling individuals. Because of the difficulty in deciding when a fall has started it is difficult to ensure that the behavior characteristics are correct. As solutions are able to walk for longer periods of times the estimates become better and better before being considered successful. In other words, the fallen individuals are only used as stepping stones during evolution. Fitness and behavior characteristics are identical to our previous work (Nordmoen et al., 2018) and are designed to evolve gait primitive repertoires. Table 1 is a summary of the MAP-Elites parameters used. Fitness of an individual, n, is defined as

$$fitness(n) = \begin{cases} T_i & \text{if the robot fell over} \\ T_i + stability(n) & \text{otherwise} \end{cases} \tag{1}$$

T_i is the time the individual was upright, and $stability$ is defined as

$$stability(n) = \begin{cases} C - SM(\omega_x, \omega_y) & \text{if } SM(\omega_x, \omega_y) < C \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

SM is the Squared Magnitude of the x and y components of the body angular momentum, ω, over the evaluation period and C is a constant allowing for maximization of $fitness$. The behavior characteristics are average turn rate defined as

$$\frac{1}{N}\sum_{i=2}^{N}\frac{(\psi_i - \psi_{i-1})}{(t_i - t_{i-1})} \tag{3}$$

and average velocity defined as

$$\frac{1}{N}\sum_{i=1}^{N}\overline{v}_i \tag{4}$$

where N is the number of samples *before* the robot fell, ψ is the yaw in radians, t_i is the time of sample i and $\overline{v}$ is the velocity of the robot in the x and y dimensions.

We also define *coverage* and *precision*, in accordance with (Mouret and Clune, 2015), as the number of filled cells divided by the total number of cells and the average fitness of filled cells. Where the fitness of each individual is normalized to the range $[0, 1]$.

Experimental Setup

To experiment with dynamic mutation in MAP-Elites we tested two different controller configurations, a simple *Standard* controller and a more complex *Hard* controller.

The simpler Standard controller has 16 parameters (out of 48), where 12 parameters control the amplitude of each individual joint and the last four parameters are phase for each leg. The duty cycle and offset parameters are set to default values of 0.5 and the rest pose of the robot, respectively.

[3] http://www.ode.org/

[4] https://folk.uio.no/jorgehn/sigma_exp/experiment_code.zip

Repetitions	30
Map size	11×21
Evaluations	*100 000* Generations: 1000 Batch size: 100
Initial population	1000
Evaluation time	10 seconds
Recombination	None
Mutation	Type: Gaussian σ: *variable* Probability: 1.0
Behavior characteristics	Dimensions: 2 X-axis: Turn rate (Eq: 3) Y-axis: Average speed (Eq: 4)
Fitness	Stability (Eq: 2)

Table 1: MAP-Elites simulation parameters.

The more complex Hard controller has 28 parameters (out of 48), where the first 16 are identical to the Standard controller and 12 parameters are individual offset values for all joints. The duty cycle is set to default 0.5. The complexity of this controller stems from the additional degrees of freedom that comes with the 12 offset parameters and the extended need for coordination.

Mutation is normalized in both parameter- and genotype-space. In parameter-space the mutation is normalized to $[0, 1]$ before being scaled to the individual range of the parameters. To ensure that the expected distance after mutation, in genotype-space, is equal for both controllers, the mutation rate is normalized according to the number of parameters in use. The normalization is performed by multiplying the mutation rate with

$$\frac{1}{\sqrt{\|g\| - 1}} \tag{5}$$

where $\|g\|$ is the number of parameters in use by the controller. This normalization is performed regardless of mutation scheme.

Static Mutation In order to compare static mutation rate with dynamic mutation we performed a grid search with several fixed mutation rates. The grid search was performed in two steps, first we tested values from 0.4 down to 0.025, halving the mutation rate at each step, before we subdivided two ranges to see if the performance could be further improved. The values tested were $0.4, 0.2, 0.1, 0.075, 0.05, 0.03, 0.025$ for a total of seven static mutation rate tests.

Evolutionary Strategy (ES) The first dynamic mutation configuration is an ES, *uncorrelated mutation with one step size* (Eiben and Smith, 2007, p. 75). For each solution we add one additional parameter that is used to mutate the other parameters in the genome. This parameter is self-adaptive during the search by mutation along with the rest of the genome following a separate scheme,

$$\sigma_i = \sigma_{i-1} \cdot e^{\tau \cdot N(0,1)} \tag{6}$$

where

$$\tau \propto \frac{1}{\sqrt{\|g\|}} \tag{7}$$

σ_0 is set to a default starting value of 0.5 and has a minimum of 0.025, to correspond with the smallest static mutation tested.

