Three-fold Adaptivity in Groups of Robots: The Effect of Social Learning

Jacqueline Heinerman
VU University Amsterdam
Amsterdam, The Netherlands
j.v.heinerman@vu.nl

Dexter Drupsteen
University of Amsterdam
Amsterdam, The Netherlands
dexter.drupsteen@gmail.com

A.E. Eiben
VU University Amsterdam
Amsterdam, The Netherlands
a.e.eiben@vu.nl

ABSTRACT

Adapting the control systems of robots on the fly is important in robotic systems of the future. In this paper we present and investigate a three-fold adaptive system based on evolution, individual and social learning in a group of robots and report on a proof-of-concept study based on e-pucks. We distinguish inheritable and learnable components in the robots' makeup, specify and implement operators for evolution, learning and social learning, and test the system in an arena where the task is to learn to avoid obstacles. In particular, we make the sensory layout evolvable, the locomotion control system learnable and investigate the effects of including social learning in the 'adaptation engine'. Our simulation experiments demonstrate that the full mix of three adaptive mechanisms is practicable and that adding social learning leads to better controllers faster.

Keywords

Evolutionary robotics; on-line evolution; individual learning; social learning; obstacle avoidance; neural networks

1. INTRODUCTION

The importance of adapting the control systems of robots on the fly in robotic systems of the future has been emphasised in for example [16] and [20]. Robots with adaptive controllers are able to adjust to circumstances not (fully) known in advance during their design time and/or when the circumstances, e.g. the environment or user preferences, change.

In this paper we adopt the conceptual framework presented in [13], distinguishing three types of adaption within population-based adaptive systems in general: evolution, lifetime learning and social learning. The differences between these mechanisms are based on distinguishing inheritable and learnable features of the individuals and postulating that inheritable properties do not change during an individuals lifetime, while learnable properties do. Consequently, we distinguish evolutionary operators that act on inheritable features and learning operators adjusting the learnable properties. Furthermore, we distinguish individual learning that can be done by a single individual alone and social learning that requires the exchange of knowledge between more individuals.

Obviously, the difference between inheritable and learnable properties depends on the application and the feasibility of implementing reproduction. This latter one is non-trivial when using real robots as the technology for truly embodied evolution is just emerging [8, 9, 22]. The motivation behind this paper is grounded in real robotics, envisioning robot populations that can adapt by all three mechanisms mentioned above. Therefore, we chose to base our experiments on real robots (e-puck), where we designate the sensory apparatus as inheritable/evolvable. This is technically feasible, because all robots actually do have all sensors, but depending on their 'genome' they may use only some of them. This makes the hardware makeup flexible enough for experimentation. In the meanwhile, this is also a sound design, where the minds of the robots (learnable controllers) must match their bodies (the evolvable sensory layout).

Given such a population of e-pucks we investigate the feasibility of integrating evolution, individual learning, and social learning in a global 'adaptation engine' and look more closely into social learning. In particular, we include a social learning mechanism that allows robots to share their individually learned controllers and test whether social learning works if the hardware of the robots (here: sensory layout) is different. In robot populations with homogeneous hardware we expect that sharing individually learned knowledge will improve adaptation. However, in populations with heterogeneous hardware this is not so obvious, because the learned controllers are, in principle, hardware specific. Thus, our experiments will seek answers to the following three questions:

1. What is the effect of social learning on the quality and speed of learning in a robot population with homogeneous sensory layout (no evolution)?
2. What is the effect of social learning on the quality and speed of learning in a robot population with heterogeneous sensory layout (no evolution)?
3. What is the effect of social learning on the quality and speed of learning in a robot population with evolving sensory layout?

GECCO '15, July 11–15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739480.2754743

2. RELATED WORK

Evolutionary Robotics (ER) is a successful field with a history of a decade and a half [3, 12, 17, 7, 19]. The majority of work in ER employs a rather conventional evolutionary algorithm to evolve controllers, morphologies, or both for robots in some environment(s) in an off-line fashion. That is to say, the design time and the operational time of the robots are separated and evolution only takes place in the design stage. After deployment the robots are not evolving anymore.

