Evolutionary swarm robotics
– Summary –

Frank Lange
Phillipp Schoppmann

June 28th 2013

Abstract

Evolutionary algorithms have proven to be useful optimization techniques in many different fields. This paper will describe the application of evolution to swarm robotics. It is based on research done by Dorigo et al. in 2004. [DTS+04]

Contents

Figure 1: A swarm of six S-Bots, five of them connected

1 Overview

Swarm robotics focus on the coordination of multiple robots trying to reach a certain goal. In most scenarios, this involves limited communication abilities between robots and the restriction to local information. Although control is decentralized, as each robot is autonomous, emergent behavior can be observed, enabling the swarm to solve problems that a single robot can't.

1.1 The Swarm Bot

Here, the relevant artifact is called *Swarm Bot*. It consists of several mobile *S-Bots*, who can connect to and disconnect from each other. The Swarm Bot can be used for various tasks, including moving objects and moving through tough physical terrain. Furthermore, due to the ability to disconnect one or more S-Bots from itself, it can use them for tasks like finding a goal or tracing an optimal path, where single S-Bots may be more efficient.
In this paper we will focus on two tasks: *Aggregation* and *coordinated motion*. Aggregation describes the ability of several scattered S-Bots to minimize the distance between each other. Coordinated motion represents the ability of a connected Swarm Bot to move in a single direction. Both tasks are considered mandatory for the development of more complex behavior.

1.2 Self organization

As mentioned before, in most scenarios swarm robots are limited to local information. Therefore, it is important to apply mechanisms that can deal with this restriction while still being able to show emergent behavior.
Self organization describes such mechanisms for biological systems, but can also be applied to swarm robotics. The System state depends on positive and negative feedback:
Positive feedback represents the amplification of a certain property emerging from random interactions. It increases exponentially over time.
Negative feedback works as a regulation which is triggered by positive feedback exhausting some resource. By the interaction of positive and negative feedback, the system is kept in a stable state. [Cam03, pp. 11–20]
The difficulty in creating self organizing Systems is the fact, that the relation between individual and global behavior is hard to analyze: Given a set of individual behaviors, it is difficult to predict the behavior that is going to emerge on a system level. On the other hand, given global behavior, it is difficult to decompose individual behaviors leading to it.

1.3 Artificial evolution

A possible solution for this problem is *artificial evolution*. It can be used to generate individual behavior, while evaluating the system as a whole and therefore bypassing the derivation of individual rules from global behavior. For the Swarm Bot, we will use artificial evolution to create controllers for each S-Bot. The controllers are realized through artificial neural networks, connecting sensory input to motor output. A genetic algorithm then determines optimal weight matrices for these networks.
Artificial evolution is based on the biological equivalent. It optimizes individuals on the basis of a given *fitness function*, using *selection*, *reproduction* and *mutation*. [ES03, pp. 37–69]
The general process of a genetic algorithm is as follows:

1. An initial population is created randomly
2. The fitness of all individuals in the current population is evaluated by applying the fitness function
3. A subset of well-fit individuals is selected for reproduction
4. Offspring is created and altered using mutation
5. Steps 2–4 are repeated until a certain stop criterion (e.g. fitness threshold, timeout) is met

Note that recombination is not used here, as it doesn't perform well for evolving weight matrices of neural networks. [Yao99, pp. 1425–1428]