Simulated Annealing (SA) In addition to testing ES we also tested with a simple linear scheme dependent on the number of completed evaluations. This mutation scheme is inspired by the dropping temperature in Simulated Annealing and is an alternative to the more complex ES. This mutation scheme proceeds by scaling the mutation rate linearly from a *start* to *stop* dependent on the number of evaluations,

$$\sigma_i = start - (start - stop) * \frac{i}{N} \tag{8}$$

where i is the current evaluation and N is the total number of evaluations. In our experiments we fixed the start value to 0.2 and stop value to 0.025.

Coverage-based (Cov) The last dynamic mutation rate tested is a linear scheme, similar to simulated annealing, but instead of number of completed evaluations uses coverage to decrease mutation rate,

$$\sigma_i = start - (start - stop) * Coverage_{i-1} \tag{9}$$

where $Coverage_{i-1}$ is the Coverage of the repertoire after the previous evaluation based on the number of solutions that were able to walk the full evaluation time.

This dynamic mutation is a first attempt at probing the relationship between mutation rate and exploration. Like simulated annealing this scheme introduces two parameters, *start* and *stop*, which need to be set. In our experiments we tested two different parameter sets; we tested a start value of 0.4 (*Cov4*) and 0.1 (*Cov1*) with a stop value of 0.025.

Results and Discussion

To illustrate the produced behaviors we have uploaded videos of a few gaits along with the generational development of their repertoires[5].

[5] https://folk.uio.no/jorgehn/sigma_exp/

	Scheme	Parameters
Static	Constant	[0.025, 0.03, 0.05, 0.075, 0.1, 0.2, 0.4]
Dynamic	ES	Start: 0.5 Minimum: 0.025
	SA	Start: 0.2 Stop: 0.025
	Cov	Start: [0.4, 0.1] Stop: 0.025

Table 2: Summary of mutation schemes and parameters.

Grid Search

The result of the grid search is shown in Figure 3a. The figure plots the precision and coverage as a point, (x, y), for each evaluation, in order. These points are then combined into a path to show how precision and coverage evolves together. This allows us to easily compare both precision and coverage at the same time over the whole evolutionary run.

For the Standard controller we see a wide range of different precision and coverage values while the Hard controller converges to smaller and less fit repertoires regardless of mutation rate.

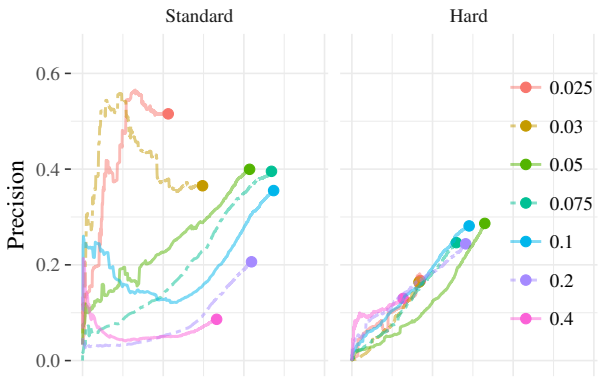
The coverage also show that none of the available controllers are able to reach the full behavior space. To allow the coverage based mutation scheme to exhibit its full potential we scale the coverage in *equation 9* with 0.6. This allows the coverage based mutation to exhibit its full range of values without the need for re-scaling the map size, which allows for direct comparison between the static and dynamic mutation experiments.

Dynamic Mutation

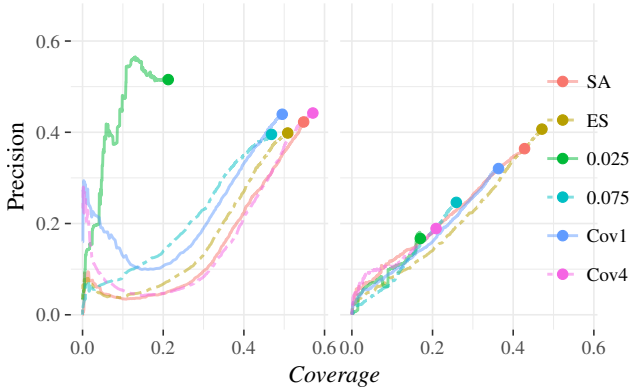
After performing a search for viable static mutation rates we now compare static and dynamic mutation. Figure 3b shows both precision and coverage for all dynamic mutation rates and two selected static mutation rates. For clarity, only two static mutation rates are selected to reduce clutter in the following figures.

