Evolving an Integrated Phototaxis and Hole-avoidance Behavior for a Swarm-bot

Anders Lyhne Christensen and Marco Dorigo
IRIDIA, Université Libre de Bruxelles, Belgium
alyhne@iridia.ulb.ac.be, mdorigo@ulb.ac.be

Abstract

This article is on the subject of evolving neural network controllers for cooperative, mobile robots. We evolve controllers for combined hole-avoidance and phototaxis in a group of physically connected, autonomous robots called *s-bots*, each with limited sensing capabilities. We take a systematic approach to finding a suitable fitness function, an appropriate neural network structure, and we explore and compare three evolutionary algorithms commonly used in evolutionary robotics: genetic algorithms, (μ, λ) evolutionary strategies, and cooperative coevolutionary genetic algorithms for optimizing weights in neural robot controllers. Finally, we show that solutions evolved in our software simulator can be transferred successfully to real robots.

Introduction

Robotics, evolutionary computation, and neural networks are each well-established research fields in their own respect. "Evolutionary robotics" is the name frequently used for research in the hybrid field combining the three. One of the goals in this field is to use evolutionary algorithms to evolve robot controllers based on artificial neural networks. A major advantage of this approach is that artificial evolution can find solutions that a human developer might not have considered or predicted. When a robot is situated in an environment, solutions found by evolution can exploit features in the environment *as they are perceived through the robot's sensors*, whereas a human developer will be limited to some degree by his or her understanding of the task (Nolfi and Floreano, 2000). Therefore, an evolutionary approach can potentially find simpler, more robust, generic, and scalable solutions to novel tasks in comparison with hand-written controllers.

Although promising, evolutionary methods have to our knowledge only been successfully applied to relatively simple tasks and are not yet used extensively in industry as a tool for synthesizing robot controllers. This is most likely explained by the fact that reaching the point where artificial evolution produces a controller that solves a given task is a difficult, tedious and time-consuming process, which involves a large amount of trial-and-error. However, if evolutionary robotics is ever to be used in more complex scenarios, we need to improve our understanding and methods for designing suitable evolutionary setups. With this in mind, we discuss and apply a structured approach for evolving robot controllers for the phototaxis and hole-avoidance task described below.

The task we are concerned with is the evolution of controllers for a number of robots called *s-bots*. An *s-bot* has a variety of sensors and actuators including a gripper, which enables multiple robots to physically connect and form an artifact called a *swarm-bot*. In *swarm-bot* formation each *s-bot* maintains autonomous control. Our objective is to obtain controllers for a number of *s-bots* in *swarm-bot* formation to allow them to safely navigate through an arena containing holes.

Cooperative navigation for multiple autonomous robots has previously been studied. In (Wang, 1991) strategies for movement in formation based on nearest neighbor tracking were developed. Collision avoidance for a group of robots in environments with moving objects was studied in (Arai et al., 1989). In both cases, controllers relied on formal and hand-coded strategies for path planning and conflict resolution and all experiments were conducted in simulation. In a more recent study, Quinn et al. showed that artificial evolution is a useful tool for automating the design of controllers capable of teamwork and role-allocation on real robots (Quinn et al., 2002).

With the development of the *s-bot* hardware platform, we have been able to study cooperation, navigation, and artificial evolution of controllers at a different level, namely for groups of *physically connected* robots. In the context of the SWARM-BOTS project there is a body of research on evolving artificial neural network controllers, see for instance (Dorigo et al., 2004) on evolving self-organizing behaviors for a *swarm-bot*. Moreover, specific studies on evolving controllers for coordinated-motion and hole-avoidance have been performed, see for instance (Trianni et al., 2006). The hole-avoidance studies performed to date have only included coordinated movement and not motion towards a predefined target such as a light source. However, other studies, such as (Groß and Dorigo, 2004), have been concerned with col-

