Can Social Learning Increase Learning Speed, Performance or Both?

Jacqueline Heinerman[1], Jörg Stork[1,2], Margarita Alejandra Rebolledo Coy[1,2], Julien Hubert[1],
Thomas Bartz-Beielstein[2], A.E. Eiben[1], Evert Haasdijk[1]

[1]Vrije Universiteit Amsterdam, The Netherlands
[2]TH Köln, Germany
j.v.heinerman@vu.nl

Abstract

Social learning enables multiple robots to share learned experiences while completing a task. The literature offers contradicting examples of its benefits; robots trained with social learning reach a higher performance, an increased learning speed, or both, compared to their individual learning counterparts. No general explanation has been advanced for the difference in observations, which make the results highly dependent on the particular system and parameter setting. In this research, we show that even within one system, the observed advantages of social learning can vary between parameter settings.

Using Evolutionary Robotics, we train robots individually in a foraging task. We compare the performance of 50 parameter instances of the evolutionary algorithm obtained by a definitive screening design. We apply social learning in groups of two and four robots to the parameter settings that lead to the best and median performance. Our results show that the observed advantages of social learning differ highly between parameter settings but in general, median quality parameter settings experience more benefit from social learning.

These results serve as a reminder that tuning of the parameters should not be left as an afterthought because they can drastically impact the conclusions on the advantages of social learning. Additionally, these results suggest that social learning reduces the sensitivity of the learning process to the choice of parameters.

Introduction

Consider a collective of autonomous robots in an environment that is not well understood or modelled at design time. It is not possible to develop and validate adequate robot controllers without a thorough understanding of the environment, so the robots need to be able to adapt their behaviour to suit. It is preferable that the robots are capable of learning autonomously, without the need for central oversight: such a centralised scheme implies substantial communication overhead and introduces a single point of failure in the combined system.

In such a setting, the robots can learn individually, e.g., by encapsulating a self-sufficient evolutionary algorithm within each robot, and they can learn collectively by sharing knowledge. We consider such collective sharing to be social learning. Social learning among robots is often considered as an extension of reinforcement learning (García-Martínez et al., 2006; Ahmadabadi and Asadpour, 2002; NG and Emami, 2015), but here, we consider social learning in the context of on-line evolutionary robotics. In terms of the taxonomy defined by Haasdijk et al. (2012), we consider in particular *hybrid* systems where robots can adapt individually, but can also exchange genetic material.

There is ample evidence that set-ups where robots can share information outperform otherwise equivalent set-ups where robots learn in isolation. When robots share information, they achieve better performance and/or the learning curve is steeper (Usui and Arita, 2003; Curran and O'Riordan, 2007; Perez et al., 2008; Pugh and Martinoli, 2009; García-Sánchez et al., 2012; Heinerman et al., 2015a,b; Miikkulainen et al., 2012; Tansey et al., 2012). In particular, evidence by Huijsman et al. (2011) and Silva et al. (2015) shows that social learning can linearly decrease learning time.

A substantial portion of research in this field has focused on various modalities of exchanging knowledge. Usui and Arita (2003) showed that the speed of adaptation (not the final performance level) improves in hybrid set-ups compared to purely distributed ones, but that this improvement depends on the size of the encapsulated population. Other research considered which robots can exchange knowledge (Pugh and Martinoli, 2009) and what knowledge to exchange (García-Sánchez et al., 2012).

An aspect that has received less attention in this field is the selection of parameter settings to achieve optimal performance. This can be accomplished on-line, adapting parameter settings during run-time (*parameter control*), or off-line, optimising the parameter settings before the actual runs (*parameter tuning*) (Eiben et al., 1999).

Methodologies for tuning in evolutionary computing were discussed by Eiben and Jelasity (2002) and Beielstein (2003), resulting in the development of software tools such as REVAC (Smit and Eiben, 2010) and SPOT (Bartz-Beielstein et al., 2005). McGeoch (2012) presented a guide to tuning and performing computer experiments.

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

Huijsman et al. (2011) did use extensive parameter tuning, but closer analysis revealed that evolution had not yet converged in the experiments presented there. In fact, the encapsulated model actually achieves better performance than the hybrid model after a substantial increase in the number of evaluations. Since this is a rare example of a comparison between separately tuned versions of evolutionary algorithms, the results lead us to hypothesise that the increase in performance and/or learning speed when adding social learning may depend on the quality of the parameter settings used in the baseline experiments. In other words: poorly tuned parameter settings benefit more from social learning.