In this paper we follow an alternative approach, that of on-line adaption as we will have evolution, individual learning, and social learning all take place during the operational period of the robots. On-line evolution in a population of robots has been first demonstrated in [21]. In this paper physical robots broadcast parameter values from the robot's control specification at a rate proportional to their fitness. Since then, multiple distributed setups have been proposed with and without the use of an explicit fitness function, see for example [5], [10], [15], and adding maturation time [24].

The main difference between existing work and the current paper is the use of a three-fold adaptation mechanism. Technically this means a separation between inheritable and learnable features and the corresponding evolutionary and learning operators, as explained in the Introduction. This distinction is not new per se, a distinction between genes and memes and genotypes and memotypes has been made earlier, for instance, in [14], [11], and [23]. It is interesting to note that social learning can be perceived as a Darwinian process in the memotype space with selection (of memes to send/receive) and recombination (of memes of the sender and the recipient). Individual learning can then be considered the informed mutation operator that changes the memotype of one individual.

Another cluster of related works concerns the evolvability of sensory layout, typically related to finding the best sensory layout to increase autonomy, adaptivity and a better understanding of how sensors evolve in nature [6]. Restricting the sensory layout to only the important sensors, results in a smaller search space and thus a faster learning process [1]. To our best knowledge, investigating the effect of social learning in this setup has not been done.

3. SYSTEM DESCRIPTION

We use e-puck robots as shown in Figure 1 in an arena with the task being obstacle avoidance. The performance of a robot measured over an evaluation time T is defined after [4] as follows:

$$f = \sum_{t=0}^{T} s_{trans} \times (1 - s_{rot}) \times (1 - v_{sens}),$$

where s_{trans} is the translational speed (not normalised) and s_{rot} the rotational speed (between 0 and 1). v_{sens} is the minimal distance to the nearest obstacle (between 0 and 1). In other words, the fitness is maximized when there is a straight movement and minimal sensory input.

3.1 Inheritable and learnable features

The e-puck is a small (7 cm) differential wheeled mobile robot that is equipped with 8 Infra-Red (IR) proximity sensors to be able to detect obstacles. We designate the sensory apparatus as inheritable/evolvable. Therefore, the *genotype* is an array with length 8 where g_i is 0 when sensor i is disabled and 1 if it is used. By definition, a genotype does not change during lifetime. Therefore, a genome can be seen as the ID of an individual and we consider the same e-puck with another genome another individual from the perspective of evolution. This implies that the lifetime of an individual is the period of having the same genome, i.e., the same sensory layout. Imposing another layout on a given e-puck will make it another individual for the evolutionary process.

Figure 1: The e-puck robot developed at the *École Polythechnique Fédérale de Lausanne* (EPFL).

The learnable controller of the robots, the *memotype* is a neural network (NN) with a variable number of input nodes depending on the genome, no hidden nodes, and two output nodes (motor speed values). This results in a maximum neural network size of 18 weights (including the bias node). The motors are activated based on a hyperbolic tangent activation function. A meme consists of information about the neural network structure and corresponding weights. Again by definition, learnable features can change during the lifetime of a given individual, but they are re-initialized if the genotype changes (assuming non-Lamarckian evolution).

3.2 Adaptive mechanisms

The general idea behind the integrated adaptation mechanism is as follows. Each individual –that is, a robot with a particular sensory layout as specified by its genotype– has a maximum lifetime, allocated at birth. This maximum lifetime can vary between the robots. This results in overlapping generations where the young robots can learn from the more experienced ones through the social learning mechanism. Time for learning and evolution is measured by the number of controller evaluations; the age of a robot is increased by one after every evaluation until the maximum lifetime is reached. Then a new genotype –hence, a new robot individual with a new sensory layout– is created. During a robots life several controllers are created and evaluated by the learning mechanisms.

The whole mechanism is outlined in Algorithm 1, the three adaptive mechanisms that form the components of the system are described in the subsequent subsections.

