Memetic Robot Control Evolution and Adaption to Reality

Else-Line Ruud, Eivind Samuelsen, and Kyrre Glette
Department of Informatics, University of Oslo, Norway
Email:{elmruud,eivinsam,kyrrehg}@ifi.uio.no

***Abstract*—Inspired by animals' ability to learn and adapt to changes in their environment during life, hybrid evolutionary algorithms have been developed and successfully applied in a number of research areas. This paper explores the effects of learning combined with a genetic algorithm to evolve control system parameters for a four-legged robot. Here, learning corresponds to the application of a local search algorithm on individuals during evolution. Two types of learning were implemented and tested, i.e. Baldwinian and Lamarckian learning. On the direct results from evolution in simulation, Lamarckian learning showed promising results, with a significant increase in final fitness compared with the results from evolution without learning. Further experiments with learning on the real robot demonstrated an efficient adaptation of the robot gait to the real world environment, and increased the performance to the level measured in simulation. This paper demonstrates that Lamarckian evolution is effective in improving the performance of robot controller evolution, and that the same learning process on the physical robot efficiently reduces the negative impact of the simulation-reality gap.**

I. Introduction

In robotic system design and development, a large number of components must be considered simultaneously, such as the motor system, morphology, control policy, sensory apparatus, etc. [1]. All of these components are closely interdependent and together determine the behavior of the robot, making optimization of each component a challenge; the changing of one component is likely to influence the functioning of the others. One way of dealing with this challenge is to examine the entire robotic system as a whole [2], [3]. Nature has mastered this procedure perfectly through the use of evolution, which has produced a vast number of examples of embodied intelligence. Evolutionary robotics (ER) attempts to recreate evolution as a mechanism instead of its biological results, which is often what has been used as inspiration in mainstream robotics. The use of metaheuristics, that is, evolutionary algorithms (EAs) in this case, is the main difference between ER and mainstream robotics, where machine learning is often used instead to optimize the control policy of a robot [2].

Recent studies on hybrid EAs show that including additional methods during evolution can lead to higher robustness and possibly better solutions [4], [5], e.g., by using local optimization of an individual's behavior between generations. In nature, a similar process is evident. Some living organisms have an *a priori* ability to learn during their lifetime, and are thereby able to adapt to changes that may occur in their environment, hence improving their own fitness during life.

When it comes to combining learning and evolution, two main models have been applied so far, Lamarckian and Baldwinian learning [4], [6]. Although based on an evolutionary theory that has been more or less rejected as a correct description of biological evolution, Lamarckian learning has been the most successful model in combination with evolutionary algorithms [4], due to the fact that the results of the local improvements are placed back in the population after learning. It should be noted that recently there have been some reports which observe a Lamarckian type of evolution happening in nature as well, such as mice inheriting sensitivity to odors related to parents' fear experiences [7]. Baldwinian learning, inspired by the Baldwin effect [8], only stores the learning potential of a solution, thereby focusing on individuals that may have better learning abilities. Much of the research on Baldwinian learning revolves around how this can aid evolution [9], [10], [11], and whether the stored potential can lead to a predisposition to learn, both during and after evolution [5]. Other work on combining evolution and learning studies the correlations between the fitness and the learning task, and how this affects the search performance [12].

While most research in evolutionary and adaptive robotics has been done on evolution and learning separately, there has also been some exploration of the effects of combining the two. Work combining evolution and reinforcement learning for a simulated mobile robot shows that such a combination outperforms systems which uses either only evolution or only learning [13]. In [14], simulated mobile robots combine evolution and learning on neural network controllers. It is shown that when using a Baldwinian learning mechanism, the neural networks evolve a predisposition to learn, both in stable and unstable environments. Using a real mobile robot, [11] demonstrates evolution of plastic agents with neural network control systems, where the robot is allowed to explore and learn from its environment in periods during evolution.

Most examples of ER research on the combination of evolution and learning have concentrated on neural network learning [5]. However, scenarios like evolution of morphology or parametric control systems may call for other types of learning, such as local optimization using a local search algorithm [15].

In many cases the performance of solutions evolved in simulation is considerably worse when transferred to the physical robot [16], [17]. This phenomenon is often referred to as the reality gap, and is thought to be largely caused

by inaccuracies in the simulator, combined with evolution's tendency to produce overfit solutions. Thus, evolution may produce solutions which exploit features nonexistent in the real world.

