The Distributed Co-Evolution of an Embodied Simulator and Controller for Swarm Robot Behaviours

Paul J. O'Dowd
Bristol Robotics Laboratory
University of the West of England
Bristol, UK
Email: paul.odowd@brl.ac.uk

Alan F.T. Winfield
Bristol Robotics Laboratory
University of the West of England
Bristol, UK
Email: alan.winfield@uwe.ac.uk

Matthew Studley
Bristol Robotics Laboratory
University of the West of England
Bristol, UK
Email: matthew2.studley@uwe.ac.uk

***Abstract*—Embodied fitness assessment of robotic controllers is slow but grounded, while assessment in a simulated environment is fast but can run foul of the 'reality gap'. We present a distributed co-evolutionary method to adapt the environmental model of an on-board simulator within the context of swarm robotics.**

I. Introduction

The design of robot controllers for swarm robotics is recognised as non-trivial[11]. A robot controller must be coded to sufficiently capture the dynamics of interaction between robots and the robots and the environment, in order to lead to the emergence of useful self-organised group behaviour[15].

Recent works in Evolutionary Swarm Robotics demonstrate it is possible to automate the design process through the use of simulation and evolutionary algorithms (EAs). The approach is well suited to swarm robotics; a simulation provides access to global metrics of group behaviour for EA evaluation[14][12]. The use of EAs and simulation has formed a simulate-and-transfer method, terminating evolutionary development for a real world demonstration.

Similarly to König et al[7] and Baele et al[1], we are interested in running an EA on-board robots evolve swarm robot controllers. An on-board EA has the advantage of being an open-ended method of adaptation[10]. The work of Watson et al[16] highlights the benefits of distributing an EA across many real robots, to which swarm robotics seems well suited.

However, to retain the scalable, flexible and robust qualities of a swarm[11], an on-board and distributed EA must be decentralised. Therefore robots are constrained to perceive only local information and thus an EA loses the global metric advantageous to evolution in the swarm robotic context.

As a solution to the loss of global evaluative scope, we propose that each robot in a swarm has an embedded simulator to evaluate a local population of controller solutions, similar to the Island Model[2]. Decentralised distribution is achieved via local communication of controller solutions between robots.

An embedded simulator also presents an alternative to the evolutionary robotics method of time-shared physical evaluation of an evolutionary population on one robot[5][13]. An embedded simulator could evaluate many controllers in the time required to physically evaluate a single controller, therefore avoiding periods of physical validation behaviour.

However a flaw is inherent to this proposal and forms the subject of this paper. If the distributed evolution of robot controllers evaluates only through embedded simulation, then the proposal runs foul of the 'reality gap'[6], where performance outside the simulator is unknown.

The reality gap can be considered in three categories of correspondence between simulation model and reality. First, robot-robot correspondence refers to physical robotic aspects, such as differences in morphology. Second, robot-environment correspondence refers to differences in the dynamic interactions between a robot and the environment, both sensory and through actuation. The third case, environment-environment correspondence relates the representation of salient features of the environment. To stop an EA exploiting erroneous representation, all three categories should be modelled with sufficient fidelity or supplemented with sufficient noise[6].

The recent works of Bongard and Lipson[3] and Bongard[4] investigate on-line adaptive self-models to address robot-robot and robot-environment correspondence discrepancies. In the swarm robotic context, decentralised mechanisms of self-organisation make swarms particularly susceptible to changes in the environment[12]. Therefore, we make the focus of this paper a specific investigation of a swarm of robots to identify discrepancy in environment-environment correspondence.

To serve this purpose, we work within an EA framework to form a co-evolutionary approach for the on-line adaptation of both robot controllers and embedded simulator environment model. We aim to make good use of the swarm of robots by distributing the evolution of the environment model across all robots. Secondly, robots individually evolve a population of robot controllers via the adaptable embedded simulator to update the physical robot controller.

We hypothesise that a measure of environment-environment correspondence of the embedded simulator can be gained indirectly through the real-world performance of the robot to complete a task. The evolution of the robot controller will exploit features of the simulated environment, and therefore a higher utility will be gained in the real world by a robot with a stronger environment-environment correspondence.