Algorithm 1 Pseudocode of the 3-fold adaptation method

```
while current evaluation < max. no. of evaluations
  for every generation
    set maximum lifetime of individual randomly
    if first generation or no genotypes collected
      initialize genotype randomly
    else
      select a mate from collected genotypes
      create child from mate and current genotype
    end if
    initialize memotype randomly
    while individual age < maximum lifetime
      choose action: individual learning /
      social learning / reevaluate champion
      broadcast genotype and fitness
      broadcast memotype if exceeding threshold
      receive genotypes and memotypes
    end while
  end for
end while
```

3.2.1 Evolution.

Genotypes are broadcasted to all other robots after every controller evaluation together with the fitness value. Robots are collecting these genotypes in their *genotype storage*. Only unique genes are stored here together with the corresponding fitness value. When the genome is already in storage, the fitness value will be replaced by the last obtained fitness value. When a robot's lifetime expires it picks a new genome through tournament selection from the genotype storage. Uniform crossover and mutation are performed on the genome of the tournament winner and the current genome of the robot. When a new genome is established, the genotype storage is cleaned an a new memotype is created where the weights of the neural network are uniform random initialized.

3.2.2 Individual Learning.

The method for individual learning can be any algorithm that can optimize neural networks efficiently. In our system it is a (1+1) Evolutionary Strategy based on [4]. The fitness function for this ES is the f defined in the beginning of this section. The algorithm works on the weights of the neural network that are mutated with a Gaussian noise $N(0, \sigma)$ whose σ value is doubled when the mutated controller (the so-called challenger) is not better than the current controller (the so-called champion). Before a controller is evaluated a recovery period is introduced. During this period, the robot can move, but the fitness is not being measured, so that the robot is able to recover from a difficult starting position. When a challenger has a higher fitness than the champion, the weights of the neural network are replaced resulting in a new controller.

3.2.3 Social Learning.

Memotypes are broadcasted to all other robots after every controller evaluation, provided that a minimum fitness threshold is exceeded. We have implemented a *low*, a *medium* and a *high* value for this, in particular, 10%, 20% or 30% of the theoretical maximum of the fitness function.[1] The place where memotypes from other robots are collected is called the *memotype storage*. A memotype is taken from the memotype storage in a *Last In First Out (LIFO)* order and combined with the current robots controller to create a challenger. To perform recombination the weights of the current controller are copied into the challenger. After that the challenger weights, now equal to the champion weights, are overridden by those of the collected meme if these weights are applicable to the current genome (i.e. the corresponding sensory layout). The resulting meme is thus a crossover between the champion and the challenger. After evaluation of the challenger, the challenger meme is either discarded, when the fitness is lower than the current champion, or promoted to the current champion.

During an evaluation, one of three possible actions are performed. The robot performs individual learning, social learning or reevaluates its current controller. The fitness value obtained by reevaluation is used to create a new fitness value for the current controller in combination with the old fitness value with an 20-80 weight distribution (where 80% is from the old fitness value). Reevaluation of a champion is necessary because of the noisy fitness function [2]. The proportion of number of executions between the three mentioned actions, individual learning, social learning and reevaluation, is a system parameter and thus can be tuned.

