Project no. 640891

DREAM

Deferred Restructuring of Experience in Autonomous Machines

Horizon 2020 Framework Programme

FETPROACT-2-2014

Knowing, doing, being: cognition beyond problem solving

Start date: 1 January 2015 – Duration: 48 months

D4.4

Publication

Due date: M48 - 12/2018

Actual submission date: M48 - 12/2018

Number of pages: 9

Lead beneficiary: VU/VUmc

Author(s): Jacqueline Heinerman, Bart Bussmann, Rick Groenendijk, Emile van Krieken, Jesper Slik, Alessandro Tezza, Evert Haasdijk and A.E. Eiben

Project Consortium

Beneficiary no.	Beneficiary name	Short name
1 (Coordinator)	UNIVERSITE PIERRE ET MARIE CURIE-PARIS	UPMC
2	UNIVERSIDADE DA CORUNA	UDC
3	QUEEN MARY UNIVERSITY OF LONDON	QML
4	ASSOCIATION POUR LA RECHERCHE ET LE DEVELOPPEMENT DES METHODES ET PROCESSUS INDUSTRIELS	ARMINES
5	STICHTING VU-VUMC	VU/VUmc

Dissemination Level

PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Dissemination Nature

R	Report	X
P	Prototype	
D	Demonstration	
O	Other	

Benefits of Social Learning in Physical Robots

Jacqueline Heinerman, Bart Bussmann, Rick Groenendijk, Emile van Krieken,
Jesper Slik, Alessandro Tezza, Evert Haasdijk and A.E. Eiben
VU University Amsterdam, The Netherlands
j.v.heinerman@vu.nl

***Abstract*—Robot-to-robot learning, a specific case of social learning in robotics, enables the ability to transfer robot controllers directly from one robot to another. Previous studies showed that the exchange of controller information can increase learning speed and performance. However, most of these studies have been performed in simulation, where robots are identical. Therefore, the results do not necessarily transfer to a real environment, where each robot is unique per definition due to the random differences in hardware.**

In this paper, we investigate the effect of exchanging controller information, on top of individual learning, in a group of Thymio II robots for two tasks: obstacle avoidance and foraging. The controllers of the robots are neural networks that evolve using a modified version of the state-of-the-art NEAT algorithm, called *cNEAT*, which allows the conversion of innovations numbers from other robots.

This paper shows that robot-to-robot learning seems to at least parallelise the search, reducing wall clock time. Additionally, controllers are less complex, resulting in a smaller search space.

***Index Terms*—Evolutionary Robotics, Neural Networks, Evolutionary Algorithms, Robot-to-Robot Learning**

I. Introduction

To enable autonomous robots to operate reliably in an environment that is not fully understood or known at design time, there is a need for self-learning robots. In such a setting, the robots can learn individually, e.g., by encapsulating a self-sufficient learning algorithm within the robot. These robots learn about and act in their environment while completing a task.

The robotic controller that we consider for the robot to learn a task is a neural network. This is a direct policy that maps the sensor inputs of the robot to actions. This mapping, consisting of nodes and connections between the nodes, are evolved with evolutionary algorithms.

Evolutionary algorithms are inspired by Darwins' theory of survival of the fittest. In nature, animals survive and procreate when they are more fit. Similarly, a robotic controller is tested by observing the behaviour of the robot and is given a corresponding fitness measure. The higher the fitness, the more chance this controller has to procreate. Over generations, the quality of the controllers will improve and lead to robots that are capable of executing a predefined task properly.

The robotic controllers are evolved online, while the robot is performing the task, as opposed to offline learning, where only the best controller is transferred to hardware. Online learning increases the difficulty of the learning process for two reasons. First, if the robot is stuck in a difficult situation, e.g. a room with one small opening, the first and only priority is to recover from this situation. With offline learning, a robot will be repositioned at the end of the controller evaluation. Second, it is important that the whole population of controllers is performing well because the robot is already performing the task. Therefore, the measurements of the performance in this paper always include all individuals in the population.

The choice of online learning is perpendicular to the choice for a physical or simulated platform. In this paper we choose a physical platform. Online learning on a physical platform has the disadvantage to be slow.

Accelerating the learning process could be reached when a collective of autonomous robots is used that can share knowledge, i.e. *socially learn*, to enhance the individual learning process. Note that social learning in robotics is not the same as the widely used definition of social learning: learning through observation of conspecifics. Regarding robots, we can add a type of social learning, that we call robot-to-robot learning, based on the ability to transfer robot controllers directly from one robot to another. (In common parlance this would be the robotic equivalent of telepathy.)