Proposed methods to deal with the reality gap include introducing noise in the simulations to make the evolved behavior more robust [16], [18], improving the simulator through real world sampling [19], [20], [21], or concentrating around behaviors which are known to match well in both simulation and reality [22]. Another approach would be to continue evolution or learning after transferral from simulation to reality, in order to adapt to the new environment. While there are several approaches doing full on-board evolution [23], [24], there could be potential for exploring further a combination of simulated evolution and post-simulation learning. In [25] an approach of continued evolution after transferral to reality shows promising results.

We have earlier conducted experiments on evolving a combination of legged robot morphologies and control systems in simulation and then constructing a selection of these in reality [26], [27], [28]. A recent investigation on the difficulty of co-evolving morphology and control in virtual creatures suggest that mutations in the morphology may cause large variations in the behavior of the controller [29], and this is in line with our own observations. One possible way of tackling this challenge could be to introduce a learning phase for the controller in order to allow it to adjust to the new morphology. The resulting gaits from our earlier experiments [28] are efficient but highly dynamic and thus prone to larger reality gap effects, and there could be potential for a learning phase in reality to adjust to the new environment.

Thus, in this paper we take some first steps in investigating the effects of applying a hybrid evolutionary algorithm in this new context of legged robots which have relatively complex dynamics: We apply learning in the evolutionary search for the control system, and also for adjusting to the real world. While we for practical reasons keep the morphology fixed throughout these experiments, by using one of the results from [28] we hope to gain some insight into whether this could in the future be applied to the full morphology-controller search.

The hybrid EA in question is a memetic algorithm [30], which consists of a combination of a genetic algorithm and a local search algorithm. The investigation involves testing the performance of evolution with and without learning under equal conditions, followed by a comparison of the results. The hypothesis is that learning can aid evolution by moving individuals towards local optima during evolution, thereby increasing the probability of discovering more fine-tuned solutions. The results from simulation do indeed indicate that it is more efficient to include learning in the evolutionary process, compared to an approach without learning.

After running the evolutionary algorithm with learning in a simulated environment, we then proceed to re-evaluate selected results more extensively, first in the simulator and then on a real robot. We then demonstrate that it is possible to efficiently adapt to the real world by running the same learning process on the real robot, resulting in performances similar to those measured in simulation.

The remainder of the paper is structured as follows: Section II presents the robot platform used in the experiments, as well as the evolutionary algorithm used to evolve the control system parameters. In Section III, the experiments are described, followed by reports of the results. Section IV contains a discussion of the results, while Section V finally gives a conclusion and possibilities for future work.

II. Robot and Implementation

This section presents the robot platform and control system used in the experiments, as well as the evolutionary algorithm and the local search algorithm which were used to perform the parameter optimization. [1]

A. The robot

The experiments were done using a four-legged modular robot which was produced in a previous experiment: A large number of robots were evolved in a simulated environment with a genotype inspired by a high-level genetic coding found in nature [26]. The encoding specified the shape of a parametrized body section template, together with a set of parameter values specifying multiple instances of the template. This resulted in robots of varying size and shape, which were grouped using a clustering method based on a morphological distance measure [28].

The representative from the third cluster from [28] was used, because it was a conventional quadruped design with two joints per limb, enabling effective movement strategies while keeping the degrees of freedom low. The robot has nine degrees of freedom, made up of nine revolute joints. Each part of the robot, excluding the joints, consists of a white capsule of a specific length and radius, as shown in Figure 1.

For the physical version of the robot, these were 3D printed in the form of hollow plastic capsules with sockets for attaching motors in the joints. The joints were actuated by Dynamixel AX-18A servo motors, and the joint constraints used in the simulator were chosen to simulate the torque and joint friction of this motor.

B. Control function and genome

The control system used to govern the movement of each joint of the robot is inspired by a simple open-loop system proposed in [31], where the joint position γ at time t is given by the following function:

$$\gamma_{\alpha,\phi}(t) = \alpha \tanh(4 \sin(2\pi(t+\phi)))$$

where α and ϕ represent amplitude and phase shift, respectively. This amplitude-phase control system produces periodical signals with a period of 1 s. The tanh function is used to keep the signal constant in parts of the cycle, thereby allowing the robot to stabilize itself in these periods.