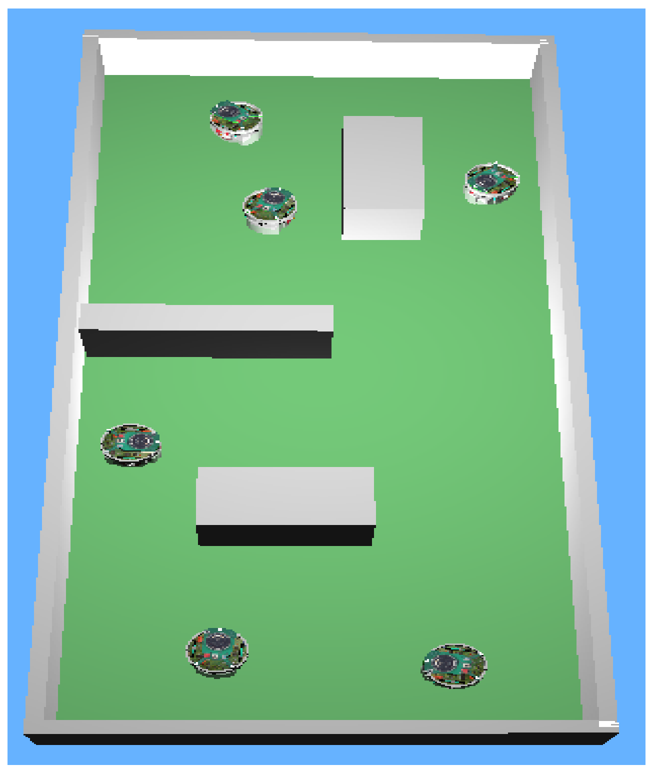

Figure 2: The Webots (version 7.4.2) environment where the group of six e-puck are simulated

[1]The maximum is calculated by assuming a robot moving in a perfectly straight line with no obstacles in sight for the full evaluation period. Let us note that the practically obtainable fitness values are around 40% of this maximum.

4. EXPERIMENTAL SETUP

We conduct our experiments [2] in a high quality simulator, Webots version 7.4.2. We limit the duration of simulations to 3 hours, because the batteries of the e-pucks do not last longer. The simulation environment is exhibited in Figure 2 showing 6 e-pucks. This is the amount of real e-pucks we have, such that we can validate the results in hardware.

The list of all relevant system parameters and the values used in the experiments is given below. When multiple parameter values are used for the experiments, the range is given.

System parameters

`Max. evaluations` Maximum number of evaluations in a run.	`200 - 800`
`Max. robot lifetime` Maximum lifetime of a robot initialized at birth.	`100 - 200`
`Evaluation duration` The duration of one evaluation measured in seconds. (Recovery time is 1.5 sec and actual evaluation is 8.75 sec).	`10.25 sec`
`Reevaluation rate` Chance that the champion is reevaluated.	`0.2`
`Challenger rate` Chance that a challenger is evaluated trough either social learning or individual learning.	`0.8`
`Social learning rate` Chance that a challenger is created by social learning.	`0.3`
`Individual learning rate` Chance that a challenger is created by individual learning.	`0.7`

Evolution

`Disable chance` Chance for each sensor to be disabled at birth.	`0.3`
`Tournament size` Size of tournament that is held among the collected genomes.	`2`
`Mutation chance` Change to enable/disable sensor after recombination.	`0.05`
`Maximum genotype memory` Maximum number of unique collected genomes.	`5`

Learning

`Weight range` Value of NN weights are between [-Weight range, Weight range].	`4`
`Sigma initial` Initial sigma value for mutating weights.	`1`
`Sigma maximum` Maximum sigma value.	`4`
`Sigma minimum` Mimimal sigma value.	`0.01`
`Maximum memotype memory` Maximal size of meme storage.	`20`

[2]The code for implementation is available on https://github.com/jvheinerman/three_fold_adaptivity_algorithm.

Fitness

`Reevaluation weight` Weight of champion fitness in reevaluation.	`0.8`
`Maximum fitness` Theoretical maximum fitness value.	`22400`
`Threshold` Percentage of maximum fitness to exceed before sending meme.	`0-30%`

5. EXPERIMENTAL RESULTS

We have arranged our experiments according to the main research questions listed in the Introduction. Thus, we conducted three series of runs under different conditions regarding the sensory layout of the robots. In each case, we compared the setup with individual learning only and that of individual and social learning together. Furthermore, we varied the threshold value that regulates the quality pressure in the social learning mechanism, cf. Section 3.2.3. This resulted in four social learning variants: no threshold at all, low, medium or high threshold. For every threshold value and the test without social learning 50 repetitions were executed with different random seeds.