The benefits of robot-to-robot learning have been shown in multiple simulation studies [1], [2], [3]. In particular, evidence by [4] and [5] suggests that robot-to-robot learning can linearly decrease learning time, e.g. the fitness measure that four robots can reach in two hours can be reached by eight robots in one hour when they learn socially. Evidence by [6] showed that for a range of parameter settings, learning speed usually increases when applying robot-to-robot learning. The learning speed increase is lower for the better parameter setting, which are parameter settings that result in a higher performance for the individual learning robot.

Implementations of a physical robotic collective learning a task are rare. The authors of [7] implemented an obstacle avoidance task and [8], [9] looked at the phototaxis task. These tasks are simple tasks that do not require the robot to have a camera. The authors of [10] did use a camera for the relative complex foraging task. This task demands from the robot to collect pucks to bring to a designated target area. However, the controller evaluation time in this implementation was too short (under 4 seconds) to score a goal, which resulted in the necessity to keep a variety of behaviours in the population. Additionally, the robotic group size did not vary in these studies. Therefore, no systematic study has been performed on the benefits of robot-to-robot learning in a physical robotic group for multiple tasks and different group sizes.

The main objective of this paper is to examine the increase

in learning speed and performance due to robot-to-robot learning, on top of individual learning, for varying robotic group sizes. Additionally, we hypothesized that robot-to-robot learning will lead to more robust solutions, as each robot has small random differences. That is to say, some cameras might be mounted under a marginally different angle for example. Learning correct behaviour in experiments with more than one robot means that optimal performance is generalised behaviour over an entire group. Intuitively, more robust solutions are those that are less complex.

Learning is implemented by evolving neural networks with a state-of-the-art evolutionary algorithm called NeuroEvolution of Augmenting Topologies (NEAT). NEAT was designed as a general method that can be applied to any (robotic) task, and for this study, we chose two tasks: obstacle avoidance and foraging. To observe the increase in learning speed and performance due to robot-to-robot learning, we compare 1 learning robot with a group of 4 and 8 robots.

The robots operate in their own arena. Consequently, the performance of the robot is only due to its own actions and not influenced by other robots in the same arena. However, the robots do communicate across arenas. Removing this inter-robot collision allows for a better comparison between the individuals and the robot-to-robot learning experiments.

The NEAT algorithm is not directly applicable for robots that exchange knowledge. We, therefore, propose an extension of NEAT, called *cNEAT*. This extension allows the conversion of innovation numbers from other robots.

This paper shows promising results when applying robot-to-robot learning. We show that on top of a parallelisation of the search, robots that learn socially are more capable of retaining the best controllers. Additionally, the controllers are less complex resulting in a smaller search space, which is extremely beneficial when using physical robots. It is shown that a relatively complex task, such as the foraging, can be learned within the hour. Therefore, online learning methods can be tested on real robots for more complex tasks than currently possible.

II. Learning mechanisms

A. Individual Learning Mechanism

Individual learning takes place through an encapsulating, self-sufficient learning mechanism. The learning mechanism used in this paper is NEAT [11]. NEAT is a state-of-the-art evolutionary algorithm that evolves both the topology and the connectivity of artificial neural networks. In NEAT, an initial population of neural networks without hidden layer is randomly generated. These networks will be referred to as individuals in the population.

Over generations, nodes and connections can be added to individuals. In order to compare different individuals, the changes are stored as innovation numbers. In the original implementation of NEAT, innovation numbers are only used within one generation. As a result, identical innovations could have different innovation numbers when they occur in different generations. Therefore, networks that are similar could have a larger distance measure and could be placed in different species. Therefore, we propose to keep the innovation numbers over generations.

B. Robot-to-Robot Learning Mechanism

Next to the individual learning mechanism, a robot-to-robot learning mechanism takes place as well. First, for every robot, all the individuals in the population are evaluated. Thereafter, the robots exchange information. Each robot sends their best controller to the others and from all received controllers, every robot chooses one controller based on fitness proportionate selection. The robot adds this controller to the population whereafter new offspring is created.

As noted before, 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 robot-to-robot learning, care must be taken to avoid conflicting innovation numbers. Previous work [12] solved this by using timestamps as innovation numbers. However, using timestamps results in a unique innovation for every mutation. This results in the same problem as described before, where two similar networks result in a larger distance than they actually might have. Thus, we propose an adjusted approach called *conversion NEAT (cNEAT)*.

C. cNEAT

When robot 1, **R1**, receives a network from robot 2, **R2**, **R1** needs to match the node IDs of the received network from **R2** with that of its own. This is because **R2** might have assigned the same innovation ID number to a different innovation as opposed to **R1**.

To solve this problem, **R1** iterates over the nodes and converts the node IDs accordingly. This ensures that:

1) the nodes of the received network from **R2** that match a node innovation of **R1** have the same ID;
2) the nodes that do not match with any innovation of **R1** get a new ID that is not assigned to any node of **R1**. This new innovation ID is thereafter added to the list of node innovations of **R1**.