For the Standard controller the dynamic mutation rates are competitive with static mutation in coverage, however, the dynamic mutation schemes are not able to achieve the same precision. For the Hard controller the dynamic mutation rates are able to surpass in both coverage and in precision.

Figure 4 shows the full distribution of all runs for the last evaluation. This figure shows that evolving the Standard controller is consistent between most of the mutation rates. For the Hard controller on the other hand we see a large variance in both metrics. This is the result of difficulties in discovering even a single viable controller, which results in some runs with zero coverage and precision.



(a) Static mutation rates



(b) Dynamic mutation rates

Figure 3: Average precision and coverage plotted as (x, y) points for each evaluation. The circle represents the final values of each mutation scheme.

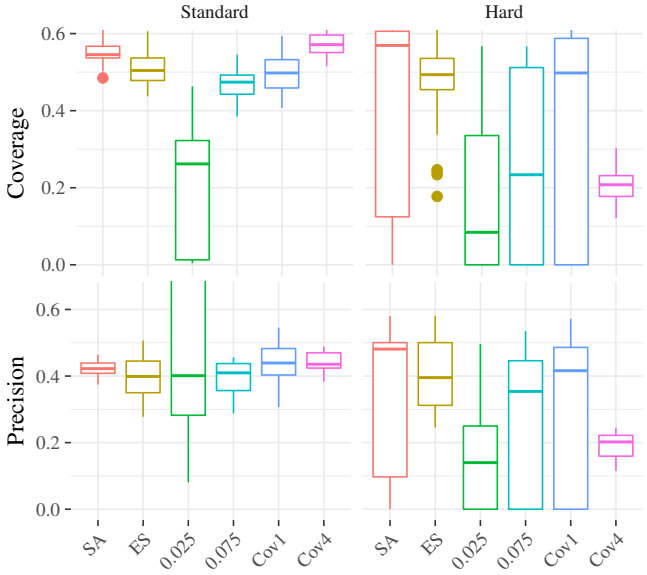


Figure 4: Box plot for the last evaluation of both coverage and precision, illustrating the full distribution of each experiment.

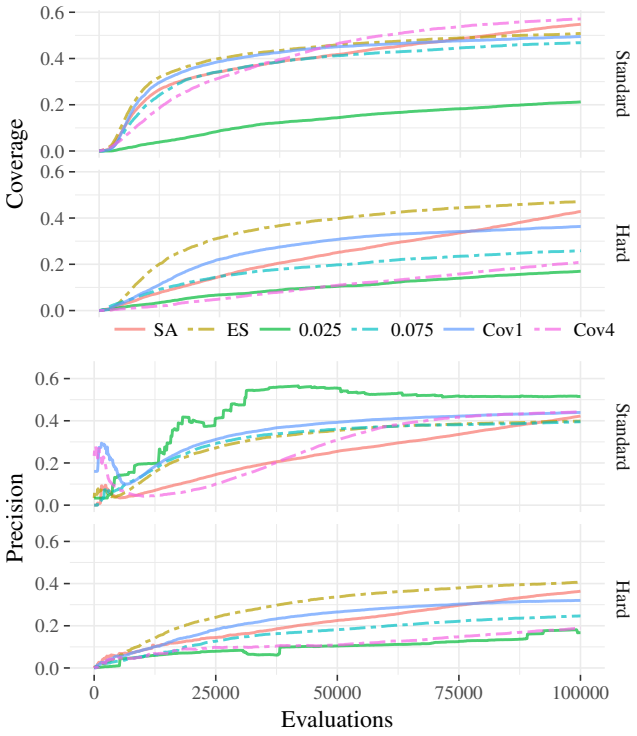


Figure 5: Development of average coverage and precision throughout evolution.

To understand how each mutation scheme develops throughout evolution we have plotted both coverage and precision in Figure 5. From the figure we can see that coverage and precision see rapid growth early in the evolution before plateauing. The only exception to these trajectories is the *SA* mutation scheme which shows almost linear growth in both coverage and precision and would seem to continue this trend even as the others converge.