In the first experiment all robots have the same layout: all eight proximity sensors are activated and there is no evolution in the genotype space. The duration of the run is 200 controller evaluations. The outcomes are displayed in Figure 3. The graph shows the average fitness (over the 50 repetitions) of the population. The plots clearly show the impact of social learning: it improves the speed of learning as well as the quality of the learned controllers. For every threshold value the average fitness increases with at least 30% (and up to 60%). It is interesting to note that social learning has a positive effect even if there is no threshold for broadcasting memotypes after every evaluation. This is good news, indicating that the method works even in cases where there is not enough information about optimal fitness values to establish a reasonable threshold. To validate this effect we re-plot the experimental results with the 95% confidence intervals for individual learning alone and social learning without threshold in Figure 4. We can see that the confidence intervals do not overlap, meaning a significant difference with the P value much less than 0.05 [18].

In the second experiment the robots have a different layout: each e-puck has different sensors that are activated. These layouts are randomly generated in the beginning of each experiment. The results are shown in Figure 5. In this case, social learning does not result in a much higher quality (average fitness increase is up to 30%), but it does make the learning process faster, reaching the same fitness levels in fewer evaluations then individual learning only. It is interesting to compare the setup with no threshold and with the high threshold. This shows that having no threshold causes a faster increase of the fitness function, while a high threshold results in higher fitness at the end of the 200 evaluations. This indicates that the threshold value can be used to calibrate the system according to user preferences: speed or quality.

In the third experiment the robots have an evolvable layout initialized randomly. To better investigate the effect of social learning we decided to use non overlapping genera-

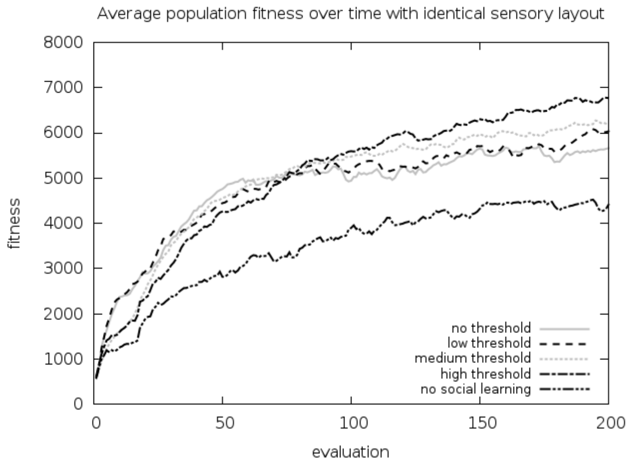

Figure 3: Average population fitness when the sensory layouts of the robots are identical. Time is measured by the number of evaluations (x-axis), fitness by the formula in Section 3 (y-axis). Averages are taken over 50 repetitions.

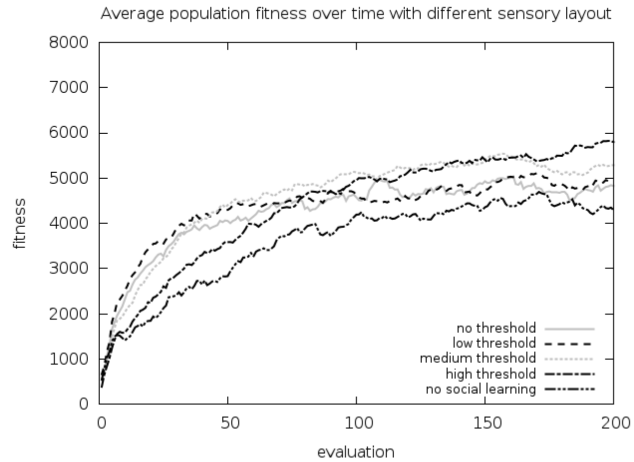

Figure 5: Average population fitness when the sensory layouts of the robots are different. Time is measured by the number of evaluations (x-axis), fitness by the formula in Section 3 (y-axis). Averages are taken over 50 repetitions.