To better understand the conversion, we provide an example shown in Figure 1, on the left. The node IDs of the receiving robot **R1** are on top and the nodes of the received network from **R2** are on the bottom. The received network has two conflicting IDs. Node ID 8 and 9 are placed between nodes 1 and 4. However, the node innovations list of **R1** claims that the node in that position should have ID 6 and 7.

Node 9 is in a more difficult situation than node 8: the ancestors node (ancestorFr and ancestorTo) of node 9 do not match with the ones specified in the node innovation list, due to the wrong assignment of node 8, that should be node 6. Thus, we first need to convert node 8 into node 6 (shown in Figure 1, on the right), and only then we will be able to match and convert node 9 into node 7.

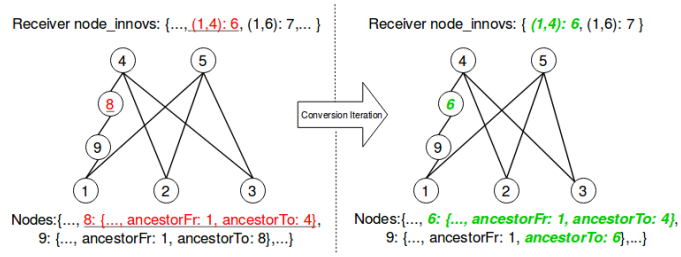

Figure 1: A first iteration of the conversion algorithm with two conflicting node IDs.

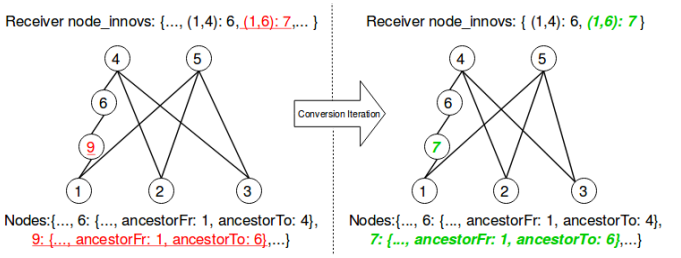

Figure 2: The second and last iteration of the conversion algorithm. The received network has one conflicting ID left (9 should be 7).

It is important to note that the ancestors' information of a node (ancestorFr and ancestorTo) could help to understand the order of creation of the nodes. In fact, node 8 and 9 are both placed between node 1 and 4. However, node 8 was first created (its ancestors are 1 and 4), and only during a next mutation node 9 was created (its ancestors are 1 and 8). For that reason, the conversion of node 9 depends on the conversion of node 8.

Hence, a conversion could trigger other conversions, because some wrong IDs could avoid the matching between the ancestors of the node and the node innovations list. As a result, after the first conversion shown in Figure 1, we can apply a second conversion, shown in Figure 2.

The conversion algorithm needs to iterate over the node ID numbers several times until all the conversion is performed. Within every iteration, the algorithm matches the information about the position of a node (ancestors) and its ID with the node innovation of the receiving robot (in our example **R1**): if the ID is wrong, the information of the node and every reference to it (e.g., ancestor information in other nodes with references to that node) are converted to the ID used in **R1**.

Algorithm 1 summarises the individual and robot-to-robot learning mechanism in pseudocode. The NEAT algorithm is a framework where the specific implementation for the variation and selection operators are not set. In the experimental setup, the list of used parameters is provided to clarify the chosen mechanisms that we used for our specific implementation.

III. Tasks

The learning mechanism is deployed on Thymio II robots to learn two tasks: obstacle avoidance and foraging. The foraging

Algorithm 1 Pseudocode of the algorithm that runs on every robot

initialise population of first generation (P_1) with individuals $i_1, ..., i_n$
while current generation $\leq$ final generation
  for every i in P
    evaluate i
    store $fitness$ of i
  sort the individuals based on fitness (i_1 is best)
  if robot-to-robot learning
    send i_1 to all other agents
    receive best individual from all other agents
    pick one individual r_1 using fitness proportionate selection
    apply the conversion method to r_1
    add r_1 to the population
  create offspring by:
    adjusting specie fitness based on age
    pick parent pool per specie
    calculate number of specie offspring with roulette wheel selection
    clone specie best
    pick parents based on tournament selection
    apply mutation or crossover and mutation
    add offspring to specie

tasks require the robot to collect pucks to bring to a target area. We extend the standard Thymio set-up with a more powerful logic board, a camera (only used for the foraging task), wireless communication, and a high capacity battery. We use a Raspberry Pi 3 that connects to the Thymio's sensors and actuators and processes the data from the Raspberry Pi Camera. The WiFi is integrated with the Raspberry Pi and enables inter-robot communication.