To test this hypothesis, we generated 50 parameter settings using Design of Experiments (DoE) and tested them in an individual learning configuration with a single robot (i.e., with social learning disabled). We then selected the 10 settings with best performance and 10 with median performance. We compared the performance and learning speed of these two groups of settings in a social learning configuration with either 2 or 4 robots to determine the impact of tuning and social learning. The robots use embedded instances of the NEAT evolutionary algorithm for on-line learning. They can also exchange individuals to enable social learning.

The results of this study are in line with earlier research: social learning can increase learning speed and/or performance. However, we argue that the observed benefits due to social learning depend on the quality of the parameter settings. While the benefits of social learning differ per parameter setting, in general, the performance and speed increase achieved by social learning is significantly higher for the median parameter settings than for the best parameter settings. Moreover, we showed that the median parameter settings cannot be distinguished from the best parameter settings in terms of performance and learning speed when social learning is added. This suggests that social learning reduces the sensitivity of the learning process to the choice of parameters.

System Description

Task, Robot, and Environment

The experiment requires the robots to learn a foraging task. The environment is a square arena as shown in figure 1. Five pucks are randomly placed in the arena at the start of a run. The robots must collect the pucks and bring them to the nest located in the centre of the arena. Once a puck is brought to the nest, it is immediately moved to a random location in the environment. The performance of each robot, i.e. its fitness, is equal to the number of pucks collected during a trial lasting 1000 time steps.

The experiments are conducted in simulation using JBotEvolver (Duarte et al., 2014)[1]. JBotEvolver is a Java-based open-source, cross-platform framework for research and education in Evolutionary Robotics featuring a 2D differential-drive kinematics engine. The robots in our experiments simulate an e-puck robot. This robot is a small (7 cm) differential wheeled mobile robot equipped with 8 infrared proximity sensors and a camera. Additionally, the robots are equipped with the following task specific sensors:

Puck carrying sensor Indicates if the robot is carrying a puck;

Puck sensor Indicates the distance to the closest puck within the 45 degree perception cone of the sensor;

Nest sensor Indicates the distance to the nest if within the 45 degree perception cone of the sensor.

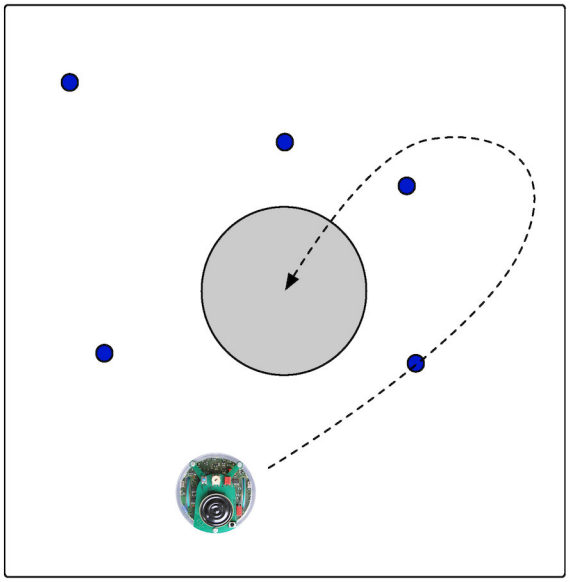


Figure 1: The environment with one robot searching for the blue pucks. The target location is indicated by the grey circle. The dashed line shows an example trajectory of a robot that picks up a puck to release at the target location.

Controller and Individual Learning Mechanism

The robot's controller is an artificial neural network. The neural network has 11 input and two output nodes. The input nodes consist of 8 proximity sensors, a nest sensor, a puck sensor, and a puck carrying sensor; the output nodes provide the right and left motor speed.

Individual learning is implemented by NEAT (Stanley and Miikkulainen, 2002). NEAT is an evolutionary algorithm that evolves both the topology and the connectivity of artificial neural networks. The initial population is composed of

[1]The code is available on `https://github.com/ci-group/ECAL_SocialLearning`.

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

randomly generated feed forward neural networks without hidden layers. Over time, complexification can add nodes and connections to the neural network, including the possibility of forming recurrent connections. All nodes have sigmoid activation functions.

Each robot possesses its own instance of NEAT, with its own population of controllers. The fitness of each controller is evaluated directly on the robot for 1000 time steps. The learning is conducted on-line, i.e. the robot is not relocated between the evaluations and each controller is tested starting from the location reached by the previous one.