[1] Experiment source code: http://folk.uio.no/kyrrehg/evorob/memetic/

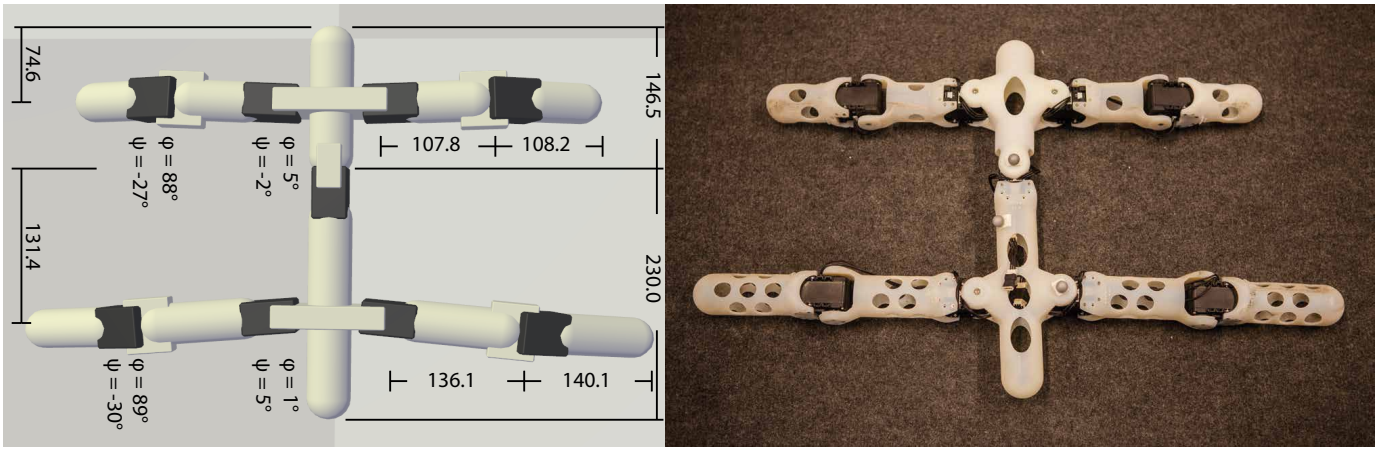

Fig. 1: Top view of the robot. The simulated robot can be seen to the left, and the real-world version to the right. All lengths are in millimeters.

The search space is decreased by introducing symmetry to the system. The limb joints are treated in pairs of left and right side joints. The joints in these pairs have separate phase parameters, but share amplitude parameter, reducing the number of parameters to three per pair. Such a simplification is supported by observations from nature, where this kind of symmetry is prevalent.

C. The evolutionary algorithm

A memetic algorithm was used in order to optimize the robot control system parameters, combining an evolutionary algorithm with a simpler local search algorithm. For evolution we use a real-valued genetic algorithm. Parent selection is done by binary tournaments, and survivor selection is done by truncation, keeping the best individuals.

A mutation operator using a Gaussian distribution $\mathcal{N}(0, 0.1^2)$ was used, from which values were drawn and added to each gene, with probability one. No recombination operator was used, since this was assumed to have a possible disruptive effect on the contribution from the local search. To generate the initial population, μ individuals with all control parameters set to zero were created, μ being the population size. Mutation was then performed ten times on each individual, in order to obtain a random initial population.

The control system parameters were optimized for maximum forward movement. Fitness was measured as the average speed of the robot, calculated from the measured forward displacement over a set time period. Forward was defined as the direction in which the robot was oriented at the beginning of the evaluation period.

Because of the unknown fitness landscape, predicting the degree of exploration provided by the genetic algorithm is difficult, but we hope that the random initialization of the population and the stochastic mutation operator will handle this sufficiently. However, due to these stochastic properties, the GA often fails to exploit the local structures of the fitness landscape, which is then left to the cooperating local search algorithm. The performance of a memetic algorithm largely depends on the choice of local search algorithm and the shape of the fitness landscape [32]. Although the fitness landscape is unknown in our case, we would expect a hill climbing procedure to be a suitable choice, based on the fact that the learning duration for each individual is limited by the learning cost.

In these experiments, an $(1+\lambda)$ evolution strategy was used [33] as the local search algorithm. It searches by generating a generation of solutions of size λ in the vicinity of the initial solution, and selects the best one from which a new generation of solutions is generated. The new solutions are generated through mutation of the selected solution, by adding a number drawn from a normal distribution $\mathcal{N}(0, \sigma^2)$ to every control parameter. σ is mutated alongside the solutions.