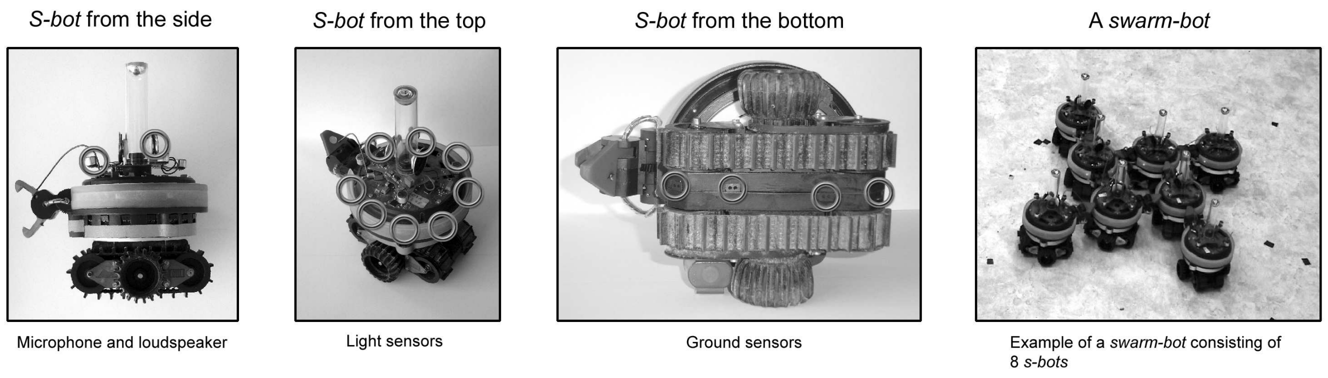

Figure 1: Different views of an *s-bot* highlighting the location of the sensors used and a *swarm-bot*. An *s-bot* has a diameter of 120 mm and a height of 190 mm.

lective transport of an object towards a light source, albeit in obstacle-free environments. To the best of our knowledge, combined phototaxis and hole-avoidance for a swarm of robots has not been studied prior to the work presented here.

This paper is organized as follows: In the next section we present our robot platform and the experimental setup. We then outline and discuss a structured approach to finding a suitable evolutionary setup. We follow this approach by first engineering a fitness function. Then we focus on the neural network structure for the robot controllers and test three evolutionary algorithms. In the final section we describe initial tests performed on real robots.

Robot Hardware and Experimental Setup

An *s-bot* and a *swarm-bot* are shown in Fig. 1. Each *s-bot* is equipped with four infra-red ground sensors, mounted between its differential treels (TRacks and whEELS), one pointing 45 degrees forward, two pointing straight downward, and one pointing 45 degrees backward. Microphones and speakers allow *s-bots* to emit and perceive sounds. An *s-bot* can sense forces acting upon it in the horizontal plane via traction sensors. These forces allow the *s-bot* to gauge the direction of motion of the swarm. Thus each *s-bot* can align its own direction of motion to that of the *swarm-bot*, allowing the *swarm-bot* to move coordinately. The traction sensors are mounted inside the robot between the bottom part (the chassis) and the top part (the turret). The turret can rotate independently with respect to the chassis: up to 180 degrees in each direction from the neutral position. The result of an action in a given situation is likely to depend on the current rotational difference between the top and bottom part of the *s-bot*. We therefore use two sensors that read the rotational difference in the clockwise and counterclockwise directions, respectively, at every control step. The relative direction of the target, identified by a light source, is perceived via 8 light-sensors distributed evenly around the plastic ring on the chassis of the *s-bot* as shown in Fig. 1.

The ground sensors are located directly under the *s-bot*, which means that the *s-bot* will only detect the presence of a hole once it is already partly over it. If a single robot tries to navigate through the arena shown in Fig. 2, it is very likely to fall into a hole unless it approaches the hole perpendicularly. In *swarm-bot* formation, however, the *s-bots* should be able to cooperate to safely navigate through the arena and reach the location of the light source.

We have preprogrammed the *s-bots* to emit a sound, which can be perceived by the other *s-bots* in the *swarm-bot*, when the presence of a hole is detected. This has previously been found to be an efficient aid when evolving hole-avoidance for a *swarm-bot* (Trianni et al., 2006).