A consequence of on-line learning is that a controller can suffer from a bad initial position caused by a previous evaluation. Having to recover from a bad starting position can impact the fitness of the new controller in a negative way (Bredeche et al., 2009). In our set-up, the most common example of a bad starting position is being placed against a wall. To mitigate the negative effect of a bad starting position, we reposition the robot to a random location at the beginning of an evaluation.

Social Learning Mechanism

Social learning is implemented as follows: first, the robots sequentially evaluate the controllers in the current generation. Then, the robots exchange information. Each robot randomly selects another robot from which it receives its current best controller, i.e. the controller with the highest fitness. The robot compares the received controller's fitness to that of its own worst controller. The new controller replaces the worst controller if it is better. The NEAT algorithm uses the updated list of controllers and fitness values to create the next generation. We consider two variants for the social learning setup: among two robots and among a group of four robots.

As noted earlier, NEAT can modify the topology of the neural networks during evolution. Every structural modification in the network is identified by a unique innovation number to enable alignment of genomes for recombination purposes. When implementing NEAT with the possibility to exchange individuals as described for social learning, care must be taken to avoid conflicting innovation numbers. Silva et al. (2012) introduced a simple but effective to ensure this by using timestamps to ensure unique innovation numbers for parallel NEAT instances.

Parameter Tuning

Control parameters handle the behaviour of an algorithm and can be adapted to a problem at hand. This has a great impact on the performance, particularly for complex evolutionary algorithms with a large number of parameters such as NEAT. In evolutionary algorithms, one approach for adjusting the parameters is parameter control, which adapts the essential parameters such as the mutation step-size dynamically during the optimisation run by self-adaptation or co-evolution (Eiben et al., 1999). For this study, off-line tuning is more relevant because it yields better insight and exposes significant parameters and beneficial settings before the optimisation run. Like parameter control, parameter tuning requires understanding of the evolutionary algorithm as a basis for assumptions regarding applicable ranges of each parameter. The number of different test settings and thus experiments can become large and computationally expensive for an extensive analysis, particularly for a great number of parameters. Design of Experiments (DoE) allows us to plan cost-efficient designs that define a sequence of experiments. DoE is an approach to create designs according to certain optimality criteria, so that appropriate data is collected with maximised resource efficiency. We then statistically analyse the data to model the parameter effects and draw valid and objective conclusions (Montgomery, 2009). We use definitive screening designs (Jones and Nachtsheim, 2011) to measure the effects on the robot's learning performance and identify significant parameters.

Definitive screening designs identify the effect the parameters have on the response either by themselves or in combination with others, so called interactions. An important feature of the design is that it allows the fitting of second order models and thus the estimation of quadratic effects, whereby it uses fewer evaluations than classical three level full factorial designs, which demand 3^k experiments with k being the number of parameters. The insights obtained from these initial experiments allow us to reduce the final set of selected parameters, reducing the total number of experiments when generating new designs.

Experimental Setup

Definitive Screening Design

NEAT has a large set of parameters that influence the behaviour of the algorithm. On basis of our experience, we first selected a set of 21 possible influential parameters from the list of all parameters. Table 1 lists this set, explanations, and the actual level values for the screening design plan. The definitive screening DoE was created with the help of JMP software[2]. For the case at hand, with 19 continuous and 2 boolean parameters, the experimental design consists of 50 different parameter combinations. The design uses D-optimality criteria, which focuses on estimating the influence of the parameters on the fitness (Pukelsheim, 1993). For each of the 50 settings, 20 replicate runs with different random seeds are performed. The mean fitness after each generation is stored. We use a fixed budget of evaluations per run. As suggested by Jones and Nachtsheim (2011), we use a forward stepwise regression model with quadratic effects and two-way interactions for the analysis of the DoE. The stepwise regression starts with a small linear model and progressively adds more model terms based on an informa-

[2]SAS Institute Inc, JMP, Version 11.1.0

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

tion criterion. We choose the Bayesian information criterion (BIC) for the stepwise regression to balance the goodness-of-fit and the model complexity. Following Saltelli (2002), the importance index of the parameters is estimated by using Monte Carlo re-sampling of the regression model. Parameters which importance is close to zero (lower than 0.01) can be deemed nonsignificant to the system performance. To normalise the effect of the varying population size (parameter $size$), we adjusted the number of final generations for each level of $size$ to match a target value of 20k fitness evaluations. The number of evaluations are $generations \times size$, thus 20k evaluations correspond to generation 200 for $size$ 100, 333 for $size$ 60 and 1000 for $size$ 20.

Social Learning

The parameter settings provided by the DoE are ranked based on their performance in the final evaluation (evaluation 20000). Rank number 1 is the parameter setting resulting in the highest performance and rank number 50 is the parameter setting resulting in the lowest performance.