It was assumed that a deep local search could be advantageous in these experiments, which should be achieved with a low lambda value. Thus, a $\lambda = 2$ was selected. In order to keep the number of tunable parameters low, this value was fixed in all experiments.

Two types of learning were implemented for the evolution, Baldwinian and Lamarckian learning. Baldwinian learning stores only the fitness value of the best solution discovered during the local search, thus only storing a possible learning potential for that individual. During evolution, individuals in a generation will therefore acquire a new fitness value at the end of the learning phase, if the local search succeeded in locating a better solution. While the actual solution discovered is not stored using Baldwinian learning, Lamarckian learning stores both the learned control system parameters and the fitness value and returns these to the population. In other words, the individual's genes are replaced with the learned solution, and the fitness value set to the new evaluated value.

III. Experiments and Results

In this section, the experiments and their results are presented. A description of the experimental setup is given in the first subsection, before the results from evolution of the robot control system parameters is presented in Subsection III-B.

General	PhysX version	3.3 beta-2		
	Timestep	$^1/_{128}$ s		
Friction	Env. static	0.20	Robot static	0.30
	Env. dynamic	0.15	Robot dynamic	0.30
	Env. restitution	0.4	Robot restitution	0.30
Motor	Static friction	0.15 Nm	Dynamic friction	$\frac{1.65\text{Nm}}{97\text{rpm}}$
	Appliable torque	1.8 Nm		
Obstacle	Width	0.02 m	Height	0.02 m
	Length	0.02 m	Spacing	0.5 m

TABLE I: Simulation parameters

Population size (μ)	40
Mutation size (σ)	0.1
Mutation probability per gene	1
Total number of evaluations	1600
Evaluation time	8 s
Total number of runs	180
Iterations of local search	10, 20
Local search parameter (λ)	2

TABLE II: Evolution parameters

Finally, in Subsection III-C, the results from the experiments where learning was applied on the real-world robot are reported.[2]

A. Experiment setup

The simulation experiments were conducted in a custom-built simulator using the PhysX physics simulation engine. Table I details the simulation and environment parameters. A custom set of joint constraints were used to simulate the motors in the robot joints [28].

The real-world experiments were done using a motion capture system for position and orientation measurements, consisting of 12 OptiTrack Flex 3 infrared cameras and reflective markers placed on the robot. For efficiency, the robot was programmed to turn around when it moved out of bounds, by replacing the control system with one that turned it either left or right until facing the center of the floor. For simplicity, the robot was controlled using an off-board computer.

Five configurations of robot control system parameters were evolved in the simulator, one without learning, and four with Baldwinian and Lamarckian learning, with 10 and 20 iterations of local search for both. One iteration corresponds to one evaluation of a new solution during the local search. An evaluation of a solution involved letting the robot move freely in the simulator for 8 seconds using the control system parameters in that solution, and then measuring the forward displacement of the head of the robot.

Each evolutionary run had a population size of 40 individuals, where each individual was evaluated 1600 times. As one iteration of local search involves doing one evaluation, an increasing number of iterations of local search must result in a decreasing number of generations to allow for comparison between configurations. With a total of 1600 evaluations available for each individual, evolution without learning lasts for 1600 generations, while evolution with 20 iterations of local search for each individual lasts 80 generations, assuming the population size is equal. Each set of robot control system parameters consisted of the results from 30 evolutionary runs, except for the configuration without learning, which was done over 60 runs in order to increase data for comparison with the other configurations. A summary of the evolution parameters is shown in Table II.

From the populations in the final generation of the evolved robot control parameters, 25 solutions were selected for further testing, five from each configuration. Solutions from the 90th, 70th, 50th, 30th and 10th percentiles were picked to represent the span of solutions found. These 25 solutions were then evaluated in the simulator and on the real-world robot, where the robot was allowed to move freely in its environment over 30 evaluations of 4 seconds each. This was done to get a better estimate of how well the solutions performed over time, as the fitness value alone might not be accurate enough as a description of the performance.