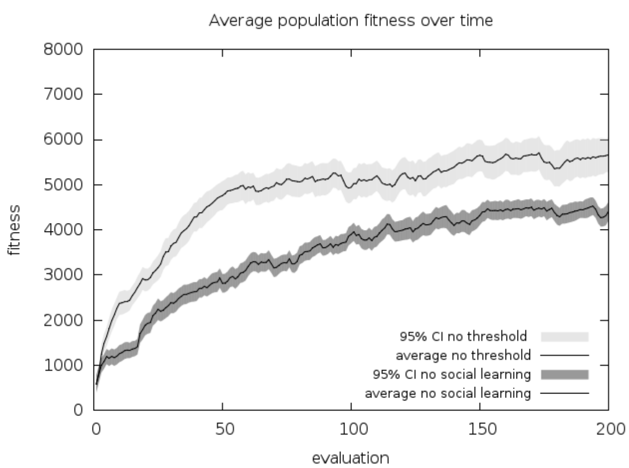

Figure 4: Average population fitness when the sensory layouts of the robots are identical including 95% confidence intervals.

tions, where every robot has the same age. The maximum lifetime is set at 100, i.e., every generation lasts for 100 evaluations, and we have 8 generations all together. Figure 6 displays the results. The first generation shows similar results to Figure 5. Again, we can clearly see a positive effect of social learning. The changes from generation to generation are subtle, but over the whole run we can see a trend: learning within a generation becomes faster.

To inspect the effect of evolution on the genotypes, that is, the sensory layout of the robots., we collected statistics about the status of the sensors that (on or off). The data shows that the number of sensors used decreases over time. The decrease is about 10 to 20 percent, compared to the number of active sensors in the beginning (which varies, depending on the random initialization). Evolution seems to discover that using fewer sensors is advantageous, because it makes the learning task easier. Furthermore, we could observe a difference between the usage of the front and the rear sensors. Intuitively, one would expect that rear sensors become obsolete because they are not necessary for detecting obstacles during forward locomotion. The data provides some confirmation for this, showing that the decrease in the usage of front sensors is lower (around 10%) than the drop for the rear sensors (around 15%). Figure 7 provides an illustration of these effects for two setups, the one with social learning and a low threshold and the one with social learning and a medium threshold.

6. CONCLUSIONS AND FURTHER WORK

In this paper we presented and investigated a three-fold adaptive mechanism based on evolution, individual and social learning to implement on the fly adaptation in a group of robots. The conceptual framework underlying this system is generic, based on distinguishing inheritable and learnable components in the robots makeup and specifying adequate evolutionary and learning operators. Such a system provides a new opportunity to investigate the mutual effects of all these adaptive mechanisms together – an option that existing systems do not offer.

We have implemented this three-fold system in a population of e-puck robots for investigating the effects of social learning. To this end, we designated the sensory apparatus as inheritable/evolvable and the NN-based controllers as learnable. The experiments clearly indicated the benefits of social learning: it makes the population learn faster and the quality of learned controllers higher. This effect is demonstrated under three different setups: for robots with identical sensory layout, for robots with different sensory layout and for robots with evolving sensory layout. We have also found that the social learning mechanism can be tuned for speed or controller quality by a simple parameter, the quality threshold maintained for sending around a robot's controller to others.