This indirect evolutionary method of validating the environment model means that a robot does not need to spend time explicitly assessing and validating features of the environment, or directly relating the robot controller performance to the environment model. Instead, the variation of embedded simulator models across the swarm, and each robots' relative utility to complete the designated task, should serve as a competitive selection mechanism to collectively converge towards strong environment-environment correspondence.

Whilst the simulator evolution is collective, we select a task with no inter-robot dependency and keep the robot controller evolution isolated between robots. In future research, we wish to continue from the basis of this paper to research the distributed evolution of self-organising robot controllers via an embedded simulator.

We conduct our investigation with a series of experiments with real robots to complete a foraging task. We test for the ability of the swarm to evolve an embedded simulator with good environment-environment correspondence varying the real task environment. In Section II we provide greater detail on the workings of the distributed co-evolved method. In Section III we explain the experiment task and the algorithms used. In Section IV we provide and discuss the results of the experiments. Section V provides concluding remarks and outlines how we would like to progress this research.

II. The Distributed Co-Evolutionary Method

The proposed co-evolutionary method has two evolutionary components. One evolutionary algorithm is distributed across the physical swarm and evolves the environment model. The second evolutionary algorithm is local to each robot and is dependent on the robot's embedded simulator. The method works within two time domains. The utility of the environment model is evaluated as a function of real-time performance of the physical robot. Controller evaluation is conducted virtually within the embedded simulation at a rate faster than real time.

Each robot owns only one environment model which it can communicate to adjacent robots, creating a distributed evolutionary population. Each robot also has a private evolutionary population of many robot controller solutions. Evolutionary selective pressure is distinct; environment models relate to real-world performance whilst robot controllers relate to simulated performance.

Fig. 1 illustrates in higher detail the processes involved for each robot. Three major processes happen in a synchronised manner; the virtual evaluation of robot controller solutions, the real-world fitness assessment of the instantiated controller, and the reception of other robots' environment model variants.

Each robot continuously communicates their environment model variant and current real world fitness value as they operate. Each robot constructs a temporary population of received environment model variants from encountered robots. At the end of a robot's real world evaluation period, a new environment model is selected and evolved from the constructed temporary population. The temporary population is then cleared, starting the next environment model evolutionary period. Further details are given in the following sections.

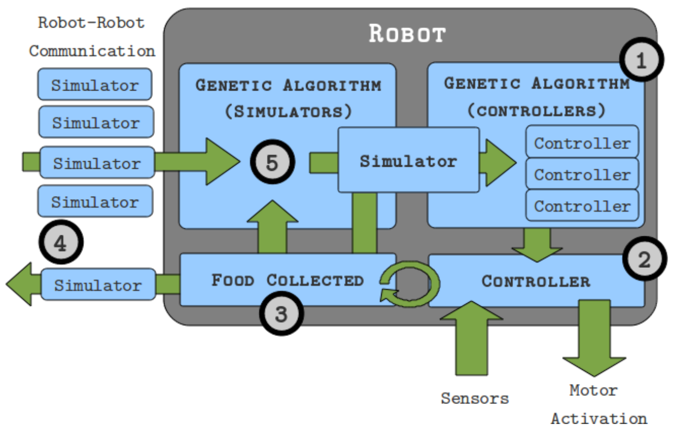

Fig. 1. Addressing numbered points: 1) A genetic algorithm evolves a local population of controller solutions through the embedded simulator. 2) The best controller from simulation is transferred to the real robot. 3) Food collected by the real robot is used to measure the fitness of the embedded simulator. 4) A robot transmits and receives embedded simulator genomes and real robot fitness values. 5) Synchronised with the end of virtual controller evaluation, the embedded simulator is evolved against the robots own perceived fitness and any encountered robots' fitness values.

The proposed co-evolutionary method works on the principle that many controller solutions can be evaluated virtually within the time required to assess the real world performance of the physical robot. The difference in the evaluation time allows the environment model to maintain a degree of stability whilst robot controllers are evolved virtually.