Each solution was then improved on the real-world robot by applying learning to it while running, in the form of the same local search algorithm that was used in the evolution. The best solution found after 20 iterations of local search was then evaluated in the same manner as was done on the solution before learning. With a $\lambda = 2$, one generation of $(1 + \lambda)$-ES involves evaluation of two individuals, and 20 iterations of local search correspond to 10 generations. Because of the elements of randomness in the local search algorithm, this was repeated five times for each solution, starting from the same evolved solution each time. The after-learning results thus consist of a merger of the evaluations of the best solutions found in each of the five learning runs.

B. Evolution with learning in simulation

Figure 2 shows the results from evolution, where the mean of the best fitness in each evolutionary run between the 30 runs has been plotted for each configuration. We can see that Lamarckian learning leads to better fitness values than Baldwinian learning. In addition, Lamarckian learning seems to yield better final results than without learning. The difference between Lamarckian learning with 20 iterations and no learning in the final generation is significant, based on a two-tailed Wilcoxon rank-sum test resulting in a *p*-value smaller than 0.01. The results from the configuration with 10 iterations of Lamarckian learning are not significantly larger than the results without learning, but there is a certain tendency towards slightly better performance.

Baldwinian learning performs significantly worse than Lamarckian learning and no learning ($p < 0.01$, Wilcoxon rank-sum test), with the exception of 10 iterations Baldwinian against no learning where no significant difference is detected.

[2]Videos showing example evolved gaits from simulation and reality can be found at http://folk.uio.no/kyrrehg/evorob/memetic/

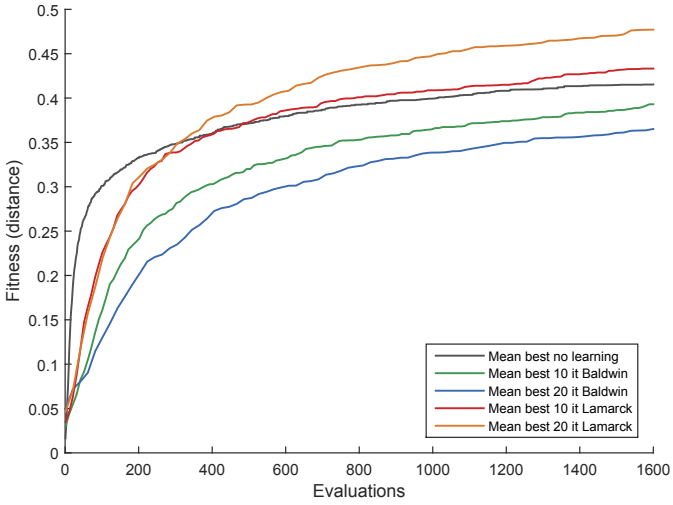

Fig. 2: Means of the best fitnesses over an increasing number of evaluations. The configurations include evolution without learning and both Baldwinian and Lamarckian learning, over a varying number of iterations of local search.

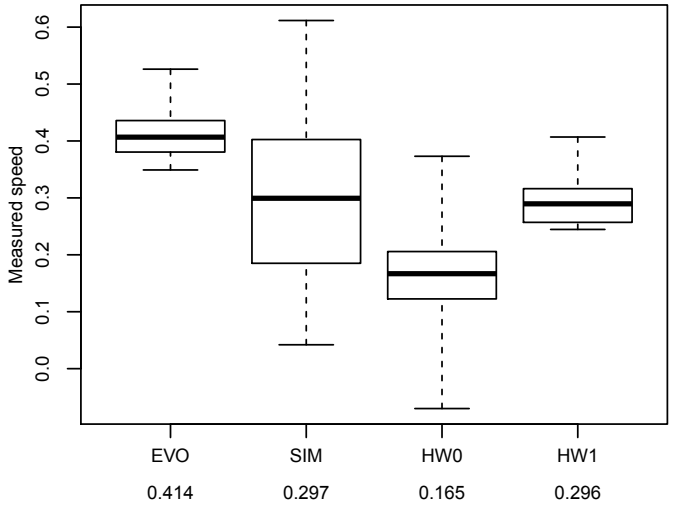

Fig. 3: Speed measurements, adjusted for percentile. Mean values are noted beneath the labels. EVO, SIM, HW0 and HW1 denote fitness values after evolution, re-evaluations in simulation, and the results from evaluations on the real-world robot before and after learning, respectively.