Methodology

We have taken a structured approach to determining an *evolutionary setup*, which produces controllers capable of solving the combined hole-avoidance and phototaxis task. By the term “evolutionary setup” we mean the following four components: (i) the fitness function, (ii) the artificial neural network type and structure (unless that is under evolutionary control), (iii) the evolutionary algorithm, and (iv) the parameters associated with the evolutionary algorithm.

It is not customary for authors in the field of evolutionary robotics to disclose how the chosen evolutionary setup was found. We believe that many evolutionary setups are found in an ad-hoc fashion. In this study we take on the challenge of finding a good evolutionary setup in a structured manner. Our objective is two-fold, namely to save time by reducing the amount of manual trial-and-error necessary and to be explicit about the various setups tested and their results. It is reasonable to assume that both issues are important for the use of evolutionary robotics in both academia and industry.

It is practically infeasible to perform an exhaustive search in the space of evolutionary setups, that is, all possible combinations of parameter values, fitness functions, neural network types, and so on. For evolutionary parameters such as the population size, mutation rate, and number of crossovers, we can discretize the intervals of values that we believe to be reasonable based on experience and/or a few initial experiments, then run a number of tests, and finally choose the combination of values yielding the best results. For other components of the evolutionary setup, for instance the fitness function, such a discretization is not obvious.

Searching for a good evolutionary setup is often done in an ad-hoc manner: an initial evolutionary setup is chosen based on experience or a more or less qualified guess and then modified until the results are satisfactory. We propose instead to consider each component of the evolutionary setup one at a time. In this way, we can focus on what we assume to be the more predominant components initially and then gradually move towards less predominant components. The most important aspects of an evolutionary setup are arguably those that require the largest amount of human intuition, are hardest to find in a brute force manner, and determine if a suitable controller can be evolved or not (as opposed to how fast an acceptable solution is found).

The component topping this list is the fitness function. The fitness function represents the element for which our understanding of the problem plays the largest role. If everything else is perfect, but the fitness function is wrong, we are highly unlikely to obtain a satisfactory behavior. The same is not necessarily true for the other aspects of the evolutionary setup; if the mutation rate, for instance, is too low, it could take longer than necessary to evolve the desired behavior. Likewise, if an artificial neural network has too many or too few nodes, it might overfit or underfit the problem, but still an approximation of the behavior we had in mind can be evolved.

An important issue is that in the context of evolutionary robotics we often cannot determine if a fitness function exerts the correct evolutionary pressure *unless we test it*. However, we cannot test it before we have chosen a neural network type and structure and unless we try to evolve controllers under an evolutionary algorithm, which has all of its parameters set to some value. This chicken-and-egg issue can only be overcome by testing a fitness function with a range of different neural network structures and an evolutionary algorithm. These might of course not be optimal, but they should be relatively *conservative*, in the sense that if a solution can be found, the evolutionary algorithm has a good chance of finding it (although it might take several evaluations more than strictly necessary). This means that every fitness function is tested with a conservative and representative selection of neural network structures and evolutionary parameters. We call the set of representative elements the *baseline set* for a given component.

After a suitable fitness function has been found, we shift our focus to the type and structure of the artificial neural network, then to the evolutionary algorithm, and finally to the parameters of this algorithm. All results presented below have been obtained by running artificial evolutions in the `TwoDee` simulator (Christensen, 2005).

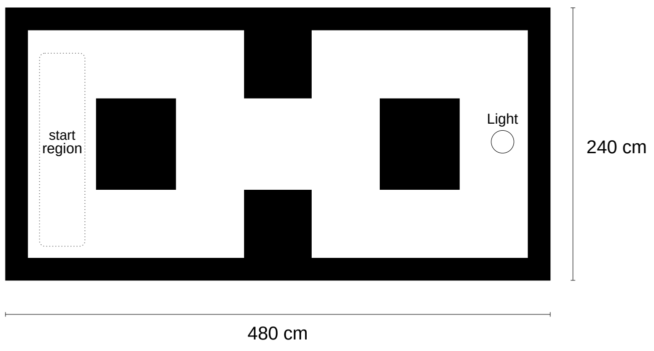

Figure 2: An example of an arena. The dark areas denote holes, while the white patches denote the arena surface on which the robots can move. The *swarm-bot* must move from the initial location shown on the left-hand side to the light source on the right without falling into any of the holes or over the edge of the arena.