A. Obstacle Avoidance

The obstacle avoidance task requires the robot to drive as fast as possible through an environment without hitting the walls. The network inputs are the 7 proximity sensors around the robot and the outputs are the motor speeds for the left and right wheel. Including the bias node, this results in an initial network of 16 weights. The Thymio II robot in an empty environment for the obstacle avoidance task is shown in Figure 3.

There is a common fitness function to evaluate the robots'

Figure 3: The environment with one robot for the obstacle avoidance task. This setup is duplicated for the number of robots used in the experiment.

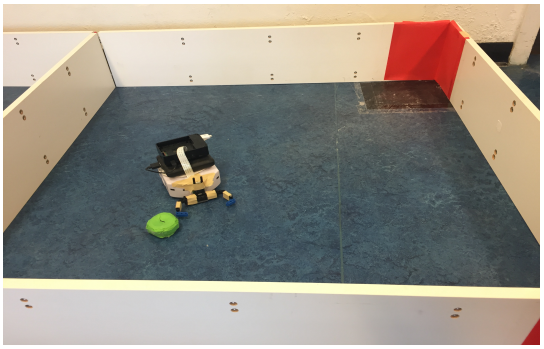

Figure 4: The environment with one robot searching for the green puck to bring to the red target location. This setup is duplicated for the number of robots used in the experiment.

performance for the obstacle avoidance task. Given an evaluation period of T time steps, this is measured as follows:

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

where:

- s_{trans} is the translational speed, calculated as the sum of the speeds assigned to the left and right motor and normalised between 0 and 1;
- s_{rot} is the rotational speed, calculated as the absolute difference between the speed values assigned to the two motors and normalised between 0 and 1;
- v_{sens} is the value of the proximity sensor closest to an obstacle normalised between 0 and 1.

B. Foraging

A foraging task requires the robot to collect pucks and bring them to the nest located in a corner of the arena. The extended Thymio II and the environment is shown in Figure 4.

We use the camera image to define three task-specific sensors: puck in sight, puck in gripper and goal in sight. The goal is visible even if the robot does not have a puck yet. This makes the task more complex. Additionally, we use the sum of proximity sensors in the front and two proximity sensors in the back to determine if the robot is colliding with the wall. As a result, the neural network controller has 6 inputs, including bias. The number of output nodes is again 2, where each node corresponds to a wheel of the robot, indicating the speed. This results in an initial network size of 14 weights.

The fitness of a robot is defined as:

$$f = c_1 \times n_{walls} + c_2 \cdot n_{puck} + c_3 \cdot n_{goal} \quad (2)$$

where:

- n_{walls} is the number of time steps the front and back proximity sensors are not activated, meaning no wall is being hit;
- n_{puck} is the number of pucks collected during an evaluation;
- n_{goal} is the number of goals scored during an evaluation;

The hyperparameters c_1, c_2 and c_3 are empirically put at 1, 1.000 and 10.000 respectively.

With online learning, the next individual inherits the state of the current individual automatically. When the controller evaluation ends with a puck in the gripper, the next individual starts with this puck without any effort. For this reason, a puck only count as collected if the robot did not have a puck within its gripper before. This is done by subtracting the fitness of the sum of the three time steps before to the fitness at the current time step. This means that when a goal is actually scored, the robot has a fitness of around 7000. If the fitness of the time step results in a negative value, the fitness value is set to 0.

IV. Experimental Setup

We distinguish two different sets of experiments: the obstacle avoidance task and the foraging task. For each setup, we compare individual learning only and individual and robot-to-robot learning together. The learning of the robot is conducted online, i.e. the robot is not relocated between the evaluations and each controller is tested starting from the location reached by the previous one.

For the foraging task, there is a necessity to calibrate the cameras of the robots automatically before the start of every run, because the orientation of the cameras across robots can be different. This orientation means that the size of the puck in each of the frames that the robot shoots can vary. After the calibration, a hand-coded controller is used to test the calibration process; the hand-coded controller should exhibit the optimal behaviour: turning until the puck is in sight, drive straight until the puck is in the gripper, turn until the goal is in sight, and drive straight to the goal. Because the quality of the camera is not what we want it to be, we added velcro on the gripper and the puck. As a result, once the robot has the puck, the corresponding sensor input of having the puck will be set to true until a goal is scored (even if the robot does lose the puck). We choose this implementation because the camera is not always accurate enough to see the colours appropriately.

Human intervention is necessary when the robot is in the corner facing the wall and tries to turn right against the wall and the motor of the robot is not powerful enough to push itself back. When the robot collects a puck for the foraging task, we relocate the puck to a random position in the arena and place the robot just behind the black square.