III. Experimental Setup

This section provides a description of the experiment and all parameters used to investigate the distributed co-evolutionary method. We have constructed a simple experiment in order to reduce the number of free variables in the experiment.

A. Task Description

We have selected the foraging problem, where robots must search for food items and deposit them back at a designated nest site. We vary the environment to alter the potential efficiency of foraging. Robots are unable to identify their own position, the location of any food, or the location of the nest site. Robots can only search for food using a random search movement. In order to test for adaptation, we construct our experiments around variation of a significant light source in the task environment which has the potential to be used as a navigational landmark to deposit food.

We construct three basic *environment scenarios* (Fig.2); a light source at the nest site (A), no light source (B), or the light source opposite the nest site (C). If the swarm collectively identify the environment scenario, phototaxis should provide a means to improve the efficiency of a robot's foraging. We have combined the three basic environment scenarios into five *experiment cases*; *No Light Source*, *Light Fixed At Nest*, *Light Fixed Opposite Nest*, *Light At Nest* $\rightarrow$ *Light Opposite Nest*, *Light Opposite Nest* $\rightarrow$ *Light At Nest*.

In the first three experiment cases, we test for a correct initial adaptation and the ability to maintain the adaptation. In the latter two experiment cases, the light source location is moved half way through the duration of an experiment, and we are testing for re-adaptation and continued foraging.

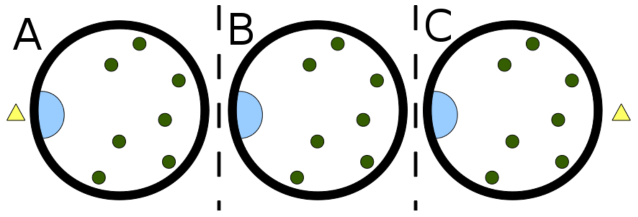

Fig. 2. The three environment scenarios (not to scale). Large circular outlines represent the arena enclosure. Small shaded circles represent food. Large shaded circle represents nest area. Shaded triangle represents light source location (when present).

B. E-Puck & Linux Extension Board

We use ten *open-hardware e-puck educational mobile robots* [9] (Fig. 3). We multi-purpose the infra-red proximity sensors (IR) for obstacle avoidance, determining ambient light levels, and for short range IR communication between robots.

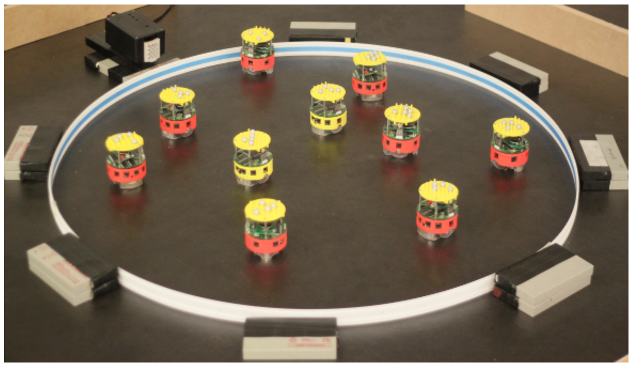

Fig. 3. Ten e-puck robots, each with a Linux extension board, inside a circular 1.2m diameter enclosed arena. Reflective markers are placed on top of each robot for tracking. The black box with a white window in the top left of the image provides the significant light source added to experiments.

IR communication is transmitted and received through all 8 IR sensors to a 20cm distance. Each robot is equipped with a Wi-Fi enabled Linux Extension Board (LEB) developed at the Bristol Robotics Laboratory [8], performing all processes as multiple threads. The e-puck is driven in a master-slave configuration from the LEB.

C. Wi-Fi Messenger Service

Local communication is considered important for decentralised systems. We use the IR communication of the e-puck robots to initiate a dialogue over Wi-Fi. Therefore communication is constrained to the transmission distance of IR but provides the superior bandwidth of Wi-Fi. After an initial IR transaction, robots update and retrieve information from a central authority. This mechanism appears local and decentralised to the robots and provides us with the ability to log all transactions.