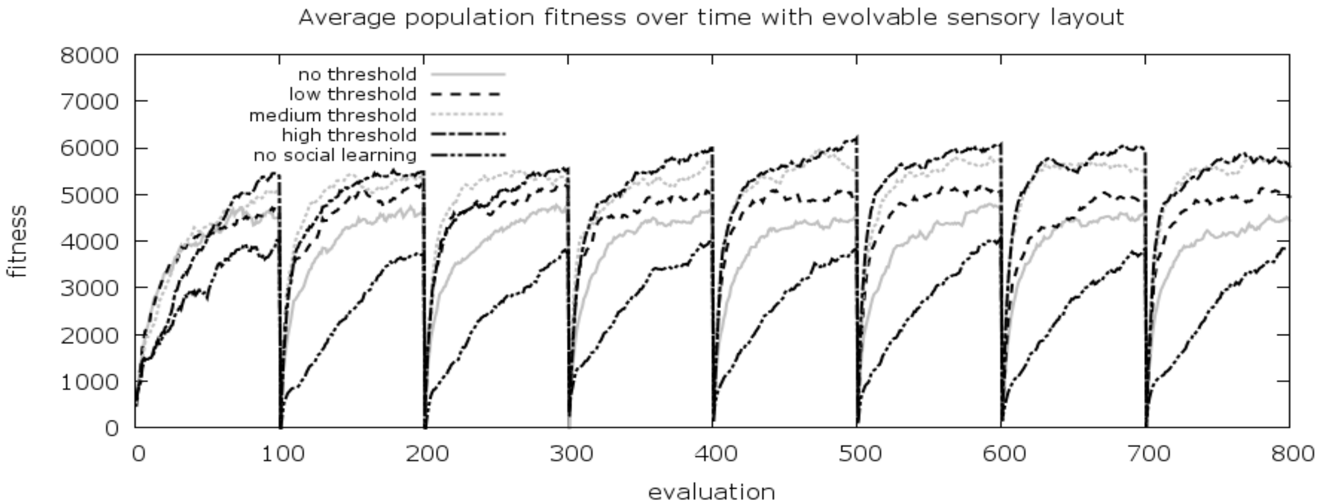

Figure 6: Average population fitness when the sensory layouts of the robots are evolving. Time is measured by the number of evaluations (x-axis), fitness by the formula in Section 3 (y-axis). Averages are taken over 50 repetitions. After 100 evaluations an evolutionary step takes place resulting in a new genotype (sensory layout) for the e-pucks and the controllers are re-initialized.

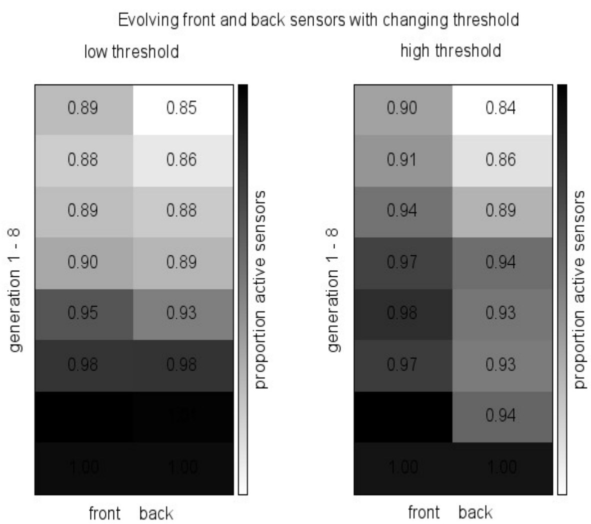

Figure 7: Evolution of the usage of front and rear sensors. The grayscale indicates the portion of active sensors w.r.t. the initial setting over time (black = 100%), the numbers show the exact figures.

Limited by the battery life of the real e-pucks we could only accommodate eight generations in the experiments with evolving sensory layouts. In general, this is not much on an evolutionary timescale, but we could observe a trend of a decreasing number of active sensors over generations.

All in all, our experiments represent a proof of concept, in the meanwhile they provide answers to the specific research questions within our implementation. Ongoing work concerns validation of these results in hardware, facing several challenges. Direct communication between e-pucks is not reliable enough for continuous and frequent broadcasting of memotypes. Extending the hardware with a Linux extension board will hopefully solve this problem. Further research will address tuning of the parameters, using multiple tasks and the use of different hardware, for example Thymio II robots.

7. ACKNOWLEDGMENTS

This work was made possible by the European Union FET Proactive Initiative: Knowing, Doing, Being: Cognition Beyond Problem Solving funding the Deferred Restructuring of Experience in Autonomous Machines (DREAM) project under grant agreement 640891.

8. REFERENCES

[1] K. Balakrishnan and V. Honavar. On sensor evolution in robotics. In *Proceedings of the 1st annual conference on genetic programming*, pages 455–460. MIT Press, 1996.

[2] H.-G. Beyer. Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):239 – 267, 2000.

[3] J. Bongard. Evolutionary robotics. *Communications of the ACM*, 56(8):74–85, 2013.

[4] N. Bredeche, E. Haasdijk, and A. E. Eiben. On-line, on-board evolution of robot controllers. In P. Collet, N. Monmarché, P. Legrand, M. Schoenauer, and E. Lutton, editors, *Proceedings of the 9th international conference on Artificial evolution*, pages 110–121, Berlin, 2009. Springer.