Robot-to-robot learning experiments are performed with a group of 4 and 8 robots. A populations size of 24 is used for the individual learning experiments resulting in a population size of 6 and 3 for the 4 and 8 robot setup respectively. The number of generations is 19, restricted by battery capacity for the 1 robot experiment, and one evaluation is 20 seconds for the obstacle avoidance task and 60 seconds for the foraging task. This results in a time required per experiment of the obstacle avoidance task of around 160 minutes, 40 minutes and 20 minutes for 1, 4 and 8 robots. And for the foraging task of 8 hours, 2 hours and 1 hour for 1, 4 and 8 robots. For

Table I: System parameters, descriptions and used values.

Mutation and crossover parameters	
$\mathbf{p_{Xover}}$ chance to apply crossover	0.75
$\mathbf{p_{Mutation}}$ chance to apply only mutation	0.25
$\mathbf{p_{weightMutation}}$ chance to apply mutation on weight	0.4
$\mathbf{p_{WeightReplaced}}$ chance to replace weight	0.05
$\mathbf{p_{Connection}}$ chance to enable / re enable a connection	0.01
$\mathbf{max_{Perturb}}$ maximum allowed change on weight	0.75
$\mathbf{p_{AddLink}}$ chance to add a link	0.1
$\mathbf{p_{AddNode}}$ chance to add a node	0.05
Species parameters	
$\mathbf{species_{Target}}$ number of target species.	2
$\mathbf{coeff_{Excess}}$ used for species compatibility score	1
$\mathbf{coeff_{Disjoint}}$ used for species compatibility score	1
$\mathbf{coeff_{Weight}}$ used for species compatibility score	0.7
$\mathbf{threshold}$ used for species compatibility score	2
$\mathbf{threshold_{Change}}$ used to change threshold value	0.1
$\mathbf{species_{AgeThreshold}}$ age to count as old	8
$\mathbf{species_{YouthThreshold}}$ age to count as young	3
$\mathbf{age_{Penalty}}$ fitness multiplier for old individual	0.5
$\mathbf{age_{Boost}}$ fitness multiplier for young individual	1.2
Other parameters	
$\mathbf{size}$ population size of all robots combined	24
$\mathbf{survival_{Threshold}}$ top % individuals that can be parents	0.6
$\mathbf{tournament_{size}}$ size of tournament to select parent	2
$\mathbf{copy_{Best}}$ clone best specie individual previous generation	TRUE
$\mathbf{copy_{BestEver}}$ clone best individual so far	FALSE

all experiments, 10 replicate runs are performed with different random seeds.

The code for the implementation is available on the first author's website. The most important parameter settings for the NEAT algorithm are presented in Table I.

V. Experimental Results

In this section, the experimental results of the tasks reported in Section III using the setup described in Section IV are discussed. Due to the small amount of runs, some explanations are more qualitative than quantitative.

A. Performance

In Figure 5 we compare the median and maximum fitness, including interquartile range, over generations for the obstacle avoidance task (left) and the foraging task (right). The individual learning robot is presented by the green colour. Robot-to-robot learning with 4 robots is presented in yellow and with 8 robots in red. From these graphs, we can draw several conclusions.

First, for both tasks the performance improves over time, meaning that the robots learn over time. For the obstacle avoidance task, the median performance is much closer to the maximum performance than for the foraging task. This indicates that the foraging task is more difficult to learn for the robot. Second, adding robot-to-robot learning, results in a more robust median performance, showed by a smoother curve. Although the median performance for the foraging task seems to increase when using more robots, the maximum fitness seems to be lower when using 8 robots when looking at the interquartile range.

Figure 6 shows a more detailed view of the performance over time for the obstacle avoidance task (top) and foraging task (bottom). It specifically shows the fitness of the individuals over the generations for all 10 independent runs and all robot-to-robot learning experiments. To explain this graph, pick one bar where the colour goes from green to red. This bar consists of dots, where each dot represents one individual in the generation presented at the top of the column and the colour represents the fitness of the individual. There are groups of 10 bars where each bar represents one run. This block of 10 runs is shown for every generation, shown at the top of the column, and the number of robots, shown at the right of the row. When using multiple robots, the individuals of the final generation for all robots are combined and sorted by fitness.

We can see that for the obstacle avoidance task, there are many well-performing controllers, while for the foraging task the good controllers are sparse. However, the effect of robot-to-robot learning seems to be similar: when a high performing controller is found, there is more chance that the next generation has a high performing controller too. This results in the smoother median curve of Figure 5. The difference in maximum performance for the foraging task shown in 5 can also be better understood with this graph. It seems that a group of 4 robots is more capable of keeping the good knowledge in the population than using a group size of 8. This might be due to the decrease in population size from 6 to an even smaller size of 3 when using 4 and respectively 8 robots.