D. Vicon Tracking System & Virtual Sensor

A vision tracking system from Vicon^{TM} is used to monitor the positions of each robot. A robot position is identified by a unique pattern of reflective markers placed on top of each robot (see Fig.3). Position data has two uses; to provide data for post-experiment analysis and to provide a virtual sensor to each robot. Every 40ms an external program transmits over Wi-Fi if a robot is located directly on a virtual food item or the virtual nest site. Each robot updates its own control states with respect to this virtual sensor.

E. Embedded Simulator

A minimal simulation has been written in the C programming language which executes on the LEB. The simulation is trigonometric where all entities are represented on a 2D plane as points with a radius. The simulated robot is moved on the basis of two-wheel differential kinematics and the dimensions of the e-puck robots. The IR sensors are emulated from raw sensor data that has been captured from the real robot sensors to which uniform noise is added. The simulation is advanced in 40ms time intervals. A genetic representation (Section III-H) determines one of three environment scenarios to simulate (Section III-A), mirroring the possible real environment. For each robot controller genome to evaluate (Section III-G), one robot is simulated to forage for 60 virtual seconds.

F. State Based Behavioural Controller

To control the robot we implement a state-based behavioural controller. Four hand-coded basic behaviours have been tested to function both in the embedded simulator and real world: *obstacle avoidance*, *random search*, *positive phototaxis* and *negative phototaxis*. The robot controller tracks two states, *Food* and *Nest*, which are either *True* or *False*. Using a virtual sensor (Section III-D), a robot will pick up a food item changing *Food* to *True*. A robot can only drop food when *Nest* $=$ *True*, returning *Food* to *False*. The same controller method is used on a simulated robot and physical robot.

Value	G_0 (Food = True)	G_1 (Food = False)
[0.00 : 0.32]	*negative phototaxis*	*negative phototaxis*
[0.33 : 0.65]	*random search*	*random search*
[0.66 : 0.99]	*positive phototaxis*	*positive phototaxis*

Fig. 4. Genes G_0, G_1 mapping of state to behaviour selection, providing a variety of possible responses to the state *Food* of the robot.

Regardless of state the *obstacle avoidance* and *random search* behaviours are always active, providing a robot with the ability to explore the environment with random motion. A genetic representation selects *positive phototaxis*, *negative phototaxis* or no additional behaviour, in response to the *Food* state of the robot (see Section III-G).

G. Local Evolution of Robot Controller

Each robot operates a simplistic steady state genetic algorithm (GA) to adapt a state-based behavioural controller (Section III-F). The GA maintains a population of 10 genomes, each composed of 2 genes (G_0, G_1) in the range [0.00 : 0.99]. G_0 corresponds to state *Food* = *True*. G_1 corresponds to state *Food* $=$ *False*. The values of G_0, G_1 are mapped to select a behaviour, as per Fig.4.

Selection for reproduction is rank based and elitist. 40% of the population is used to overwrite the lower ranking percentage. Each gene of the child genome is subjected to a 20% chance of a random mutation on a Gaussian distribution (mean = 0, s.d = 2). Mutation is the only mechanism to introduce variation. The fitness of each genome is determined by evaluating the performance of the controller phenotype in the embedded simulator as summation of deposited food as a function of time:

$$F = \sum_{D=1}^{D_{Total}} T_{Max} - T_D \tag{1}$$

where F is the derived fitness metric, D is a deposited food item, T_{Max} is the evaluation time limit of 60 seconds, T_D is the recorded time to successfully deposit a food item. Time has been used rather than quantity of food for stronger differentiation between efficiency in solutions. When all 10 genomes have been evaluated in the embedded simulator, the genome with the highest simulated fitness value is immediately instantiated for use with the real robot.

H. Distributed Evolution of Simulator

The environmental model of the embedded simulator is determined by the value of S_0 (see Fig. 5). Each robot maintains one value of S_0 for the duration of a complete generation of controller evaluation. S_0 is evolved at the end of controller evolution, updating the embedded simulator and creating a periodically stable environment model.

Value	S_0
[0.00 : 0.32]	*Light Opposite Nest*
[0.33 : 0.65]	*No Light*
[0.66 : 0.99]	*Light At Nest*

Fig. 5. Gene S_0 mapping to the embedded simulator scenario.