For the social learning experiments, we use the best and median parameter settings. The best parameter settings are defined as the 10 settings that lead to the highest performance (rank 1-10). The median parameter settings are defined as the settings resulting in a median performance (rank 21-30).

When social learning is applied, the robots have the same individual learning mechanisms as the one robot setup except for the population size. The population size for the social learning setup is the population size from the 1 robot setup divided by the number of robots to ensure the same number of evaluations per generation This, if the original setup specifies a population size of 100, the social learning experiments use a population size of 50 and 25 for the 2 robot and 4 robot setup, respectively.

The robots operate in their own arena but they communicate across arenas. Consequently, the performance of the robot is only due to its own actions and not influenced by other robots in the same arena. Removing this inter-robot collision allows for a better comparison between the individual and the social learning experiments. For each experiment, 20 replicate runs are performed with different random seeds.

The performance of the system is measured as the number of collected pucks by one robot. For the learning speed, we use two measures:

Speed ratio The speed ratio is measured by the time needed for multiple robots to reach 80% of the one robot performance, divided by the time that the one robot needed to reach 80% of this performance. A ratio lower than 1 means that social learning results in a steeper learning curve compared to individual learning. For example, suppose that the final performance of 1 robot is 2.055. The time, in number of generations, to reach 80% of this performance (1.644) is 116. The time to reach the performance of 1.644 with 2 and 4 robots is 61 and 57 generations respectively. The speed ratio for 2 and 4 robots therefore is 0.526 (61/116) and 0.491 (57/117).

Learning speed The learning speed is defined by one divided by the learning ratio. The learning ratio is calculated by dividing the time, in number of generations, needed to reach 80% of the final performance, divided by the total number of generations. For example, the final performance of a setup is 3.71. 80% of this performance is 2.97. This performance is reached at generation 64 out of the total 200 generations. The learning ratio is 0.32 (64/200) and the learning speed 3.13 (1/0.64). The learning speed can be interpreted as a measure of convergence. A higher learning speed means that the 80% performance was reached earlier in time. Therefore, the remaining 20% performance increase took a longer time, indicating a flattening learning curve.

Results

Parameter Screening

The performance differences between the parameter settings are significant: the lowest ranked setting has a mean fitness of 0.245, while the highest ranked setting reached 3.71 after 20k evaluations. The fitted stepwise regression model has an adjusted R^2 value of 0.925 with an RMSE of 0.25, whereby the mean value of the response is 1.69. This indicates a good model fit, in other words, the model closely characterizes the data. The residuals give no indications of a breach of the normality assumption. Moreover, only the parameter $threshold_{Change}$ was found to have a quadratic effect on the response. The importance of the parameters after the stepwise regression was evaluated and those with importances too close to zero were deemed insignificant to the model. Table 2 shows the relevant model parameters selected after the stepwise regression ordered by their effect importance and the best performing parameter setting. The main effect corresponds to the effect that a parameter has by itself on the response. The total effect column refers to the main effect of a factor plus interaction and higher-order effects. From the initial 21 parameters only 9 seem to have a significant impact on the model response. The NEAT parameters not included in table 2 will be disregarded when further analysing the performance and speed of the system in future research. From the results we conclude that $p_{WeightReplaced}$ has a dominant main effect on the model response. The parameters $p_{Mutation}$, $size$ and $threshold_{Change}$ also have a considerable main influence. Parameter interactions have only minor overall effects, as can be seen from the differences in the main and total effect values.

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

Table 1: Selected set of NEAT control parameters with description and values for the definitive screening design. Low, medium (mid) and high levels are used for numeric parameters