B. Network complexity

In Figure 7 we can see the median and interquartile range of the average number of edges in the final generation over the 10 replicate runs for the obstacle avoidance task (left) and foraging task (right). The network complexity is expressed as the number of edges, as the increase in the number of nodes already implies the increase in the number of edges. We can confirm our hypothesis that the complexity of the network significantly drops when the robotic group size increases, indicated by the non-overlapping quartile range [13]. However, we must note that when using more robots, the chances of creating extra nodes and edges go down because a higher

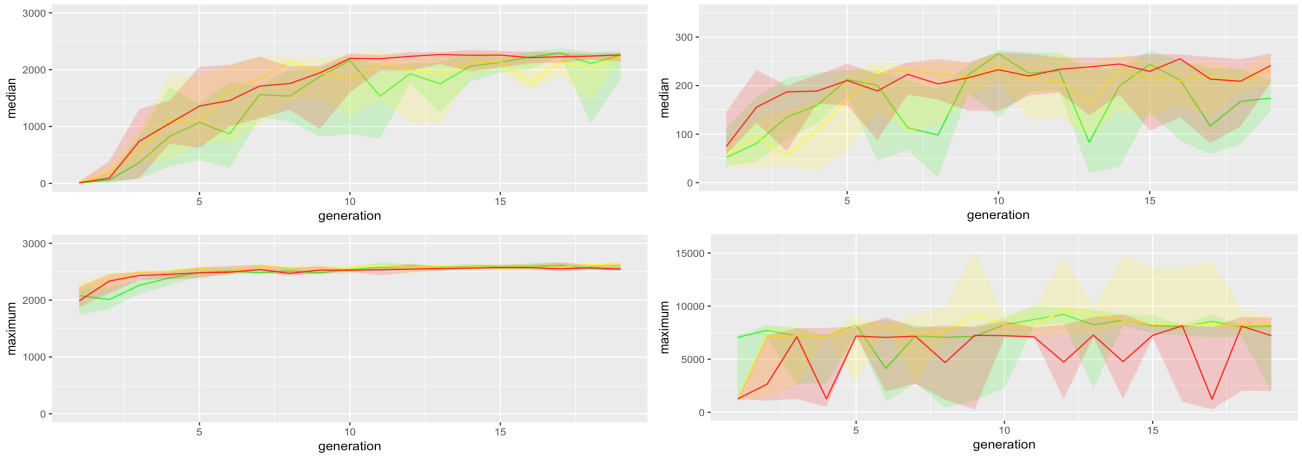

Figure 5: Median (top) and maximum (bottom) performance with interquartile range over generations for the obstacle avoidance (left) and foraging task (right). The individual learning robot is presented by the green colour. Robot-to-robot learning with 4 robots is presented in yellow and with 8 robots in red. The results are compiled over 10 replicate runs.

percentage of the population is used by the clone component of the algorithm.

C. Selection pressure

Although the algorithm that is running on every robot is the same and the same parameters are used, the overall dynamics of the system are different. An analysis of the selection pressure will show this. The selection pressure expresses the correlation between the fitness of an individual and the number of offspring. We specifically use the Kendall's τ-b measure explained in [14]. In short, it measures the correlation between fitness and number of offspring. A high value for τ-b indicates a strong correlation between fitness and offspring and therefore high selection pressure.

In Figure 8 the selection pressure over generations are shown for 1 (green),4 (orange) and 8 (red) robots. On the left, the selection pressure is aggregated over all robots, while on the right the selection pressure for the first robot is isolated, meaning only one robot is chosen in the robot-to-robot learning experiments to express the selection pressure.

Looking at the left graph, we can observe that the selection pressure for 1 robot is a bit higher than for more robots. This is logical because the whole population is divided over multiple robots and only the population of one robot is considered to create offspring. While the overall best controller can be selected every time for the 1 robot experiment, it does not participate in the creation of offspring on the other robots. As a result, the correlation between fitness and number of offspring is not as high as the correlation of 1 robot.

The right graph shows the selection pressure on one robot (robot number 1). Therefore, the selection pressure for the 1 robot experiments is identical to the left graph. For the robot-to-robot experiments, we see an increase in the selection pressure but also an increase in volatility. This is because, on one robot, the population size decreases and the best controller has more impact on the creation of offspring when using tournament selection.

VI. Discussion and Conclusion

In this paper, the effect of robot-to-robot learning on the learning speed, performance and network complexity using physical robots is investigated for two tasks.

We have shown that the median performance is more robust when using more robots. Although the increase in robustness is not overwhelming, robot-to-robot learning at least distributes the search and reduces the wall clock time needing to learn a specific task without loss of performance compared to an individual learning robot.

The increase in robustness of the median performance is probably because more robots are more likely to retain the controllers of the good performing controllers. A possible explanation for this is that good controllers are tested by multiple other robots that all start at a different position. Even though the fitness function of a foraging task is very stochastic, a good controller will probably perform well on one of the other robots.

Additionally, we have shown that using more robots results in less complex controllers. While this might be a logical result of our specific implementation details, the fact remains that we can reach a similar performance level with less complex controllers when using multiple robots. A decrease in controller complexity means that the search space is reduced: this is especially useful in a physical setup where the time is an important restriction.