Fitness Function Engineering

We engineered the fitness function by testing it in evolutionary setups containing the elements of the *baseline sets* for the neural network structures and the evolutionary algorithms. We made it progressively more complex as we found it necessary. At each stage another component of the fitness function was added.

The first component scores controllers depending on how close they manage to get to the light source. In case they manage to reach the light source[1] they are scored based on how fast they do so:

$$f_{light} = \begin{cases} 1 - \dfrac{\textit{min distance}}{\textit{initial distance}} & \text{if light is not reached,} \\ 2 - \dfrac{\textit{time light is reached}}{\textit{total time}} & \text{if the light is reached.} \end{cases}$$

Notice that if a *swarm-bot* falls into a hole, the trial is stopped and the controller's fitness is computed. However, using this fitness function only resulted in *swarm-bots* frequently falling into holes close to the light source. Therefore, another component was added, penalizing controllers for falling into a hole:

$$f_{stayalive} = \begin{cases} 0.5 & \text{if the } \textit{swarm-bot} \text{ falls into a hole,} \\ 1.0 & \text{otherwise.} \end{cases}$$

Previous studies have shown that coordinated motion can be obtained by minimizing the traction between the *s-bots*

[1]If the *swarm-bot* gets within 50 cm of the light source, we consider it as having reached the light source.

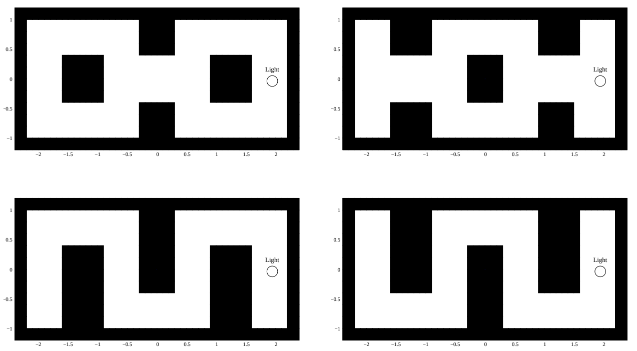

Figure 3: The four arenas used to evolve controllers. *Swarm-bots* start in the left-hand side of the arenas.

in a *swarm-bot* (Trianni et al., 2004). It is reasonable to assume that coordinated motion is a prerequisite for performing hole-avoidance and phototaxis successfully. Therefore, we added a component that rewards controllers which minimize the traction between the *s-bots*. The traction forces are measured in the two dimensions of the horizontal plane with 0 corresponding to no traction perceived and 1 to the maximum traction force perceivable. At each control step *i*, we record the maximum traction τ_i^{max} perceived by any of the *s-bots* in the simulation:

$$f_{minimizetraction} = \frac{\sum(1-\tau_i^{max})}{total\ number\ of\ control\ steps}$$

The three components listed above are multiplied to obtain the fitness score used by the evolutionary algorithm for selection and reproduction.

In order to avoid specialization, we evaluate each genotype in *swarm-bots* of different sizes and in multiple arenas. We use the four arenas shown in Fig. 3 and rectangular *swarm-bot* formations consisting of 2x2, 3x2 and 4x4 robots. Controllers are evaluated in each of the arenas twice and their overall fitness score is the average of the eight scores obtained.

Neural Network

The neural controller has 19 inputs (8 light sensors, 4 ground sensors, 4 traction sensors, 2 rotation sensors, and one sound sensor) and 2 outputs (one for each treel). Our aim is to find a simple neural network capable of solving the task, since simpler networks require less computational resources, which are often limited on physical robots. Moreover, simpler networks result in search spaces of lower dimensionality and we assume that the lower the dimensionality of the search space the faster an evolutionary algorithm will converge to a good solution if such a solution exists.