As a real robot operates in the real world it transmits it's current S_0 and current real world fitness value (F_R), and receives those of other operating robots. F_R is determined as the robot operates by the same equation used in simulated evaluation (see equation.1). A temporary population of 10 encountered S_0:F_R pairs are stored and updated by each robot, representing the variation and fitness of environment models across the swarm. Selection from the S_0:F_R pairs is rank based elitist and always subjected to a random mutation on a Gaussian distribution (mean = 0, s.d = 2).

Importantly, an individual robot must compare its own S_0:F_R pair against the S_0:F_R values encountered from other robots. The evaluation, selective pressure and evolution of S_0 is therefore distributed across the swarm. With fewer than two robots, there is no selective pressure to direct the EA of S_0. A robots accumulated population of foreign S_0:F_R pairs and own F_R value are cleared at the update transition of controller and environment model. We observed an average of 34 real-time seconds to conduct a full generation of simulated controller trials. We chose to pad controller evaluation time to 60 real-time seconds to give a robot sufficient time to evaluate in the real world and accumulate foreign S_0:F_R pairs.

I. Experiment Settings

We run the five experiment cases (Section III-A) 10 times, each for a duration of 60 minutes. Any light source relocation occurs at the 30 minutes interval. The experiments are conducted within an enclosed circular arena of diameter 1.2m free of obstructions. A single circular nest site of radius 20cm intersects the arena boundary (see Fig 2) and maintains the same coordinates through all experiment runs. A food density of 7 randomly placed items is maintained irrespective of collected food items. We use a total of 10 robots, each randomly positioned within the arena. All the robots are manually started asynchronously in random order within seconds of each other.

For comparison, we also run three variations of fixed optimal controllers (no evolutionary components) 10 time each. The optimal controllers match the stable environment situations *No Light*, *Light At Nest* and *Light Opposite Nest*, combining the search behaviour with the appropriate phototaxis when a light source is present. These supplementary trials are otherwise run with the same circumstances of the main experiments.

IV. Results & Discussion

The following compiles the results of the 10 test runs for each experiment case presented in Section III-A. First we distinguish the overall performance of each experiment case by the total food collected. We then investigate the efficiency of foraging in each experiment by the time before and after a change to the environment (if any). Finally, we discuss the convergence of the swarm on the parameter S_0 over time for each experiment case which ultimately determines the swarm behaviours.

A. Food Collected

Fig 6 shows the quantity of total food items deposited at the nest site in each experiment. First, the fixed search controller performed as poorly as the co-evolutionary method in the *No Light* scenario. These cases are complimentary in that robots were unable to perceive a light source, and sets a bench mark for a worst case foraging efficiency.

The experiment case *Light Fixed At Nest* and the fixed positive phototaxis controllers both performed the most efficiently. Compared to the absence of a light source, this indicates that the swarm was able to realise a utility in the addition of a light source to the environment.

The *Light Fixed Opposite Nest* experiment performed only marginally better than the scenarios where a search behaviour was the only viable solution. However, the fixed negative phototaxis controller performed exceptionally well, which indicates that either the swarm was unable to converge on the environment scenario or evolve a competent controller. Examining the S_0 variable should provide more information.

In both experiments with an environmental change, the total food collected does not match the better two fixed optimal controllers. This inefficiency may be reasonable, given that the swarm must converge on the environment model and subsequently evolve a controller. The *Light At Nest* $\rightarrow$ *Light Opposite Nest* experiment has bias in favour of the prior *Light*

At Nest scenario, whilst the *Light Opposite Nest* $\rightarrow$ *Light At Nest* has a more balanced collection of food.