Observing the selection pressure showed that the overall dynamics change when using multiple robots, even though the algorithm on the specific robot remains identical. We showed that the selection pressure over all robots decreases while the selection pressure on the individual robot increases. This might result in an overall better exploration versus exploitation balance.

Figure 6: Fitness of the obstacle avoidance task (top) and foraging task (bottom). The x-axis presents the individuals per generation. The y-axis presents the run. The rows represents the results for 1, 4 and 8 robots. Within one run, the individuals are sorted on fitness of which the colour reflects the value. When using multiple robots, the individuals of the final generation for all robots are combined and sorted on fitness.

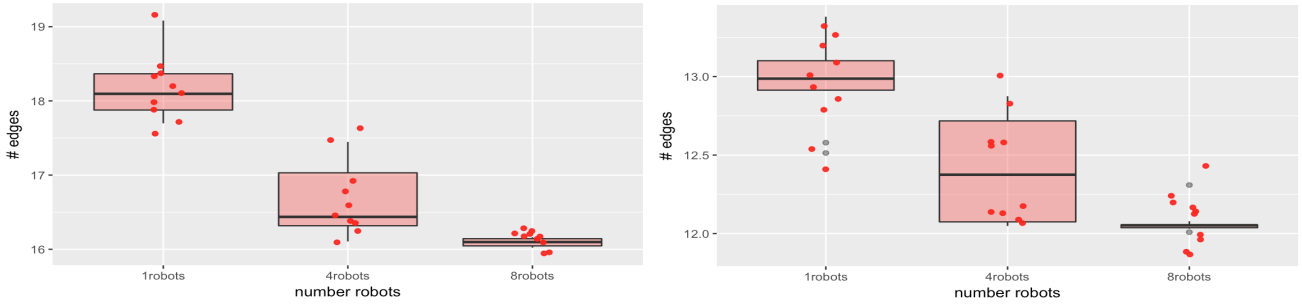

Figure 7: Median network complexity with interquartile range at the final generation for the obstacle avoidance task (left) and foraging task (right). Network complexity is expressed as the average number of edges in the network. The results are compiled over 10 replicate runs.

One can argue that our specific implementation of robot-to-robot learning has a link with parallel EAs and island models [15], [16]. Most of the work in parallel EAs and island models are focussed mainly on runtime analyses [15]. Measuring this in ER is of lower importance because the evaluation time of the robot is much larger than the computational effort. Additionally, the fitness function in evolutionary robotics is extremely stochastic. This is due to the specific location of the robot and the behaviour required in that location. This relevance of the location of the robot is not present with parallel EAs and island models. Despite the differences, we do believe that there are some common elements too. Especially, studying the effect of the number of islands/robots on the diversity of the whole population is of interest to both fields.

It is clear that using multiple robots changes the dynamics of the learning algorithm. However, based on the limited number of physical experiments executed in this paper, we are unable to identify the explicit ”magic” of robot-to-robot learning. In future work we will return to a simulation platform for an in-depth analysis of the impact of robot-to-robot learning. Because the robots learn in an online fashion, the learning mechanisms in the physical robots and the simulation platform will be identical. Therefore, we can use results in simulation to validate the physical experiments.