	Low	Mid	High
Mutation and crossover parameters			
$\mathbf{p_{Xover}}$ chance to apply crossover	0.05	0.20	0.35
$\mathbf{p_{Mutation}}$ chance to apply mutation on each node/link	0.1	0.25	0.4
$\mathbf{p_{WeightReplaced}}$ chance to replace weight	0.0	0.25	0.5
$\mathbf{max_{Perturb}}$ maximum allowed change on weight	0.25	0.5	0.75
$\mathbf{p_{AddLink}}$ chance to add a link	0.01	0.05	0.1
$\mathbf{p_{AddNode}}$ chance to add a node	0.01	0.03	0.05
Species parameters			
$\mathbf{species_{Count}}$ Maximum number of species.	3	6	9
$\mathbf{max_{SpeciesAge}}$ maximum age of species	6	18	30
$\mathbf{coeff_{Excess}}$ used for species compatibility score	0.5	1.0	1.5
$\mathbf{coeff_{Disjoint}}$ used for species compatibility score	0.5	1.0	1.5
$\mathbf{coeff_{Weight}}$ used for species compatibility score	0.1	0.4	0.7
$\mathbf{threshold}$ used for species compatibility score	0.3	0.5	0.7
$\mathbf{threshold_{Change}}$ used to change threshold value	0.01	0.1	0.2
$\mathbf{species_{AgeThreshold}}$ percentage of age to count as old	0.7	0.75	0.8
$\mathbf{species_{YouthThreshold}}$ percentage of age to count as young	0.2	0.25	0.3
$\mathbf{age_{Penalty}}$ fitness multiplier for old individual	0.5	0.7	0.9
$\mathbf{age_{Boost}}$ fitness multiplier for young individual	1.1	1.25	1.4
Other parameters			
$\mathbf{size}$ population size of one robot	20	60	100
$\mathbf{survival_{Threshold}}$ top % individuals that can be parents	0.1	0.45	0.8
$\mathbf{copy_{Best}}$ clone best individual previous generation	TRUE	FALSE	
$\mathbf{copy_{BestEver}}$ Clone best individual so far.	TRUE	FALSE	

Social Learning

Baseline Experiments The 50 parameter settings from the DoE are referred to as the baseline experiments. For the baseline experiments, there is only 1 robot, and it is learning individually. Figure 2 shows the mean performance of the baseline experiments at the final evaluation (evaluation 20000). Performance, i.e., number of collected pucks, is plotted against the rank of the mean performance of the 50 parameter settings. As an example: the best setting (rank 1) has a mean of 3.71 collected pucks per controller evaluation. The data in figure 2 confirms that parameter settings significantly influence the performance of the controllers (Pearson's r(50)= -0.99, $p < 2.2e\text{-}16$).

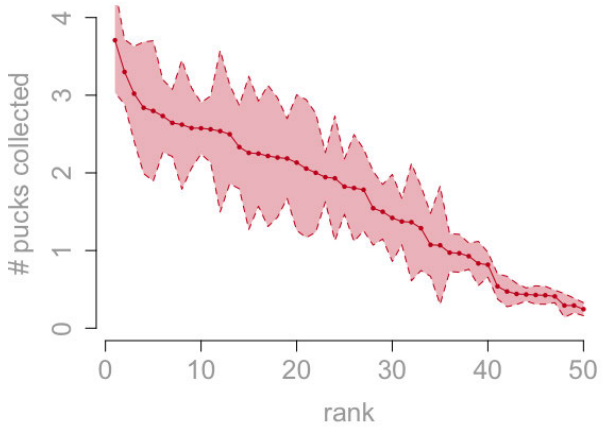


Figure 2: Mean performance with 95% confidence interval of the baseline experiments for all DoE parameter settings. The y-axis shows the performance, measured as the number of collected pucks. The x-axis shows the rank of the parameter setting. The results are compiled over 20 replicate runs.

Social Learning: Speed Figure 3 shows the speed ratio for the best and median parameter settings. Ranks 1-10 refer to the best settings and ranks 21-30 to the median settings. Every setting is tested for a setup with 2 (red) and 4 (blue) robots in 20 replicate runs.

From figure 3 we can conclude that most parameter settings benefit from more robots, indicated by the speed ratio values lower than 1. A Mann-Whitney test indicated that the speed ratio was significantly higher for the best settings (Mdn=0.74) compared to the median settings (Mdn=0.51), p=0.005. There is no correlation between the specific rank number and the speed ratio (for 2 robots Pearson's r(20) = -0.39, $p = 0.085$; for 4 robots Pearson's r(20) = -0.39, $p = 0.089$).

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

Parameter	Effects		Best Setting	
	Main (sd)	Total (sd)		
$\mathbf{p}_{\mathbf{WeightReplaced}}$	0.510 (0.009)	0.545 (0.01)	`0.0`	Low
$\mathbf{p}_{\mathbf{Mutation}}$	0.103 (0.008)	0.130 (0.005)	`0.4`	High
size	0.092 (0.008)	0.117 (0.005)	`100`	High
$\mathbf{threshold}_{\mathbf{Change}}$	0.098 (0.007)	0.111 (0.005)	`0.2`	High
$\mathbf{age}_{\mathbf{Penalty}}$	0.025 (0.005)	0.041 (0.003)	`0.9`	High
$\mathbf{species}_{\mathbf{Count}}$	0.021 (0.004)	0.029 (0.005)	`9`	High
$\mathbf{survival}_{\mathbf{Threshold}}$	0.014 (0.003)	0.021 (0.002)	`0.1`	Low
$\mathbf{max}_{\mathbf{Perturb}}$	0.010 (0.003)	0.016 (0.002)	`0.25`	Low
$\mathbf{copy}_{\mathbf{Best}}$	0.010 (0.002)	0.015 (0.002)	`FALSE`	