For two of the three dynamic mutation rates the development of σ is pre-defined, for ES it is more difficult to predict. To understand this development we plot the average σ in the maps in Figure 6. The figure shows several interesting moments in regards to dynamic mutation. For one we can see that ES appears initially in the plot at different σ values for the different controllers. This is due to the removal of solutions that fall, which allows the mutation rate to diverge from the initial value, before any solutions are able to walk the full simulation time. Additionally it is apparent that our coverage based scheme with a starting value of 0.4 is not able to increase coverage, and thus lower the mutation rate, for the Hard controller. This relation further explains the poor performance of this mutation scheme, for the Hard controller, in Figure 4. It should also be pointed out that the reason that the SA mutation rate is not a perfect straight line, in Figure 6, is that we are measuring the average mutation rate of solutions in the map, if a solution from an earlier

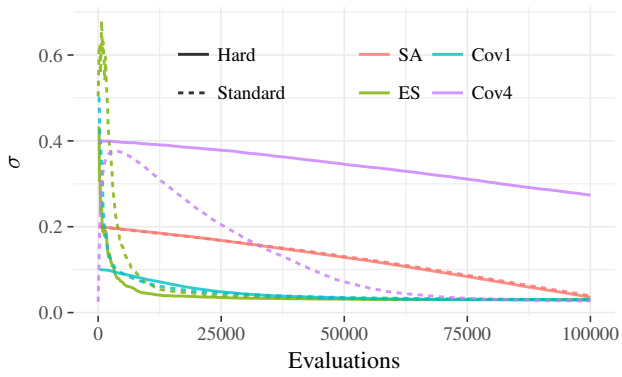


Figure 6: Development of average mutation rate throughout evolution. The standard deviation is not shown, but is consistently less than 0.1 for all mutation schemes.

evaluation is not replaced it will still contribute to the average σ. This also explains why the two controllers for the SA mutation scheme are not perfectly overlapping. Another observation we can make on the basis of this is that solutions are constantly being replaced. If this were not the case then we would expect a stagnation at some fixed mutation rate.

Effect Size Comparison

In order to compare all mutation schemes we calculated Cliff's delta (Cliff, 1993), with a 0.99 confidence level, accounting for Pareto dominance in both coverage and precision and the two different control algorithms, in accordance with (Samuelsen and Glette, 2018). This gives us the confidence interval of the effect size, which indicates whether or not one scheme is better, by how much and if the difference is statistically significant. The result is summarized in Table 3. From the table it is clear that the dynamic mutation schemes perform better than the static mutation rates. It is also clear that our simulated annealing inspired mutation rate performs best of the dynamic rates, by beating most of the other dynamic schemes. However, it is interesting to note that ES seem to outperform it when compared to the worst performing static mutation rates, $0.025, 0.2, 0.4$, which may indicate that ES performs more consistently than SA. Finally we can see that there is little difference between the coverage based and the two other dynamic mutation rates.

Mutation Rate

Since the SA mutation scheme performs better than the other dynamic mutation schemes it is interesting to see if we can detect any relationship between mutation rate and coverage that could be exploited in a more explicit mutation scheme. Figure 7 shows mutation rate as a function of coverage together with a simple quadratic decay model. We then performed a linear fit of this model resulting in an adjusted R^2 of 0.97 with all predictors showing statistical significance

	SA	ES	Cov1	Cov4
0.025	$\mathbf{0.51 \pm 0.21}$	$\mathbf{0.63 \pm 0.14}$	$\mathbf{0.46 \pm 0.22}$	$\mathbf{0.40 \pm 0.20}$
0.03	$\mathbf{0.62 \pm 0.19}$	$\mathbf{0.66 \pm 0.14}$	$\mathbf{0.54 \pm 0.19}$	$\mathbf{0.46 \pm 0.22}$
0.05	$\mathbf{0.47 \pm 0.22}$	$\mathbf{0.35 \pm 0.20}$	$\mathbf{0.33 \pm 0.22}$	0.14 ± 0.23
0.075	$\mathbf{0.52 \pm 0.19}$	$\mathbf{0.31 \pm 0.19}$	$\mathbf{0.32 \pm 0.21}$	$\mathbf{0.32 \pm 0.23}$
0.1	$\mathbf{0.67 \pm 0.18}$	$\mathbf{0.42 \pm 0.19}$	$\mathbf{0.37 \pm 0.22}$	$\mathbf{0.32 \pm 0.21}$
0.2	$\mathbf{0.73 \pm 0.21}$	$\mathbf{0.86 \pm 0.11}$	$\mathbf{0.61 \pm 0.21}$	$\mathbf{0.17 \pm 0.14}$
0.4	$\mathbf{0.72 \pm 0.21}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.68 \pm 0.21}$	$\mathbf{0.92 \pm 0.55}$
ES	$\mathbf{0.30 \pm 0.21}$	N/A	N/A	N/A
Cov1	$\mathbf{0.21 \pm 0.20}$	-0.02 ± 0.21	N/A	N/A
Cov4	0.01 ± 0.22	$\mathbf{0.19 \pm 0.11}$	-0.02 ± 0.24	N/A