C. Re-evaluation and re-learning

The performances of the 25 selected solutions in the different environments are represented in Figure 3. The first box, EVO, indicates the fitness score from the evolutionary search, while SIM indicates the score from the extended evaluation in simulation. HW0 and HW1 indicate the score on the real-world robot before and after learning, respectively. The solution scores have been compensated for the percentile they are in by subtracting the difference between the mean of all measurements in that percentile and that of the 50th percentile.

The decrease in average performance from the final fitness values from evolution (EVO) to re-evaluations in simulation (SIM) is 28%, and it decreases further from SIM to HW0 with 44%. However, from HW0 to HW1, there is an increase in average performance of 79%. With five learning runs of 20 iterations each for each gait, the duration of active learning was less than seven minutes, not counting additional overhead such as setting up the robot. With the exception of SIM vs. HW1, all average differences in performance are statistically significant ($p < 0.01$, T-test).

IV. Discussion

Based on the results, we discuss the experimental results along two main topics: The effects of applying learning in the simulation-based evolutionary search, and the effects observed when re-evaluating the results and performing real world learning.

A. Evolution with learning in simulation

The direct results from evolution with the Lamarckian learning configurations indicate that a certain minimum number of iterations of local search are necessary before the effects of the local search are beneficial over the baseline genetic algorithm. With 20 iterations, the search can reach further than with 10 iterations, and thus does more local exploration which increases the chances of moving towards a local optimum. This seems to be an advantage over evolutionary selection over more generations.

Figure 2 also shows that the lifetime learning configurations take more evaluations to converge than the baseline, probably due to the fact that global exploration through generations happens at a lower rate due to the evaluations spent on local search.

The difference between Lamarckian learning and Baldwinian learning during evolution is as expected. Baldwinian learning obviously performs local exploration as well, but since it only stores the new fitness value and not the discovered results in the control parameters, the fitness value only indicates if it is close to a good solution or not. This seems not to be enough to outperform evolution with no learning, probably because a good solution that was discovered during local search will only be used if it is found by chance during evolution, because of the randomness in the variation operators.

While the Baldwinian search in this case does not perform better than the baseline for the given evaluation budget, the mechanism of storing the learning potential still seems to have some effect on the evolutionary search. This can be seen from Figure 2, where the fitness values are higher than evolution without learning for an equal number of *generations*. For example, at 1600 evaluations, the 20-iteration Baldwinian evolution configuration has performed 80 generations, with a higher average fitness than the configuration without learning at 80 generations.

While it was not surprising that the Lamarckian scheme performed well, the environment in the evolutionary experi-

ments was static. The introduction of more dynamic environments could possibly favor Baldwinian learning, as indicated in [34]. It would also be interesting to evaluate Baldwinian learning during simultaneous evolution of morphologies and controllers, as changes in the morphology may be seen as a change in the controller's interface to the environment [29].

B. Re-evaluations and real world learning

The results from evolution and the extended re-evaluations are evaluated in the same environment, using the same evaluation method. However, when comparing the measurements (EVO and SIM in Figure 3), we see that there is a large discrepancy, both in variance and average performance. Since these are highly dynamic gaits, being open-loop and displaying fast movement, we can expect significant variation in performance, with sensitivity to initial conditions. We hypothesize that the cause of the performance discrepancy is selection bias; individuals with lucky fitness evaluations tend to accumulate across generations, with fitnesses somewhat misleading for their actual performance. With SIM as a better representation of the true simulation performance of the robots, we can use the difference between SIM and HW0 as a measure of the reality gap related to inaccuracies in the simulated environment.

The experimental results show that learning on the real robot can efficiently adapt a robot gait to the real world environment. After an initial performance drop going from simulation (SIM) to real world (HW0) evaluation, learning proves to be an efficient method to regain performance, as can be seen in Fig.3. Relatively few evaluations were performed to achieve this increase in performance, with a total of 100 evaluations for each robot gait. With evaluation time taken into account, this involves less than 7 minutes of learning.

However, this increase in performance after learning can be regarded as more of an adaptation to the new environment rather than a *reduction* of the reality gap, as there is no guarantee that the result of learning is closer to what we saw in simulation. In that sense, evolution in simulation provides solutions with a good starting point for further adaptation to the real world environment, and does not necessarily produce solutions that will have a real world performance as close to the simulator performance as possible.

While we collected data from the different memetic learning methods in the different re-evaluation and re-learning environments, these results were combined in Fig.3. Our experiments did not reveal any clear differences in learning abilities on the real robot related to the learning methods, and a larger number of real world measurements are required in order to discover such correlations, if they exist.