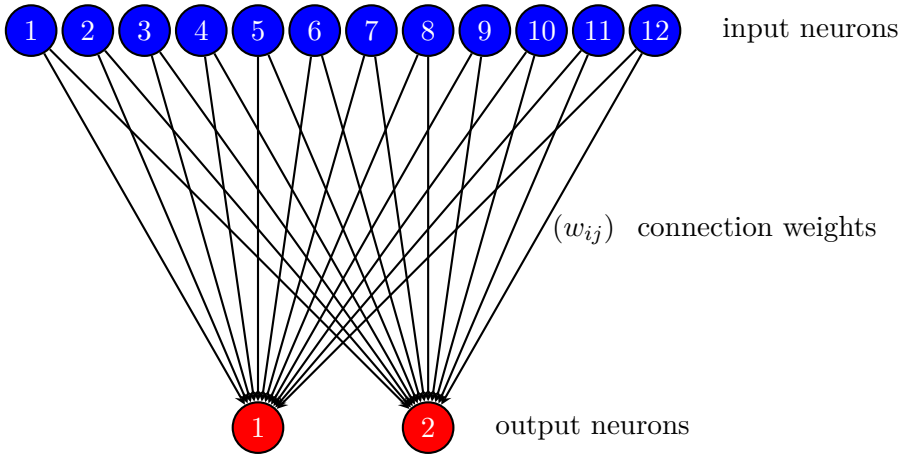


Figure 2: A single layer perceptron with 12 input neurons and two output neurons

1.4 Artificial neural networks

As mentioned above, an artificial neural network is used to connect sensory input to motor output. Like genetic Algorithms, artificial neural networks are inspired by biology. There is a vast variety of different types of neural networks. Here, the relevant network architecture is the *single layer perceptron*.
The Neurons are arranged in two layers, one input and one output layer. Each input neuron is connected to each output neuron. All neural connections are *weighted*. A neuron's weighted input is mapped to its output using a given *activation function*. Here, a *sigmoid function* will be used:

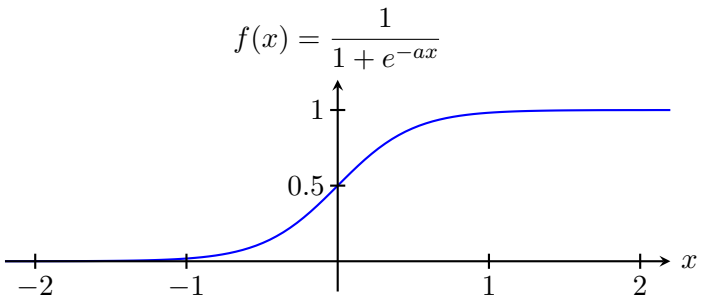


Figure 3: The sigmoid function used in the experiment

The functionality of the network depends on the following parameters:

1. Network topology, i.e. number of layers and neurons per layer
2. The activation function
3. Connection weights.

In the following experiment, the S-Bot controllers only differ in (3).

2 Experiment

In the following section the experimental procedure is described. First we introduce the general experimental enviroment and then continue to describe the simulation detailes used in the aggregation and coordinated movement experiments.

2.1 Experimental setup

As a general approach to the overall experiment a five step process was used that can be roughly described as follows:
First the real robot hardware needs to be defined. After that a simulator is developed that allows different detail levels for the agent models representing the robots. Then a simplified model is used as a starting point to construct experiments to evolve controllers for the sbots in a reasonable amount of time. Successfully evolved controllers using the simplified model are then improved using an increased level of detail. Finally, the results of the experiments featuring the highest level of detail are then ported to the real hardware.

3 Aggregation

For the aggregation experiments a very simplified model of the sbots was used. Nearly every feature of the sbot was omitted, leaving only 2 wheels, 8 proximity sensors and 3 sound sensors.

3.1 Simulation details

The simulation environment consisted of a flat, obstacle free area covering an equivalent of 3x3 meters. In this area the sbots could move freely, permantly emitting a sound signal that could be recognized within an approximate radius of 75cm. Noise was simulated by a uniformly distributed random signal within 5% of the senders saturation value.