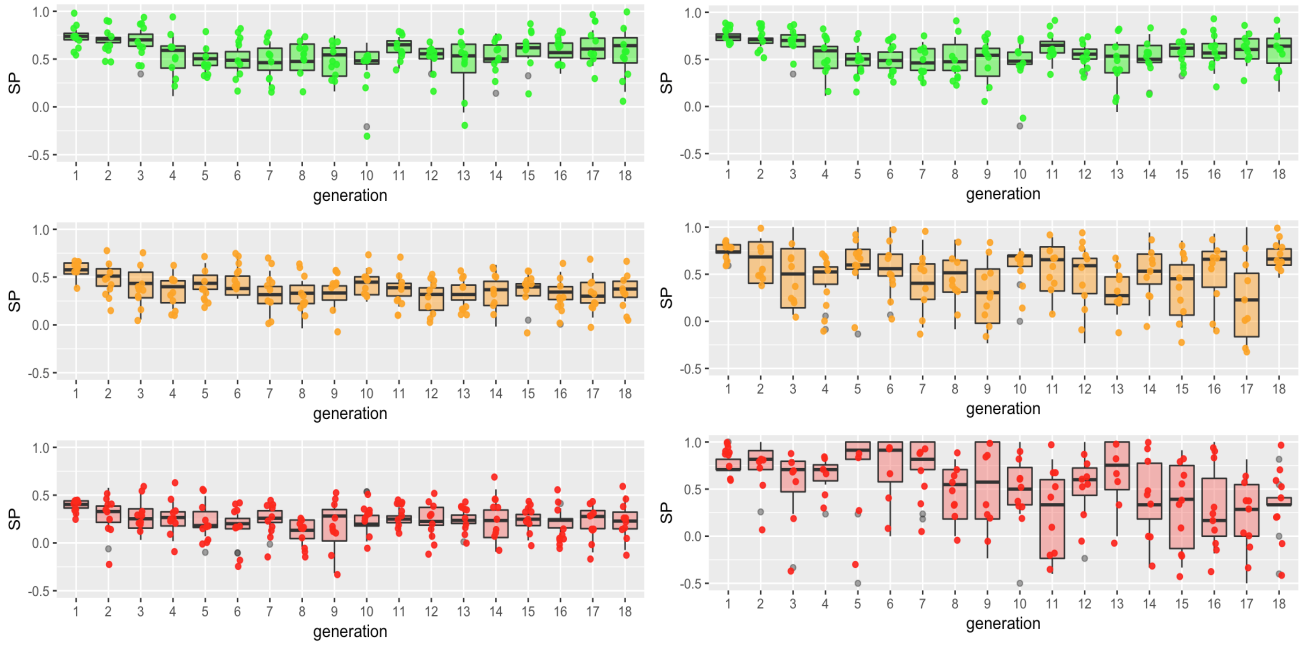

Figure 8: Selection pressure over generations for 1 robot (green), 4 robots (orange) and 8 robots (red), including interquartile range over generations for the foraging task combined over all robots (left) and robot number 1 (right), meaning that the results of only one robot are presented for the robot-to-robot learning experiments. The results are compiled over 10 replicate runs.

To conclude, this paper showed some promising results when applying robot-to-robot learning. It is shown that a complex task, such as the foraging, can be learned within the hour due to robot-to-robot learning. Therefore, online learning methods can be tested for more complex tasks than currently possible when using robot-to-robot learning. This will hopefully help the field of ER to be a worthwhile alternative to hand-coded robots for an environment not fully known to the designers.

VII. Acknowledgments

This work is supported by the European Union's Horizon 2020 research and innovation programme under grand agreement No 640891 (DREAM project). A special thanks to Thomas Webbers for his support with the experiments.

References

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

[2] W. Tansey, E. Feasley, and R. Miikkulainen, "Accelerating evolution via egalitarian social learning," in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, T. Soule, Ed. New York, NY, USA: ACM, 2012, pp. 919–926.

[3] B. P. Jolley, J. M. Borg, and A. Channon, "Analysis of social learning strategies when discovering and maintaining behaviours inaccessible to incremental genetic evolution," in *International Conference on Simulation of Adaptive Behavior*. Springer, 2016, pp. 293–304.

[4] R.-J. Huijsman, E. Haasdijk, and A. E. Eiben, "An On-line On-board Distributed Algorithm for Evolutionary Robotics," in *Artificial Evolution, 10th International Conference Evolution Artificielle*, ser. LNCS, J.-K. Hao, P. Legrand, P. Collet, N. Monmarché, E. Lutton, and M. Schoenauer, Eds., no. 7401. Springer, 2011, pp. 73–84.

[5] F. Silva, L. Correia, and A. L. Christensen, "A case study on the scalability of online evolution of robotic controllers," in *Portuguese Conference on Artificial Intelligence*. Switzerland: Springer International Publishing, 2015, pp. 189–200.

[6] J. Heinerman, J. Stork, R. Coy, J. Hubert, A. E. Eiben, T. Bartz-Beielstein, and E. Haasdijk, "Can social learning increase learning speed, performance or both?" 2017.

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

[8] P. J. O'Dowd, M. Studley, and A. F. T. Winfield, "The distributed co-evolution of an on-board simulator and controller for swarm robot behaviours," *Evolutionary Intelligence*, vol. 7, no. 2, pp. 95–106, 2014.

[9] F. Silva, L. Correia, and A. L. Christensen, "Evolutionary online behaviour learning and adaptation in real robots," *Royal Society open science*, vol. 4, no. 7, p. 160938, 2017.

[10] J. Heinerman, A. Zonta, E. Haasdijk, and A. E. Eiben, "On-line evolution of foraging behaviour in a population of real robots," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2016, pp. 198–212.

[11] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

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

[13] 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*, vol. 3, no. 1, p. 34, 2003.

[14] E. Haasdijk and J. Heinerman, "Quantifying selection pressure," *Evolutionary computation*, no. Early Access, pp. 1–23, 2017.

[15] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE transactions on evolutionary computation*, vol. 6, no. 5, pp. 443–462, 2002.

[16] D. Whitley, S. Rana, and R. B. Heckendorn, "The island model genetic algorithm: On separability, population size and convergence," *Journal of Computing and Information Technology*, vol. 7, no. 1, pp. 33–47, 1999.