Table 2: The importance index is estimated by using Monte Carlo re-sampling of the regression model. The effects indicate the influence that each parameter has on the response, either by itself (main) or in combination with other parameters (total). The rank 1 setting, which is best performing setting of the screening design, is shown with values and their corresponding level (low, middle or high). Parameters with total effects > 0.01 are displayed; the not shown remaining parameters have no significant impact on the response.

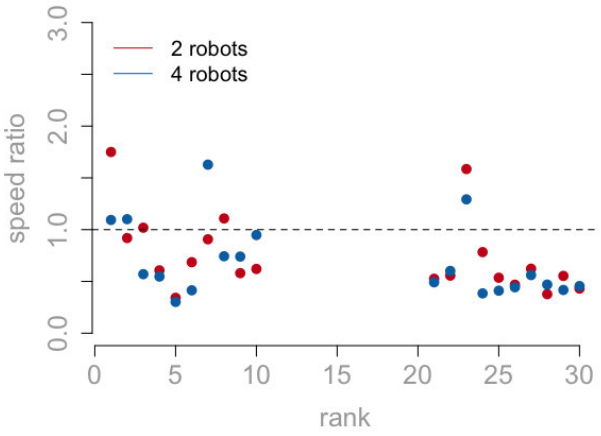


Figure 3: Speed ratio of 2 (blue) and 4 (red) robots compared to the baseline experiments. The y-axis shows the speed ratio, measured by the time needed for 2 or 4 robots to reach 80% of the baseline performance, divided by the time that the one robot needed to reach 80% of the baseline performance. The x-axis shows the rank of the parameter setting (rank 1 is the rank resulting in the highest performance). The results are compiled over 20 replicate runs.

Social Learning: Performance Figure 4 shows the performance and learning speed for 1, 2 and 4 robots. The red dots represent the best (rank 1-10) parameter settings of the baseline experiments and the blue dots the median (rank 21-30) parameter settings of the baseline experiments.

Figure 4 (a) shows the 1 robot performance and learning speed of the best (red) and median (blue) parameter settings of the baseline experiment. As can be seen, the median parameter settings have a clear division between parameter settings that are still improving (low learning speed) and settings that are converged (high learning speed). When the baseline parameter settings are applied to the multiple robot setup, as explained in the experimental setup, we can see a trend for the median settings towards a higher performance and a decreased learning speed in figure 4 (b) and (c). The performance increase due to social learning (calculated by dividing the performance of multiple robots by the baseline performance) is positively correlated with the rank (for 2 robots Pearson's r(20) = 0.61, p = 0.004 ; for 4 robots Pearson's r(20) = 0.68, $p < 0.001$).

In figure 4 (c), it is not possible to separate the best from the median parameter settings. Therefore, we can conclude that the use of more robots, in this setup, reduces the sensitivity of the learning process to the choice of parameters.

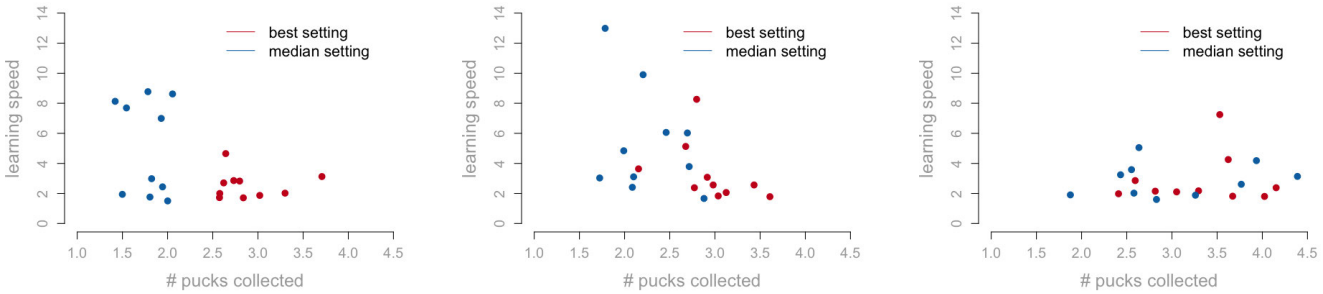
Discussion

In this paper, we investigated the advantages of social learning in terms of performance and learning speed in a foraging scenario. More specifically, we compared the advantages for two kinds of parameter settings: parameter settings that lead to high performance and parameter settings that lead to median performance.