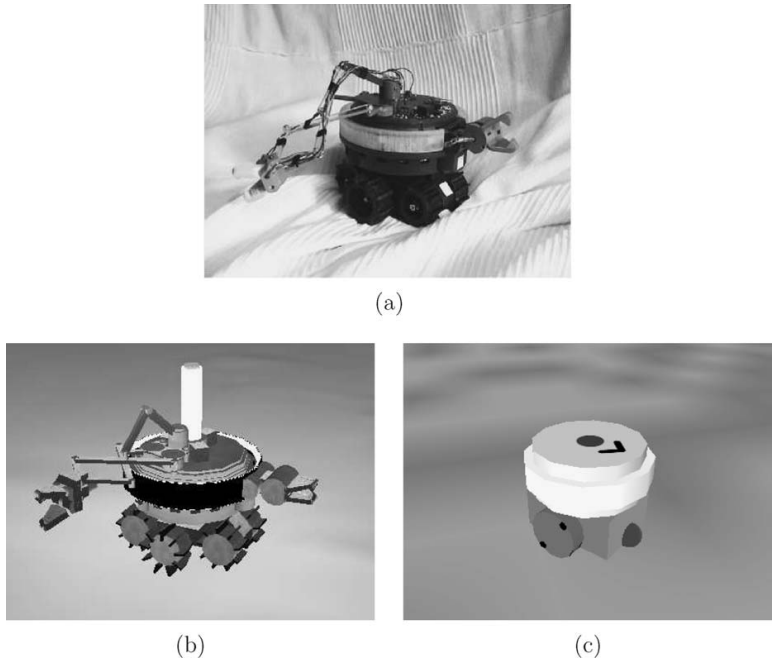


Figure 4: a) Real hardware sbot b) Detailed simulation model c) Simplified simulation model

As described earlier a neural network was used to represent a successful mapping of the data received by the 3 sound and 8 proximity sensors to the motors of the 2 wheels of each sbot. Therefore the neural network consisted of 12 input neurons (including a bias input) representing the input sensors and which where each directly connected to the 2 output neurons representing the motor outputs. Therefore the neural network contained 24 connections, with each connection beeing weighted within [-10, +10] and represented by a 8-bit bitstring. This results in a (12x2) x 8 = 192 bitstring representing a full controller which is called the *genotype*.

3.2 Evolutionary algorithm

The evolutionary algorithm starts with a random population of 100 genotypes, each containing all the needed connection weights of the neural network and therefore representing a full controller for the sbots.
These 100 random genotypes represent the first generation of an evolutionary run, which in this experiment stretches over 100 generations. To obtain a new generation of genotypes, each of the genotypes of the current generation needs to be tested in the simulation environment described above.

To evaluate the aggregation performance of a single genotype, it is tested over 8 *epochs* where the final performance value is the average of these

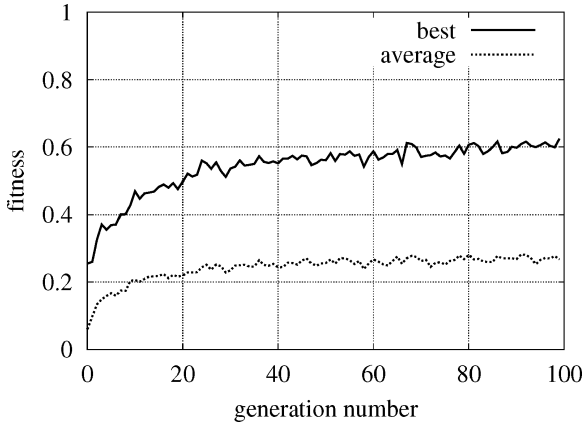


Figure 5: Aggregation performace averaged over 20 evolutionary runs

8 epochs. An epoch represents an experiment where a random number of sbots between 4-8 is spawned in the simulation environment described above, each configured with the current genotype to be tested. Each epoch lasts 900 simulation cicles and each simulation cicle represents 100 ms of real time.
The aggregation quality of a genotype for one epoche is measured by 2 performance critea: how much did the sbots move to their center of mass and how much did they move straight forward in doing so. Both criteria are measured for each simulation cicle by averaging over the number of sbots used in the specific epoch and are then multiplied. The final fitness of an epoch is then achieved by averaging over the whole epoch duration. Again, this is repeated 8 times for each of the 100 genotypes of the current generation.

After the evaluation of the entire generation the 20 best genotypes are then selected to produce 5 offsprings each, by randomly flipping each bit with a 3% chance.
After 100 generations the evolution stops, thus calling this an evolutionary run. In the aggregation experiment 20 of those evolutionary runs where done and the averaged performance of those 20 runs can be seen in Figure 2 where it is important to keep in mind, that the performance is the product of both, the gathering and the straight forward movement quality of the controllers.