[5] N. Bredeche, J.-M. Montanier, W. Liu, and A. F. Winfield. Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):101–129, 2012.

[6] K. Dautenhahn, D. Polani, and T. Uthmann. Guest editors' introduction: Special issue on sensor evolution. *Artificial Life*, 7(2):95–97, 2001.

[7] S. Doncieux, N. Bredeche, and J.-B. Mouret, editors. *New Horizons in Evolutionary Robotics*, volume 341 of *Studies in Computational Intelligence*. Springer, 2011.

[8] A. E. Eiben. Grand challenges for evolutionary robotics. *Frontiers in Robotics and AI*, 1(4), 2014.

[9] A. E. Eiben. In Vivo Veritas: towards the Evolution of Things. In T. Bartz-Beielstein, J. Branke, B. Filipič, and J. Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII*, volume 8672 of *LNCS*, pages 24–39. Springer, 2014.

[10] S. Elfwing, E. Uchibe, K. Doya, and H. Christensen. Biologically inspired embodied evolution of survival. In *2005 Congress on Evolutionary Computation (CEC'2005)*, volume 3, pages 2210–2216, Edinburgh, UK, 2005. IEEE Press, Piscataway, NJ.

[11] D. Federici. Combining genes and memes to speed up evolution. In *2003 Congress on Evolutionary Computation (CEC'2003)*, volume 2, pages 747–754, Edinburgh, UK, 2003. IEEE Press, Piscataway, NJ.

[12] D. Floreano, P. Husbands, and S. Nolfi. Evolutionary robotics. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, volume Part G.61, pages 1423–1451. Springer, 2008.

[13] E. Haasdijk, A. E. Eiben, and A. F. Winfield. *Individual, Social and Evolutionary Adaptation in Collective Systems*, chapter 12, pages 413–471. Pan Stanford, Singapore, 2013.

[14] F. Heylighen and K. Chielens. Cultural evolution and memetics. *Encyclopedia of complexity and System Science*, pages 1–27, 2008.

[15] R.-J. Huijsman, E. Haasdijk, and A. E. Eiben. An on-line on-board distributed algorithm for evolutionary robotics. In J.-K. Hao, P. Legrand, P. Collet, N. Monmarché, E. Lutton, and M. Schoenauer, editors, *Artificial Evolution, 10th International Conference Evolution Artificielle*, number 7401 in LNCS, pages 73–84. Springer, 2011.

[16] A. L. Nelson, G. J. Barlow, and L. Doitsidis. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345 – 370, 2009.

[17] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA, 2000.

[18] M. E. Payton, M. H. Greenstone, and N. Schenker. Overlapping confidence intervals or standard error intervals: what do they mean in terms of statistical significance? *Journal of Insect Science*, 3(1):34, 2003.

[19] P. Vargas, E. D. Paolo, I. Harvey, and P. Husbands, editors. *The Horizons of Evolutionary Robotics*. MIT Press, 2014.

[20] J. H. Walker, S. M. Garrett, and M. S. Wilson. The balance between initial training and lifelong adaptation in evolving robot controllers. *IEEE Trans. Syst., Man, Cybern. B*, 36(2):423–432, 2006.

[21] R. A. Watson, S. G. Ficici, and J. B. Pollack. Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18, Apr. 2002.

[22] A. Winfield and J. Timmis. Evolvable robot hardware. In M. Trefzer and A. Tyrrell, editors, *Evolvable Hardware*, Natural Computing Series, pages 331–348. Springer, 2015.

[23] A. F. T. Winfield and M. D. Erbas. On embodied memetic evolution and the emergence of behavioural traditions in robots. *Memetic Computing*, 3(4):261–270, July 2011.

[24] S. Wischmann, K. Stamm, and F. Wörgötter. Embodied evolution and learning: The neglected timing of maturation. In F. Almeida e Costa, editor, *Advances in Artificial Life: 9th European Conference on Artificial Life*, volume 4648 of *Lecture Notes in Artificial Intelligence*, pages 284–293. Springer-Verlag, Lisbon, Portugal, September 10–14 2007.