V. Conclusion and future work

In this paper we applied a memetic algorithm for evolution of control system parameters for a legged robot with highly dynamic locomotion. The algorithm consists of a merger between a standard genetic algorithm and a local search algorithm, and was implemented with two models of learning, Lamarckian and Baldwinian.

The algorithm was tested in simulation over different configurations of these models. The results show that when an adequate number of iterations were used, Lamarckian learning had a positive effect on the results from evolution, with a significant improvement over evolution without learning. This is a promising result which should encourage further studies with lifetime learning when evolving controllers for robots with dynamic locomotion, and also when evolving morphologies and controllers simultaneously. Baldwinian learning performed, in our experiments, significantly worse than both evolution with Lamarckian learning and evolution without learning. For future work it would be interesting to investigate if this changes when environments are more dynamic, or when learning is applied in combination with evolution of morphology.

In our experiments the fitness evaluations could have large variations for similar controllers, related to the highly dynamic gaits and sensitivity to initial conditions. Re-evaluation of the evolved solutions in simulation showed that the final fitness values after evolution were not necessarily an accurate measure of the actual performance, due to the selection bias for lucky individuals introduced by evolution. Measures to reduce this effect should be investigated, such as re-evaluation schemes [23] or a more robust fitness evaluation procedure. However this typically involves spending more time on evaluation of each individual, and the tradeoff between evaluation robustness and total number of search iterations should be studied carefully.

Real world learning proved to be an efficient method to regain performance on the real world robot. Using relatively few evaluations, the learning process adapted the robot gait to the new real world environment. Thus, it might be interesting to see if the adaptation ability learning seems to have in our experiments is equally strong if other environments are introduced. Future work could investigate the effects of introducing different real world environments to the evolved robot gaits, perhaps with obstacles. This could also be interesting to test in simulation, both as post-evolution learning in different environments and introduction of new simulated environments during evolution with and without learning. No clear tendencies in differences in learning ability between the memetic learning methods were found in our experiments, however, it seems likely that such differences would exist. In order to investigate this further, a larger data set should be generated for further statistical analysis.

References

[1] S. Doncieux, N. Bredeche, J.-B. Mouret, and A. E. G. Eiben, "Evolutionary robotics: what, why, and where to," *Frontiers in Robotics and AI*, vol. 2, p. 4, 2015.

[2] J. C. Bongard, "Evolutionary robotics," *Communications of the ACM*, vol. 56, no. 8, pp. 74–83, 2013.

[3] K. Sims, "Evolving virtual creatures," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1994, pp. 15–22.

[4] M. N. Le, Y.-S. Ong, Y. Jin, and B. Sendhoff, "Lamarckian memetic algorithms: local optimum and connectivity structure analysis," *Memetic Computing*, vol. 1, no. 3, pp. 175–190, 2009.

[5] S. Nolfi and D. Floreano, "Learning and evolution," *Autonomous Robots*, vol. 7, no. 1, pp. 89–113, 1999.

[6] D. Whitley, V. S. Gordon, and K. Mathias, "Lamarckian evolution, the baldwin effect and function optimization," in *Parallel Problem Solving from Nature-PPSN III*. Springer, 1994, pp. 5–15.

[7] B. G. Dias and K. J. Ressler, "Parental olfactory experience influences behavior and neural structure in subsequent generations." *Nature neuroscience*, vol. 17, no. 1, pp. 89–96, 2014.

[8] J. M. Baldwin, "A new factor in evolution," *American naturalist*, pp. 536–553, 1896.

[9] G. E. Hinton and S. J. Nowlan, "How learning can guide evolution," *Complex systems*, vol. 1, no. 3, pp. 495–502, 1987.

[10] D. Ackley and M. Littman, "Interactions between learning and evolution," in *Artificial Life II, SFI Studies in the Sciences of Complexity*, 1991, vol. 10, pp. 487–509.

[11] D. Floreano and F. Mondada, "Evolution of plastic neurocontrollers for situated agents," in *From Animals to Animats 4, Proceedings of the 4th International Conference on Simulation of Adaptive Behavior (SAB" 1996)*. MA: MIT Press, 1996, pp. 402–410.