Our results confirm that the choice of parameter settings significantly impacts the performance of 1 robot learning. More interestingly, our results indicate that the choice of parameter settings also influences the advantages of social learning.

Our initial analysis showed that 9 of NEAT's parameters have a significant impact on the learning performance. In this case, $p_{WeightReplaced}$ has a big effect on the response and performance is best if it is turned off. Moreover, a large

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).



(a) 1 robot (b) 2 robots (c) 4 robots

Figure 4: Learning speed and performance of the best (red) and median (blue) parameter settings for 1 (a), 2 (b) and 4 (c) robots. The y-axis shows the learning speed, measured as one divided by the learning ratio. The learning ratio is measured by the time, in number of generations, to reach 80% of the final performance, divided by the total number of generation. The x-axis shows performance, i.e. the average number of collected pucks per robot. The results are compiled over 20 replicate runs.

population $size$, a high $p_{Mutation}$ probability and a high $threshold_{Change}$ rate have a significantly beneficial influence on the fitness. This indicates that a large and diverse population of controllers seem to be advantageous for the learning rate. Moreover, as the best settings are mostly situated at the extreme values tested, we assume that extending the initially defined parameter limits may lead to a further performance improvement.

Second, we showed that the speed increase and the performance increase due to social learning depends on the quality of the parameter settings. The correlation between the quality of the parameter settings and the performance improvement was significant: better parameter settings benefit less from social learning. For the learning speed, there is no significant correlation between the specific rank and speed increase. However, when grouping the speed increase of the best and median parameter settings, there is a significant difference between them.

Finally, we showed that the median parameter settings can not be distinguished from the best parameter settings when used with social learning between 4 robots. This suggests that social learning reduces the sensitivity of the learning process to the choice of parameters. A potential explanation could be that social learning increases the diversity and therefore avoids the problem of premature convergence, but further research is needed to confirm this hypothesis.

For the social learning experiments, we used the parameters of the 1 robot setup while these parameters might not be optimal for the multiple robot setup. A multiple robot setup itself can also benefit from parameter tuning. In future research, we will study the the effect of different parameter settings on the performance of the multi robot setup.

In the social learning experiments, the total number of controller exchanges between robots varies depending on the number of generations. The influence of the frequency of controller exchanges on the speed of convergence in social learning is still an open question for future research.

Conclusions

Existing literature in social learning typically compares individual learning with social learning for only one parameter setting. Our study extended this comparison by using different parameter settings. We showed that the quality of the parameter settings influences how much social learning can improve the system performance: the better the parameter settings, the less social learning can contribute in terms of performance and learning speed without further tuning. As a consequence, research in social learning must consider the quality of the used parameter settings, as they can drastically impact the conclusion. Additionally, we showed that, in our system, social learning seems to reduce the sensitivity to the choice of parameters.

Acknowledgements

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.

References

Ahmadabadi, M. N. and Asadpour, M. (2002). Expertness based cooperative q-learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 32(1):66–76.

Bartz-Beielstein, T., Lasarczyk, C., and Preuss, M. (2005). Sequential Parameter Optimization. In McKay, B. et al., edi-

tors, *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, pages 773–780, Piscataway NJ. IEEE Press.

Beielstein, T. (2003). Tuning Evolutionary Algorithms—Overview and Comprehensive Introduction. Technical report, Technische Universität Dortmund.

Bredeche, N., Haasdijk, E., and Eiben, A. (2009). On-line, on-board evolution of robot controllers. In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 110–121, Berlin Heidelberg. Springer.

Curran, D. and O'Riordan, C. (2007). Cultural Learning in a Dynamic Environment: an Analysis of Both Fitness and Diversity in Populations of Neural Network Agents. *Journal of Artificial Societies and Social Simulation*, 10(4):1–3.

Duarte, M., Silva, F., Rodrigues, T., Oliveira, S. M., and Christensen, A. L. (2014). Jbotevolver: A versatile simulation platform for evolutionary robotics. In *Proceedings of the 14th International Conference on the Synthesis & Simulation of Living Systems.*, pages 210–211, Cambridge, MA. MIT Press.