To test the performance of different neural network types and structures, we ran 20 evolutions of 1000 generations for each of the following neural networks: a perceptron and three multi-layered feed-forward networks with one hidden layer of 2, 10 and 19 nodes, respectively.

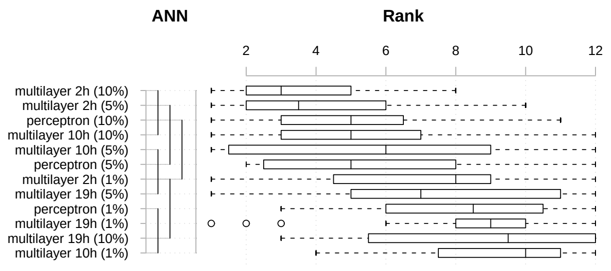

Figure 4: Box-plots for all neural network structures evolved with different mutation rates ordered according to their average ranking obtained in the post-evaluation phase. Each box comprises observations ranging from the first to the third quartile. The median is indicated by a bar, dividing the box into the upper and lower part. The whiskers extend to the farthest data points that are within 1.5 times the interquartile range. Outliers are shown as circles. The solid vertical lines on the left-hand side indicate lack of statistical significance at confidence level 95%. The results show that the multi-layer network with 2 hidden nodes and weights evolved using a genetic algorithm with a mutation rate of 10% received the highest average ranking. Statistically, however, the results for this setup are not significantly different from the next three evolutionary setups listed, including a perceptron whose weights were evolved with a mutation rate of 10%.

For each of the neural network structures we tested evolutionary setups with three different mutation rates: 1%, 5%, and 10%, on genotypes consisting of one floating-point value for each weight in the neural network.

In order to compare the performance of the controllers evolved in the different evolutionary runs, we took the highest scoring controller of the last generation, post-evaluated it 25 times in each of the four arenas shown in Fig. 3, and recorded the average fitness score.

We use Friedman's test to compute statistical significance of our results and to determine which neural network structure yields the best performance (Conover, 1999). Friedman's test is convenient to use because it is non-parametric and therefore no strong assumptions need to be made about the distributions underlying the phenomenon of interest. The fitness scores obtained during the different post-evaluation runs are mutually independent and we can rank them according to their numerical value, which are the only requirements for Friedman's test to be applicable.

A box-plot of the results is shown in Fig. 4. The results have been ordered according to the average ranking of the evolutionary setups and the lack of statistical significance between them is indicated by the solid vertical lines on the left-hand side of the figure (i.e. those evolutionary setups covered by the same vertical line are not significantly different). The results shown in the figure suggest that the simpler networks outperform the more complex networks.

Based on the results shown in Fig. 4, a perceptron seems to perform well in comparison with the other networks whilst also being the simplest structure we tested. The best performing networks were the multi-layered neural network with 2 hidden neurons and the perceptrons. Statistically, there is no significant difference between the performances of these networks. In order to choose which network to proceed with, we analyze the nature of the solutions found.

The solutions can be divided into three strategy types: *fail* strategies, *reverse* strategies, and *turn* strategies. While *fail* strategies fail to solve the task, both *reverse* and *turn* strategies solve the task, but in different ways. Controllers displaying a *reverse* strategy invert the direction of motion once a hole is detected and then reapproach the hole at a slightly different angle, see Fig. 5a. Controllers displaying a *turn* strategy do not reverse their direction of motion, but instead they keep turning away from the hole until it is no longer perceived, see Fig. 5b.

An interesting result is that none of the evolutionary runs produced controllers that move *swarm-bots* directly towards the light source as one might expect given that phototaxis should be performed. The *swarm-bots* instead move left (or right) with respect to the direction of the light source and follow one of the sides of the arena and/or holes until the light source is reached. The reason for this behavior is likely that the evolutionary setup comprises evaluation of individuals in the four arenas shown in Fig. 3, where for instance the first arena in the bottom row requires the *swarm-bots* to move left around the first hole, then right around the second, and finally left again around the third hole before the light source is reached. A controller which moves *swarm-bots* directly towards the light source would have to decide whether to move left or right when a hole is encountered. In the arenas shown in the bottom row of Fig. 3 a wrong choice of direction could lead to the *swarm-bots* getting stuck in a corner, which would be difficult to escape given the reactive nature of the neural network controllers tested. Simpler "hole-following" strategies perform better since the light source can be reached in this way in all of the areas used during evolution.