[12] S. Nolfi, D. Parisi, and J. L. Elman, "Learning and Evolution in Neural Networks," *Adaptive Behavior*, vol. 3, no. 1, pp. 5–28, 1994.

[13] M. Schembri, M. Mirolli, and G. Baldassarre, "Evolution and learning in an intrinsically motivated reinforcement learning robot," *Advances in Artificial Life*, pp. 294–303, 2007.

[14] S. Nolfi and D. Parisi, "Learning to Adapt to Changing Environments in Evolving Neural Networks," *Adaptive Behavior*, vol. 5, no. 1, pp. 75–98, 1996.

[15] F. Neri and E. Mininno, "Memetic compact differential evolution for cartesian robot control," *Computational Intelligence Magazine, IEEE*, vol. 5, no. 2, pp. 54–65, 2010.

[16] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *Advances in Artificial Life*. Springer, 1995, pp. 704–720.

[17] K. Glette, G. Klaus, J. C. Zagal, and J. Torresen, "Evolution of locomotion in a simulated quadruped robot and transferral to reality," in *Proceedings of the Seventeenth International Symposium on Artificial Life and Robotics*, 2012.

[18] K. Glette, A. L. Johnsen, and E. Samuelsen, "Filling the reality gap: Using obstacles to promote robust gaits in evolutionary robotics," in *Proceedings of the 2014 IEEE International Conference on Evolvable Systems (ICES)*. IEEE, 2014, pp. 181–186.

[19] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[20] J. C. Zagal and J. Ruiz-Del-Solar, "Combining simulation and reality in evolutionary robotics," *Journal of Intelligent and Robotic Systems*, vol. 50, no. 1, pp. 19–39, 2007.

[21] G. Klaus, K. Glette, and J. Torresen, "A comparison of sampling strategies for parameter estimation of a robot simulator," in *Simulation, Modeling, and Programming for Autonomous Robots*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7628, pp. 173–184.

[22] S. Koos, J.-B. Mouret, and S. Doncieux, "The transferability approach: Crossing the reality gap in evolutionary robotics," *Evolutionary Computation, IEEE Transactions on*, vol. 17, no. 1, pp. 122–145, Feb 2013.

[23] J.-M. Montanier and N. Bredeche, "Embedded evolutionary robotics: The (1+ 1)-restart-online adaptation algorithm," in *New Horizons in Evolutionary Robotics*. Springer, 2011, pp. 155–169.

[24] V. Zykov, J. C. Bongard, and H. Lipson, "Evolving dynamic gaits on a physical robot," in *Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO*, 2004.

[25] S. Nolfi and D. Floreano, "How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics," in *Artificial life IV*, vol. 1997, 1994, pp. 190–197.

[26] E. Samuelsen, K. Glette, and J. Torresen, "A hox gene inspired generative approach to evolving robot morphology," in *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*. ACM, 2013, pp. 751–758.

[27] E. Samuelsen and K. Glette, "Some distance measures for morphological diversification in generative evolutionary robotics," in *Proceedings of the 2014 conference on Genetic and evolutionary computation*. ACM, 2014, pp. 721–728.

[28] ——, "Real-world reproduction of evolved robot morphologies: Automated categorization and evaluation," in *Applications of Evolutionary Computation - 18th European Conference, EvoApplications 2015*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2015.

[29] N. Cheney, J. Bongard, V. Sunspiral, and H. Lipson, "On the Difficulty of Co-Optimizing Morphology and Control in Evolved Virtual Creatures," in *Proceedings of the Artificial Life Conference 2016*. MIT Press, 2016, pp. 226–233.

[30] N. Krasnogor, "Memetic algorithms," in *Handbook of Natural Computing*. Springer, 2012, pp. 905–935.

[31] S. Koos, A. Cully, and J.-B. Mouret, "Fast damage recovery in robotics with the t-resilience algorithm," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1700–1723, 2013.

[32] Y. S. Ong and A. J. Keane, "Meta-lamarckian learning in memetic algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 2, pp. 99–110, 2004.

[33] H.-G. Beyer, "Toward a Theory of Evolution Strategies: Some Asymptotical Results from the (1,+ λ)-Theory," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 2, pp. 165–188, 1993.

[34] T. Sasaki and M. Tokoro, "Adaptation toward Changing Environments: Why Darwinian in Nature?" in *Fourth European Conference on Artificial Life (ECAL97)*, 1997, pp. 145–153.