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 Jelasity, M. (2002). A Critical Note on Experimental Research Methodology in EC. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'2002)*, pages 582–587, Piscataway NJ. IEEE.

García-Martínez, R., Borrajo, D., Maceri, P., and Britos, P. (2006). Learning by knowledge sharing in autonomous intelligent systems. In *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*, pages 128–137. Springer, Berlin Heidelberg.

García-Sánchez, P., Eiben, A. E., Haasdijk, E., Weel, B., and Merelo-Guervós, J.-J. (2012). Testing Diversity-Enhancing Migration Policies for Hybrid On-Line Evolution of Robot Controllers. In *European Conference on the Applications of Evolutionary Computation*, pages 52–62, Berlin Heidelberg. Springer.

Haasdijk, E., Smit, S. K., and Eiben, A. E. (2012). Exploratory analysis of an on-line evolutionary algorithm in simulated robots. *Evolutionary Intelligence*, 5(4):213–230.

Heinerman, J., Drupsteen, D., and Eiben, A. (2015a). Three-fold adaptivity in groups of robots: The effect of social learning. In Silva, S., editor, *Proceedings of the 17th annual conference on Genetic and evolutionary computation*, GECCO '15, pages 177–183, New York, NY, USA. ACM.

Heinerman, J., Rango, M., and Eiben, A. (2015b). Evolution, individual learning, and social learning in a swarm of real robots. In *Proceedings of the 2015 IEEE International Conference on Evolvable Systems (ICES)*, pages 1055–1062, New York, NY, USA. IEEE Press.

Huijsman, R.-J., Haasdijk, E., and Eiben, A. E. (2011). An On-line On-board Distributed Algorithm for Evolutionary Robotics. In Hao, J.-K., Legrand, P., Collet, P., Monmarché, N., Lutton, E., and Schoenauer, M., editors, *Artificial Evolution, 10th International Conference Evolution Artificielle*, number 7401 in LNCS, pages 73–84. Springer.

Jones, B. and Nachtsheim, C. J. (2011). A class of three-level designs for definitive screening in the presence of second-order effects. *Journal of Quality Technology*, 43(1).

McGeoch, C. C. (2012). *A Guide to Experimental Algorithmics*. Cambridge University Press, New York, NY, USA, 1st edition.

Miikkulainen, R., Feasley, E., Johnson, L., Karpov, I., Rajagopalan, P., Rawal, A., and Tansey, W. (2012). Multiagent learning through neuroevolution. In *IEEE World Congress on Computational Intelligence*, pages 24–46, Berlin Heidelberg. Springer.

Montgomery, D. C. (2009). *Design and Analysis of Experiments*. John Wiley & Sons, Inc.

NG, L. and Emami, M. R. (2015). Concurrent individual and social learning in robot teams. *Computational Intelligence*, 32:420–438.

Perez, A. L. F., Bittencourt, G., and Roisenberg, M. (2008). Embodied Evolution with a New Genetic Programming Variation Algorithm. In *icas*, volume 0, pages 118–123, Los Alamitos, CA, USA. {IEEE} Press.

Pugh, J. and Martinoli, A. (2009). Distributed scalable multi-robot learning using particle swarm optimization. *Swarm Intelligence*, 3(3):203–222.

Pukelsheim, F. (1993). *Optimal Design of Experiments*. Wiley, New York NY.

Saltelli, A. (2002). Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, 145(2):280–297.

Silva, F., Correia, L., and Christensen, A. L. (2015). A case study on the scalability of online evolution of robotic controllers. In *Portuguese Conference on Artificial Intelligence*, pages 189–200, Switzerland. Springer International Publishing.

Silva, F., Urbano, P., Oliveira, S., and Christensen, A. L. (2012). odneat: An algorithm for distributed online, onboard evolution of robot behaviours. *Artificial Life*, 13:251–258.

Smit, S. K. and Eiben, A. E. (2010). Using Entropy for Parameter Analysis of Evolutionary Algorithms. In Bartz-Beielstein, T., Chiarandini, M., Paquete, L., and Preuss, M., editors, *Experimental Methods for the Analysis of Optimization Algorithms*. Springer.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127.

Tansey, W., Feasley, E., and Miikkulainen, R. (2012). Accelerating evolution via egalitarian social learning. In Soule, T., editor, *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 919–926, New York, NY, USA. ACM.

Usui, Y. and Arita, T. (2003). Situated and Embodied Evolution in Collective Evolutionary Robotics. In *Proceedings of the 8th International Symposium on Artificial Life and Robotics*, pages 212–215.

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).