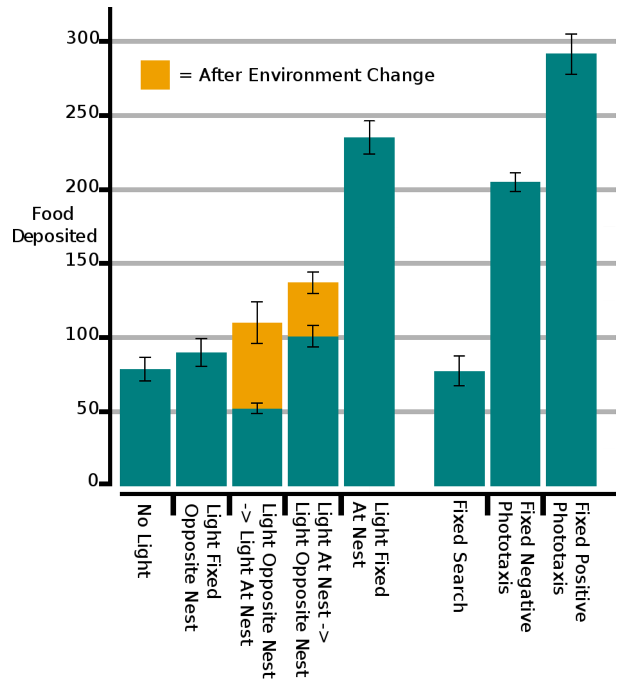

Fig. 6. The total food collected in each experiment case, showing average and the distribution of results for all tests. A double shading separates the food deposited before and after a relocation of the light source. The last three bars represent success of the three optimal case controllers.

This may be a result of faster convergence on the *Light At Nest* scenario allowing the robots a longer period of higher foraging efficiency. This may be because moving away from a light source is much more ambiguous than moving towards a light source, and therefore converging on the *Light Opposite Nest* would take longer. This process of convergence is absent in the fixed optimal controllers.

B. Collection Rate

Fig. 7 plots the average time taken for the swarm to deposit each food item at the nest site. Each experiment case has been split at *T* < *30 minutes* and *T* > *30 minutes* to view changes in the performance of the swarm with respect to any change in the environment. A lower retrieval time indicates an efficient foraging rate of food.

In the three fixed scenario experiment cases the distribution of results decrease beyond *T* > *30 minutes* indicating an improvement in foraging efficiency. This is likely to be due to successful robots propagating and reinforcing a good environment model, and therefore increasing the parallel activity by the swarm. The small distribution of data for *Light At Nest* emphasises the efficiency of the scenario, where more repeatable behaviour between robots reinforces the embedded simulator, providing behavioural stability.

In both experiment cases with scenario change we see that the swarm is able to re-optimise the foraging behaviour, indicated by a continued average rate of foraging through *T* > *30 minutes*. However, both *Light Opposite Nest* $\rightarrow$ *Light At Nest* and *Light At Nest* $\rightarrow$ *Light Opposite Nest* show a significant growth in the distribution after the environment transition. Initially this may seem surprising given that in the *Light Opposite Nest* $\rightarrow$ *Light At Nest* case the environment was altered to the more favourable light scenario.

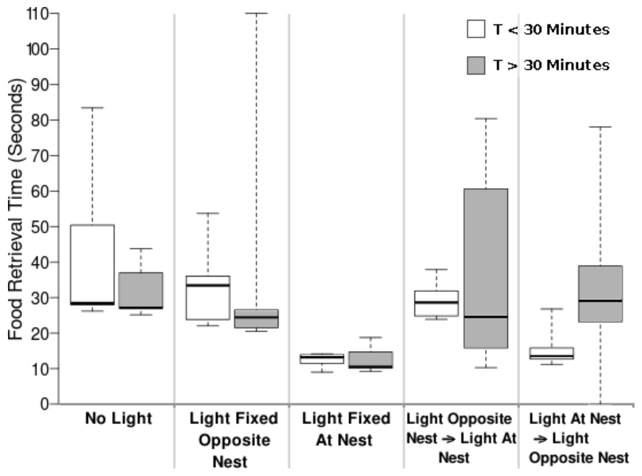

Fig. 7. The time taken to return food to the nest in each experiment case, showing the distribution of results for all tests before and after any change to the environment.