Table 3: Cliff's delta for an abridged number of the mutation schemes, with 0.99 confidence level. The values indicate the confidence interval of the effect size. Positive values favour the column while negative favour the row. Significant results are marked in bold.

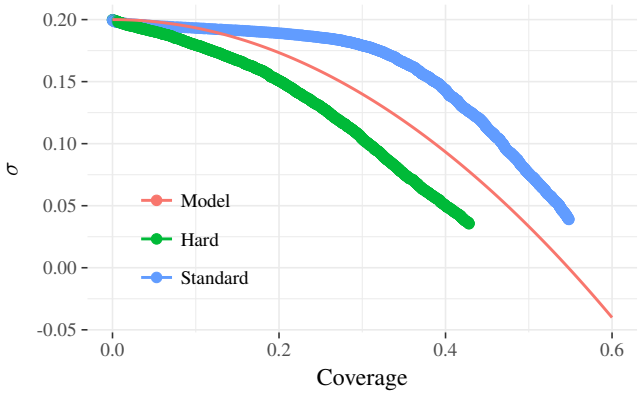


Figure 7: Modelling σ as a function of coverage. The data is for the SA dynamic mutation scheme, shown in green and blue, and the fitted quadratic decay model in red.

of $p < 0.001$. This shows that a possible further extension to our coverage based mutation scheme should utilize a quadratic form.

Discussion

Our results show that dynamic mutation has the potential to increase both coverage and precision when applied to quadruped repertoire generation in MAP-Elites. For both controllers tested it is clear that dynamic mutation is able to achieve similar or better results than any static mutation rate. The reason for this is likely the added opportunity to change between exploration and exploitation. Any static mutation rate must necessarily trade-off exploration against exploitation and the optimal balance is difficult and time-consuming to discover. The dynamic mutation rates are able to adapt this trade-off during the search and therefore out-compete the static mutation rates tested in this paper. We cannot exclude the possibility that we have not found the optimal static mutation rate that would be able to compete with the dynamic mutation schemes. However, when taking into account that each repetition of our experiment requires > 50 hours to generate the necessary statistics this also illustrates the difficulty in finding such an optimal value and why dynamic mutation can be worth the additional implementation complexity.

When comparing different dynamic mutation schemes there are several trade-offs to be made. The results show little difference between the schemes, however, they indicate some of their respective strengths and weaknesses. For ES we see that the results are consistent and, even for the difficult Hard controller, the method is able to generate high performing repertoires without fault. We can contrast this with the coverage based mutation scheme, which is able to generate preferable repertoires for the simple Standard controller, yet for the Hard controller is not able to consistently generate a repertoire. The same is true for the *SA* mutation scheme, which falters in a few runs for the Hard controller, and can be seen in Figure 4. The reason for this is likely the initial parameter, which could be tuned better, however, this highlights the advantage ES has by being almost parameter free.

When taking performance into account, we see that the ES scheme is not the overall best performer. The reason for this seems to be that ES greedily optimizes either precision or coverage. This can be seen in Figure 6 where ES appears with a large mutation rate initially, exploration, before rapidly decreasing the mutation rate, exploitation. In contrast the two other dynamic mutation schemes decreases mutation rate over a longer evolutionary period that we hypothesize leads to better solutions later by generating more diversity early in the search.

Our analysis of the SA mutation scheme, Figure 7, showed that there is a link between coverage and mutation rate. This indicates that as coverages increases, in MAP-Elites, the need for exploration declines and the search can transition to exploit already discovered solutions. We believe that this observation can be used to further improve the coverage based mutation scheme and that the pre-defined search space in MAP-Elites can facilitate online adaptation of search parameters.