To evaluate the scalibility of the evolved controllers each of those 20 controllers was then again tested against different groupsizes in the pattern 4, 8, 12, ..., 40 with 100 testruns for each group size. The final results of that test can seen in Figure 3. This shows, that in theory, one can evolve controller in an limited environment using only limitied group sizes which then still deliver acceptable performance once that is limit doesn't apply

anymore.However, it is worthy to keep in mind, that the simulation area of 3x3 meters was not scaled to cope with an increasing group size, hence rendering it somewhat easy for larger groups to aggregate.

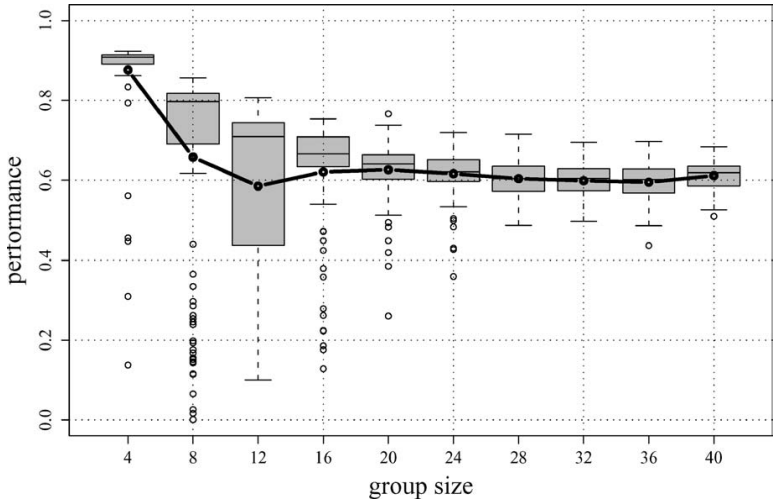


Figure 6: Aggregation performace when confronted with increasing group sizes

3.3 Observed behavior

Completing the aggregation experiment are observed behaviors that seemed to have been evolved in all evolutionary runs. These observations include that solitary sbots seem to roam the complete area within large circles, whereas smaller groups tend to develop a certain following behavior once aggregated, where each sbots revolves around the center of the group and the group as a whole moves around the area in small circles, always sticking together once aggregated. Largers groups however never seem to reach a comparable stable state. They slowly roam the area in larger circles, continuesly pickung up but also losing single sbots.

4 Coordinated Movement

In the coordinated movement experiment the simplified simulation model was used once more. However, the experiment is completely different compared to the aggregation experiment, since now, it was tested whether controller could be evolved that result in collective movement when a certain number of sbots already are aggregated and physically connected.

4.1 Simulation details

In this experiment the same general simulation setup as within the aggregation experiment was being used. Now however, the sbots are already connected via rigid links representing their grippers. Therefore the sound and proximity sensors from the aggregation experiment are being omitted and four traction sensors are being introduced.
Hence the neural network now only features 5 input neurons, representing the 4 traction sensors and a bias neuron, being directly mapped to the 2 motor output neurons. A connection weight is once again represented by a 8-bit bitstring mapping to the interval [-10, 10] but since there are less connections in the neural network a single genotype only consists of (5x2) x 8 = 80 bits.

Since the rigid link between the sbots does not prevent each sbot from having an orientation of it's own, the traction sensors are connected to the links and are measuring the angle and force of the traction received by each sbot. More precisely, via the traction sensors each sbot receives the average direction and force in which the group as a whole is trying to move and therefore can calculate the difference between the average direction of the group and it's own orientation and hence the severity of it's own misalignment compared to the majority of the group.

4.2 Evolutionary algorithm

The evolutionary algorithm did not change significantly, compared to the aggregation experiment. Once again it starts with 100 random genotypes as a first generation and evolves each next generation by evaluating each genotype for 5 epochs, where each epoch features a fixed group size of 4 sbots, including each sbots spawning with a different orientation. The fitness of a controller is measured at the end of each epoch and is the Euclidean distance between the group's starting position and the group's position at the end of the epoche. The final fitness of a controller is again the average of all 5 epochs, where each epoch concists of 150 simulation cicles.

One evolutionary run lasts 100 generation where in each generation the 20 best genotypes are selected to produce 5 offsprings by flipping each bit with a 3% chance. After 20 evolutionary runs, the best genotype of each run was selected and then tested again for increasing group sizes in the pattern 4, 8, 12, ..., 40. The results are shown in Figure 4.