We tested both *turn* strategies and *reverse* strategies on real robots and found that controllers based on the *turn* strategy transfer to the real robots better than controllers based on the *reverse* strategy. This is due to the fact that *reverse* strategies require control steps in all members of the *swarm-bot* to be highly synchronous, which is the case in simulation but not for real robots, for which control programs are not started at the exact same instant and clocks can drift. Fig. 6 shows the number of strategies from each group found by the different evolutionary setups.

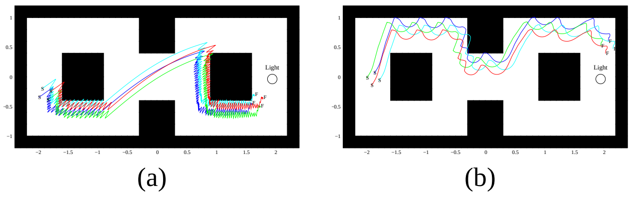

Figure 5: Examples of the two different strategies found by evolution: *reverse* (a), and *turn* (b).

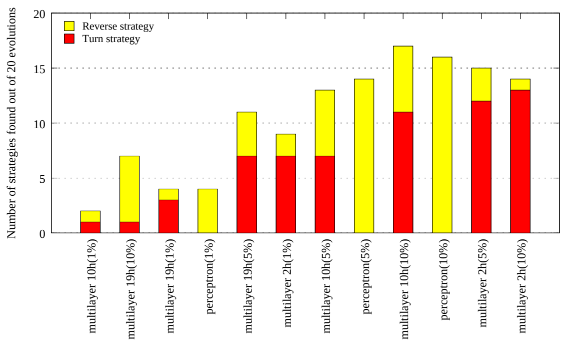

Figure 6: The number of *reverse* and *turn* strategies found by the different evolutionary setups.

In setups using a single layer perceptron, artificial evolution found only *reverse* strategies. On the other hand, a multi-layer network with 2 hidden nodes out-performed the other setups, both in terms of average fitness score obtained during post-evaluation and the number of transferable strategies found. We therefore choose this network structure.

Evolutionary Algorithm

The results presented above have all been obtained with a genetic algorithm (GA) with rank-based selection, mutation, and single-point crossover (Goldberg, 2002), (Mitchell, 1996). In this section we compare the results obtained using a GA with two other evolutionary algorithms often used in evolutionary robotics, namely the (μ, λ) evolutionary strategy ((μ, λ)-ES), (Schwefel, 1995), and a cooperative coevolutionary genetic algorithm (CCGA) (Potter and De Jong, 1994; Potter and De Jong, 2000). One of the main differences between a GA and the (μ, λ)-ES is that in the former crossover is used, while in the latter mutation is the only genetic operator used to perform the search. The CCGA differs from both the GA and the (μ, λ)-ES, because it employs multiple, isolated sub-populations. Each sub-population con-

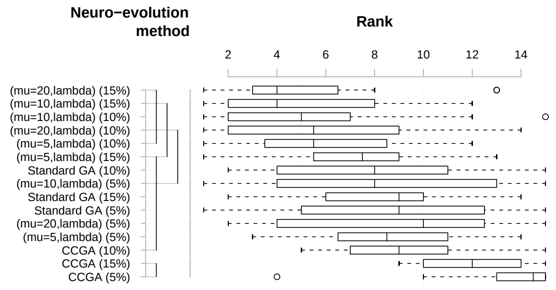

Figure 7: Box-plots of the ranking obtained during the post-evaluation phase for the GA, the (μ,λ)-ES, and the CCGA with various mutation rates. The results show that the (μ,λ)-ES achieves the highest average ranking, and that several evolutionary setups involving the (μ,λ)-ES perform better than the GA and the CCGA.