Future extension of this work would be to develop a more complete dynamic mutation method that would be able to utilize both coverage and precision in order to change mutation rate. Ideally this should be parameter free, yet retain the quadratic decay property to ensure that diversity is maintained along with growing quality. Further testing on different control architectures would also be beneficial to discover the limits of the design.

Conclusion

In this paper we compared static and dynamic mutation to understand how MAP-Elites could benefit from different mutation schemes. Our results show that dynamic mutation is compatible with MAP-Elites and is able to surpass static

mutation rates when evolving a repertoire of behavior primitives for a quadruped robot with two different complex control algorithms. The results show that by applying dynamic mutation to MAP-Elites parameter tuning can be reduced to a minimum, decreasing the turn-around time for experiments. For computationally complex search algorithms, like MAP-Elites, this can be a considerable benefit.

Our experiments on different dynamic mutation schemes revealed a trade-off between explicitly defined mutation scaling and an almost parameter free ES. The results indicate that for difficult problems the ES is able to evolve repertoires consistently, yet is not able to attain the same quality as the other dynamic mutation schemes.

This work is a modest extension to the growing body of research into MAP-Elites. By demonstrating that established methods such as ES can be combined successfully with MAP-Elites we have peered into a new avenue of research and opened the door for more advanced application domains.

Acknowledgements

This work was partially supported by The Research Council of Norway as a part of the Engineering Predictability with Embodied Cognition (EPEC) project, under grant agreement 240862 and through its Centres of Excellence scheme, project number 262762.

References

Auerbach, J. E., Iacca, G., and Floreano, D. (2016). Gaining insight into quality diversity. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 1061–1064. ACM.

Bäck, T., Foussette, C., and Krause, P. (2013). Evolution strategies. In *Contemporary Evolution Strategies*, pages 7–45. Springer.

Bäck, T. and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554.

Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological Bulletin*, 114(3):494.

Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature*, 521(7553):503–507.

Cully, A. and Demiris, Y. (2017). Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*.

Cully, A. and Mouret, J.-B. (2013). Behavioral repertoire learning in robotics. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 175–182. ACM.

Cully, A. and Mouret, J.-B. (2016). Evolving a behavioral repertoire for a walking robot. *Evolutionary computation*, 24(1):59–88.

Duarte, M., Gomes, J., Oliveira, S. M., and Christensen, A. L. (2017). Evolution of repertoire-based control for robots with complex locomotor systems. *IEEE Transactions on Evolutionary Computation*.

Eiben, A. E., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on evolutionary computation*, 3(2):124–141.

Eiben, A. E. and Smith, J. E. (2007). *Introduction to evolutionary computing*, volume 53. Springer.

Fukuoka, Y., Kimura, H., and Cohen, A. H. (2003). Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *The International Journal of Robotics Research*, 22(3-4):187–202.

Gaier, A., Asteroth, A., and Mouret, J.-B. (2017). Data-efficient exploration, optimization, and modeling of diverse designs through surrogate-assisted illumination. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 99–106. ACM.

Gregory, M. D., Bayraktar, Z., and Werner, D. H. (2011). Fast optimization of electromagnetic design problems using the covariance matrix adaptation evolutionary strategy. *IEEE Transactions on Antennas and Propagation*, 59(4):1275–1285.

Hansen, N., Müller, S. D., and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18.

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

Mouret, J.-B. and Clune, J. (2015). Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.

Mouret, J.-B. and Doncieux, S. (2010). SFERESv2: Evolvin' in the multi-core world. In *Proc. of Congress on Evolutionary Computation (CEC)*, pages 4079–4086.

Nordmoen, J., Ellefsen, K. O., and Glette, K. (2018). Combining MAP-Elites and Incremental Evolution to Generate Gaits foraMammalian Quadruped Robot. In *Applications of Evolutionary Computation*, pages 719–733. Springer.

Nygaard, T. F., Martin, C. P., Tørresen, J., and Glette, K. (2018). Self-Modifying Morphology Experiments with DyRET: Dynamic Robot for Embodied Testing. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page to appear in. ACM.

Nygaard, T. F., Tørresen, J., and Glette, K. (2016). Multi-objective evolution of fast and stable gaits on a physical quadruped robotic platform. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*.

Pugh, J. K., Soros, L. B., and Stanley, K. O. (2016). Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40.

Samuelsen, E. and Glette, K. (2018). Multi-objective Analysis of MAP-Elites Performance. *arXiv preprint arXiv:1803.05174*.