The significantly larger spread in foraging efficiency for the *Light Opposite Nest* $\rightarrow$ *Light At Nest* at *T>30 minutes* can be explained through observation. A prior convergence on negative phototaxis to locate the nest can incorrectly drive the swarm to the opposite arena extremity. Any adapted robots successfully foraging are likely to be using positive phototaxis and working in relative seclusion at the other end of the arena. This spatial segregation persists until the random search behaviour can spatially transfer the correct adaptation to be exploited by the whole swarm.

C. Embodied Simulator Gene, S_0

Fig.8 shows the progression of the S_0 gene through time for each experiment case. The graphs are shaded into 3 horizontal regions, corresponding to the 3 experiment scenarios that gene S_0 maps to (see Section III-A). Each box on the plot is a sample of the average S_0 value for that time period.

Both cases *No Light* and *Light Fixed Opposite Nest* show the value of S_0 fluctuating through all 3 horizontal regions during the course of the experiments. The majority of the time is spent in the middle region, which maps to a *No Light* scenario. The correct convergence in the first case, but it demonstrates the inability of the swarm to converge on the *Light Opposite Nest* scenario in the second case. As a result, both experiments will have evolved a controller based on the search behaviour, which explains the similar food deposition totals. However, the first half of *Light Opposite Nest* $\rightarrow$ *Light At Nest* is in conflict to this observation, and highlights that convergence in the *Light At Nest* scenario cannot be guaranteed.

The graph for *Light Fixed At Nest* shows a definite trend for the value of S_0 to occupy the upper region, signifying a correct convergence on the *Light At Nest* embedded simulator. Again, this matches the collection rate and collected food for this experiment case, and produces the positive phototaxis behaviour.

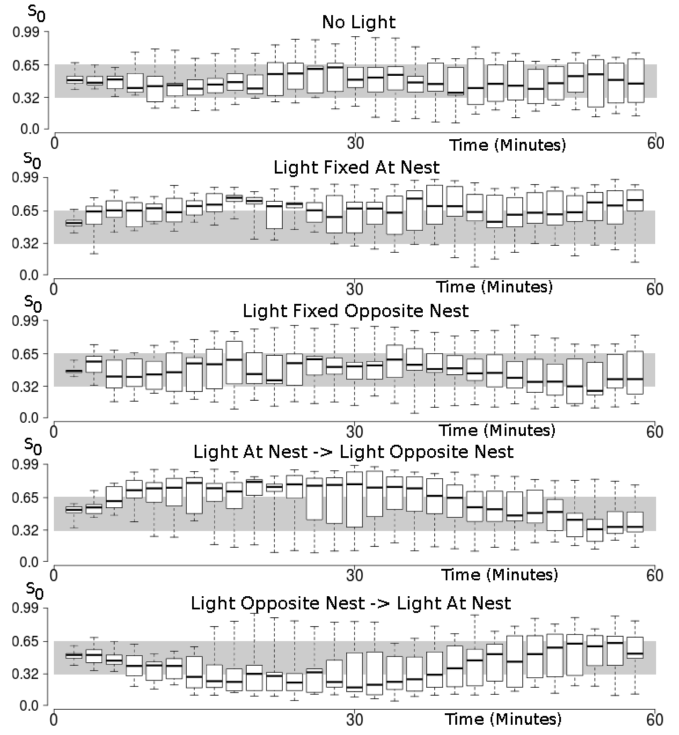

Fig. 8. Graphs of the sampled S_0 gene value over time for each experiment case conducted. Each graph has been shaded horizontally to highlight the mapping of S_0 to an environment model scenario variant (see Section 2). We are interested in how the value of S_0 changes between these horizontal regions with respect to time. The bottom two graphs have a real world environmental change at 30 minutes.

The last two graphs of Fig. 8 are the experiment cases with a change in the environment at $T = 30$. In both cases, convergence sustains for approximately 7 minutes after the environment change at $T = 30$, confirming our assertion that a period of re-adaptation would reduce foraging efficiency.

V. Conclusions & Future Work

In this paper we have outlined a distributed co-evolutionary method to validate the environment-environment correspondence of an embedded simulator. The method is fully decentralised and working within the criteria for swarm robotics. We demonstrated flexibility in swarm behaviour to changes in the real world task environment by the virtue of the embodiment of a simulator. The results also show an improvement in foraging efficiency in stable environments.