tains a component of the final solution. Applied to neural networks, a sub-population contains neurons for a specific position in the final network. A neuron, in this case, is described by a set of weights for its incoming connections. When a neuron in a sub-population is evaluated, a complete network is constructed using the neuron itself and neurons from the other sub-populations. During evolution the sub-populations should *co-evolve* by specializing and adapting to each other. This can result in better solutions being obtained faster (Potter and De Jong, 2000). However, our results, presented below, indicate that this is not always the case.

In order to compare the performance of GAs, (μ,λ)-ESs, and CCGAs on our task, we have chosen a subset of evolutionary parameters for each method and conducted 20 evolutionary runs for each of the resulting setups. The approach is similar to the approach taken for finding an appropriate neural network structure described above.

For all three evolutionary algorithms we use the fitness function described earlier. The controller consists of a multi-layer feed-forward network with 2 hidden nodes. We tested setups with three different mutation rates, namely: 5%, 10% and 15%, and for the (μ,λ)-ES we tested three different values for the number of parents selected for reproduction, μ, in each generation: 5, 10 and 20. For the CCGA we use 4 sub-populations (one for each of the hidden nodes, and one for each of the two output nodes) of 25 individuals each. For both the GA and the (μ,λ)-ES, population sizes of 100 individuals were used.

Box-plots of the post-evaluation results for the best controllers obtained in the different evolutionary setups are shown in Fig. 7. The most successful evolutionary setups are those involving the (μ,λ) ES, followed by the GA. The CCGA performs poorly for all three mutation rates tested.

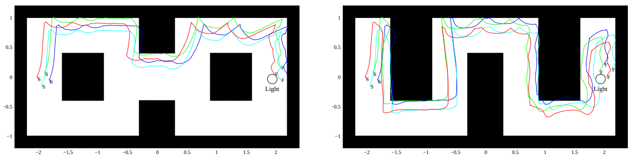

Figure 8: Example of how a controller capable of solving the task behaves in two of the arenas using a simple, but nonetheless general and successful, hole-following strategy.

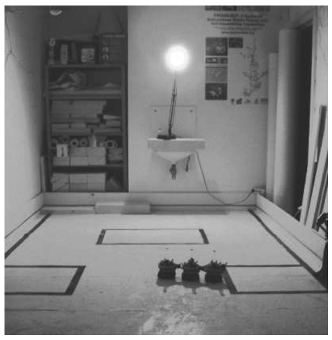

Figure 9: A photo of the arena for the real robots where holes are replaced by black duct tape. The arena is a replica of the simulated arena shown on the left-hand side in Fig. 8.

The final choice is therefore to use an evolutionary setup in which the weights in a multi-layered neural network with two hidden neurons are optimized by a (μ,λ)-ES, with $\mu =$ 20 and a mutation rate of 15%. Fig. 8 shows an example of a successful phototaxis and hole-avoidance behavior evolved in this evolutionary setup.

Validation on Real Robots

We ported the best controllers evolved in simulation to real *s-bots* and conducted initial experiments with a *swarm-bot* consisting of three robots. The *swarm-bot* was tested in the arena shown in Fig. 9, which has the layout and measures shown in Fig. 2.

Two main differences between simulation and the real world were introduced: First, we reduced the maximum speed of the real robots compared with the speed used for the simulated robots in order to compensate for the differences in the types of noise, sensor delays, and so on, between simulation and reality. By reducing speed, noise was smoothed and the impact of each action (or control cycle) was decreased. Second, black duct tape was used instead of holes. A black surface and holes are perceived in the same manner by the *s-bots*' ground sensors, while using duct tape instead of holes prevents physical damage to the robots in case they should "fall in". The interaction of a *swarm-bot* with an arena containing holes differs from the interaction with an arena in which holes are replaced by black duct tape. In an arena with real holes, an *s-bot* which is over a hole cannot directly influence the rest of the *swarm-bot* by moving its treels. If duct tape is used instead of holes, an *s-bot* still has surface contact while moving on the duct tape. The motion of its treels therefore has some influence on the *swarm-bot*.