4.3 Observed behavior

The behavior, that successful controllers seem to have evolved during the evolutionary run seems to be, that the traction intensity determines how

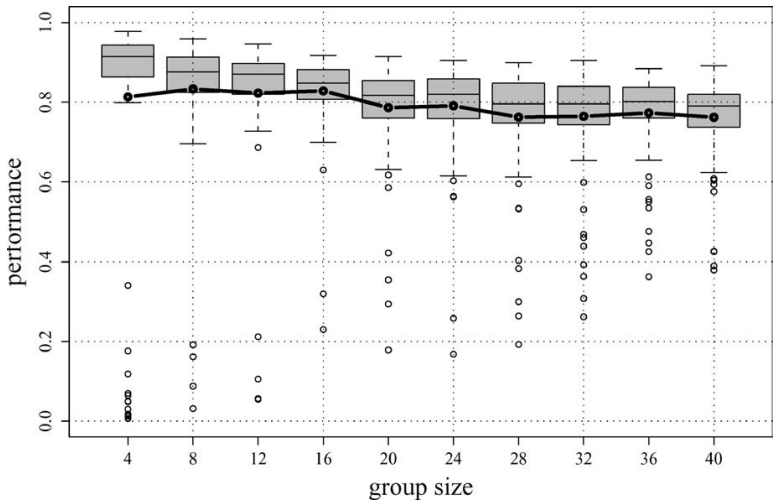
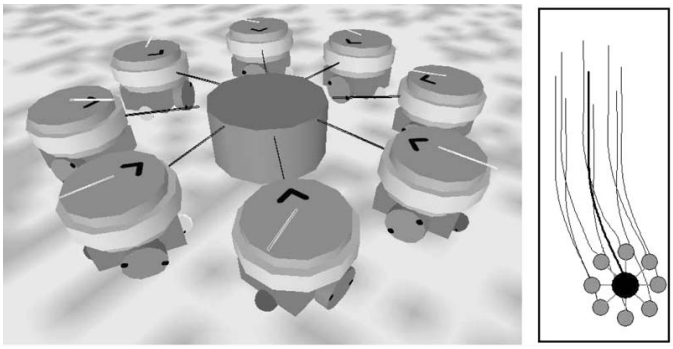


Figure 7: Coordinated movement performance for increasing group sizes

drastically a single sbots changes it's orientation. This means that for example if the sbots spawn with almost identical orientations, then the traction received by each sbot is near 0 and so each can continue in that exact direction with full speed. If however 3 sbots spawn with near identical orientations and only one sbots spawns with a highly misaligned orientation, then this sbot will receive an immense traction and will change it's orientation drastically, whereas the other sbots will only tune their orientation gracefully.

This results in 2 side effects which are object avoidance and object pulling behavior. When the group hits an object, the touching sbots receives the impact as a high traction in the opposite direction to it's orientation and the other sbots receive a similiar traction and therefore change their orientation rather drastically.

Object pulling is possible with more evolved controllers where a certain numbness has evolved in a positive fashion because it can be benefitial for the group if a single sbots continues in it's own direction even if it receives a slight traction in the opposite direction. Therefore, if the sbots are connected to an object, the pulling sbots receive the object as a slight traction in the opposite direction which to them is identical to an sbot still trying to match the groups direction. Luckely they ignore this traction hence resulting in them continuing their movement and pulling the object as can be seen in Figure 5.



(a) (b)

Figure 8: a) Sbots connected to an object b) Resulting object pulling behavior

References

[Cam03] Scott Camazine. *Self-organization in biological systems*. Princeton University Press, 2003.

[DTS+04] M. Dorigo, V. Trianni, E. Sahin, R. Groß, T.H. Labella, G. Baldassarre, S. Nolfi, J.-L. Deneubourg, F. Mondada, D. Floreano, and L.M. Gambardella. Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots 17*, pages 223–245, 2004.

[ES03] A.E. Eiben and J.E. Smith. Genetic algorithms. In *Introduction to Evolutionary Computing*, Natural Computing Series, pages 37–69. Springer Berlin Heidelberg, 2003.

[Yao99] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.