The unreliability of convergence on the *Light Opposite Nest* scenario is an interesting issue. There are many reasons why this artefact may occur, but principally the environmental differentiation of a light source is biased in its utility. With this consideration, the mis-convergence of the swarm on the environment model can be viewed as an adaptation to retain a competitive level of activity.

In order to prove the principle the presented experiments have made a number of simplifications. In particular, the discrete simulator representations are an external classification by the designer of the system. The feasibility of this aspect needs further investigation. However, automated methods for isolating environmental features could be integrated within the distributed co-evolutionary method presented in this paper.

Whilst we highlight the evolutionary utility of global assessment in swarm robotics and use this as a primary motivation, the experimental task of foraging has no interdependency between operating robots. In the future we intend to utilise this methodology to investigate the possibility of a robot simulating itself interacting with other encountered robots. In this way, the embedded simulators will be collectively evolved, and the swarm behaviour will be virtually evolved specifically in terms of interacting robots. We tentatively hypothesise that when a robot cannot accurately perceive the actions of another robot, an embedded simulation may provide an alternative to develop coordinated and cooperative robotic behaviours in real time.

References

[1] G. Baele, N. Bredeche, E. Haasdijk, S. Maere, N. Michiels, Y. Van de Peer, T. Schmickl, C. Schwarzer, and R. Thenius. Open-ended on-board evolutionary robotics for robot swarms. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 1123 –1130, May 2009.

[2] Theodore C. Belding. The distributed genetic algorithm revisited. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 114–121, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[3] J Bongard. Exploiting multiple robots to accelerate self-modeling. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 214–221, New York, NY, USA, 2007. ACM.

[4] J. C. Bongard. Accelerating self-modeling in cooperative robot teams. *IEEE Transactions on Evolutionary Computation*, 13(2):321–332, 2009.

[5] S. Elfwing, E. Uchibe, K. Doya, and H.I. Christensen. Biologically inspired embodied evolution of survival. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2210 – 2216 Vol. 3, sept. 2005.

[6] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life: Proc. 3rd European Conference on Artificial Life*, pages 704–720. Springer-Verlag, 1995.

[7] L. König, K. Jebens, S. Kernbach, and P. Levi. Stability of on-line and on-board evolving of adaptive collective behavior. In Herman Bruyninckx, Libor Preucil, and Miroslav Kulich, editors, *European Robotics Symposium 2008*, volume 44 of *Springer Tracts in Advanced Robotics*, pages 293–302. Springer Berlin / Heidelberg, 2008.

[8] Wenguo Liu and Alan F.T. Winfield. Open-hardware e-puck linux extension board for experimental swarm robotics research. *Microprocessors and Microsystems*, 35(1):60 – 67, 2011.

[9] Bonani M. Raemy X. Pugh J. Cianci C. Klaptocz A. Magnenat S. Zufferey J.-C. Floreano D. Mondada, F. and A. Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, 2009.

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

[11] Erol Sahin. Swarm robotics: From sources of inspiration to domains of application. *Swarm Robotics*, pages 10–20, 2005.

[12] Valerio Sperati, Vito Trianni, and Stefano Nolfi. Evolving coordinated group behaviours throughmaximisation of mean mutual information. *Swarm Intelligence*, 2:73–95, 2008. 10.1007/s11721-008-0017-1.

[13] Yukiya Usui Takaya and Takaya Arita. Situated and embodied evolution in collective evolutionary robotics. In *In Proc. of the 8th International Symposium on Artificial Life and Robotics*, pages 212–215, 2003.

[14] V. Trianni, S. Nolfi, and M. Dorigo. Cooperative hole-avoidance in a swarm-bot. *Robotics and Autonomous Systems*, 54 (2):97–103, 2006.

[15] Vito Trianni. *On the Evolution of Self-Organising Behaviours in a Swarm of Autonomous Robots*. PhD thesis, Faculty of Applied Sciences, Universite Libre de Bruxelles, Brussels, Belgium, 2006.

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