Initial tests on the real robots showed that despite these differences the evolved controllers are capable of performing integrated phototaxis and hole-avoidance using a *turn* strategy similar to what was observed in simulation.

Conclusion

In this study we discussed and applied a structured approach to finding a suitable evolutionary setup. We obtained a fitness function that exerts the correct evolutionary pressure. Moreover, we showed that a multi-layer feed-forward network with a layer of 2 hidden nodes was the simplest network structure tested for which evolution found transferable controllers. During the evaluation of evolutionary algorithms it was found that evolutionary setups involving (μ,λ)-ESs outperformed the setups with GAs and CCGAs. Finally, the controllers evolved in our software simulator do perform integrated phototaxis and hole-avoidance and they can be successfully transferred to real robots.

Acknowledgements

Anders Christensen acknowledges support from COMP2SYS, a Marie Curie Early Stage Research Training Site funded by the European Community's Sixth Framework Programme (grant MEST-CT-2004-505079). The information provided is the sole responsibility of the authors and does not reflect the European Commission's opinion. The European Commission is not responsible for any use that might be made of data appearing in this publication. Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Research Director, and from the "ANTS" project, an "Action de Recherche Concertée" funded by the Scientific Research Directorate of the French Community of Belgium.

References

Arai, T., Ogata, H., and Suzuki, T. (1989). Collision avoidance among multiple robots using virtual impedance. In *Proceedings of IEEE/RJS International Workshop on Intelligent Robots and Systems (IROS) '89*, pages 479–485. IEEE Computer Society Press, Los Alamitos, CA.

Christensen, A. L. (2005). Efficient neuro-evolution of hole-avoidance and phototaxis for a swarm-bot. Technical Report TR/IRIDIA/2005-14, IRIDIA, Université Libre de Bruxelles, Belgium. DEA Thesis.

Conover, W. J. (1999). *Practical Nonparametric Statistics*. Wiley & Sons, New York, 3rd edition.

Dorigo, M., Trianni, V., Şahin, E., Groß, R., Labella, T. H., Baldassarre, G., Nolfi, S., Deneubourg, J.-L., Mondada, F., Floreano, D., and Gambardella, L. M. (2004). Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, 17(2–3):223–245.

Goldberg, D. E. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Boston, MA.

Groß, R. and Dorigo, M. (2004). Group transport of an object to a target that only some group members may sense. In *Parallel Problem Solving from Nature – 8th Int. Conf. (PPSN VIII)*, volume 3242 of *LNCS*, pages 852–861. Springer Verlag, Berlin, Germany.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA.

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

Potter, M. A. and De Jong, K. (1994). A cooperative coevolutionary approach to function optimization. In *Proceeding of the Third Conference on Parallel Problem Solving from Nature – PPSN III*, volume 866 of *LNCS*, pages 249–257, Springer Verlag, Berlin, Germany.

Potter, M. A. and De Jong, K. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29.

Quinn, M., Smith, L., Mayley, G., and Husbands, P. (2002). Evolving teamwork and role allocation for real robots. In *Proceedings of 8th International Conference on Artificial Life*, pages 302–311. MIT Press, Cambridge, MA.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. Wiley & Sons, New York.

Trianni, V., Labella, T. H., and Dorigo, M. (2004). Evolution of direct communication for a swarm-bot performing hole avoidance. In *Ant Colony Optimization and Swarm Intelligence – Proc. of ANTS 2004 – 4th Int. Workshop*, volume 3172 of *LNCS*, pages 131–142. Springer Verlag, Berlin, Germany.

Trianni, V., Tuci, E., and Dorigo, M. (2006). Cooperative hole avoidance in a swarm-bot. *Robotics and Autonomous Systems*, in press.

Wang, P. K. C. (1991). Navigation strategies for multiple autonomous mobile robots moving in formation. *Journal of Robotics Systems*, 8(2):177–195.