Evolutionary swarm robotics

Frank Lange
Phillipp Schoppmann

12. Juni 2013

Swarm robotics

- robots cooperating to reach a certain goal

Swarm robotics

- robots cooperating to reach a certain goal
- decentralization of control

Swarm robotics

- robots cooperating to reach a certain goal
- decentralization of control
- limited communication abilities

Swarm robotics

- robots cooperating to reach a certain goal
- decentralization of control
- limited communication abilities
- use of local information

Swarm robotics

- robots cooperating to reach a certain goal
- decentralization of control
- limited communication abilities
- use of local information
- emergence of global behavior

Swarm robotics

- robots cooperating to reach a certain goal
- decentralization of control
- limited communication abilities
- use of local information
- emergence of global behavior

$\Rightarrow$ *Although each robot is autonomous, the swarm can solve problems that a single robot can't.*

The Swarm Bot

In this presentation, we will focus on a so-called *Swarm Bot*, which is a swarm of *S-Bots*

The Swarm Bot

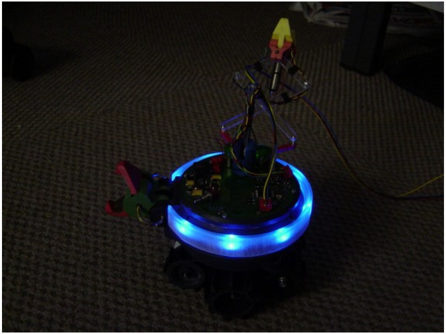
In this presentation, we will focus on a so-called *Swarm Bot*, which is a swarm of *S-Bots*

S-Bots

Mobile robots that can connect to/disconnect from each other

S-Bots

Mobile robots that can connect to/disconnect from each other



Tasks

Possible task for a Swarm Bot

Tasks

Possible task for a Swarm Bot

- move objects

Tasks

Possible task for a Swarm Bot

- move objects
- move through (tough) physical terrain

Tasks

Possible task for a Swarm Bot

- move objects
- move through (tough) physical terrain

For some tasks however, acting as single S-Bots might be more efficient, like finding a goal location or determining an optimal path.

Tasks

Possible task for a Swarm Bot

- move objects
- move through (tough) physical terrain

For some tasks however, acting as single S-Bots might be more efficient, like finding a goal location or determining an optimal path.

Focus of this presentation

Tasks

Possible task for a Swarm Bot

- move objects
- move through (tough) physical terrain

For some tasks however, acting as single S-Bots might be more efficient, like finding a goal location or determining an optimal path.

Focus of this presentation

- Aggregation

Tasks

Possible task for a Swarm Bot

- move objects
- move through (tough) physical terrain

For some tasks however, acting as single S-Bots might be more efficient, like finding a goal location or determining an optimal path.

Focus of this presentation

- Aggregation
- Coordinated motion

Self-organization

- system changes from a disordered to an ordered state using only local interactions

Self-organization

- system changes from a disordered to an ordered state using only local interactions
- uses positive/negative feedback

Self-organization

- system changes from a disordered to an ordered state using only local interactions
- uses positive/negative feedback
- Positive feedback:

Self-organization

- system changes from a disordered to an ordered state using only local interactions
- uses positive/negative feedback
- Positive feedback:
  - amplification of some property that emerges from random interactions (snow ball effect)

Self-organization

- system changes from a disordered to an ordered state using only local interactions
- uses positive/negative feedback
- Positive feedback:
    - amplification of some property that emerges from random interactions (snow ball effect)
    - increases exponentially over time

Self-organization

- system changes from a disordered to an ordered state using only local interactions
- uses positive/negative feedback
- Positive feedback:
  - amplification of some property that emerges from random interactions (snow ball effect)
  - increases exponentially over time
- Negative feedback:

Self-organization

- system changes from a disordered to an ordered state using only local interactions
- uses positive/negative feedback
- Positive feedback:
  - amplification of some property that emerges from random interactions (snow ball effect)
  - increases exponentially over time
- Negative feedback:
  - regulation that often gets triggered by positive feedback exhausting some resource

Self-organization

- system changes from a disordered to an ordered state using only local interactions
- uses positive/negative feedback
- Positive feedback:
  - amplification of some property that emerges from random interactions (snow ball effect)
  - increases exponentially over time
- Negative feedback:
  - regulation that often gets triggered by positive feedback exhausting some resource
- Positive and negative feedback interact, keeping the system in a stable state.

Problem

- Given a set of individual behaviors, it is difficult to predict the behavior that is going to emerge on a system level.

Problem

- Given a set of individual behaviors, it is difficult to predict the behavior that is going to emerge on a system level.
- Given a global behavior, it is difficult to decompose individual behaviors.

Possible solution

Artificial Evolution

Possible solution

Artificial Evolution

- bypasses decomposing the rules/mechanisms for the target behavior (which may not even be possible)

Possible solution

Artificial Evolution

- bypasses decomposing the rules/mechanisms for the target behavior (which may not even be possible)
- relies on the evaluation of the system as a whole

Possible solution

Artificial Evolution

- bypasses decomposing the rules/mechanisms for the target behavior (which may not even be possible)
- relies on the evaluation of the system as a whole
- can deal with the richness/complexity of the dynamic system, involving not only a multiple-agent scenario but also a possible physical link between the agents

Possible solution

Artificial Evolution

- bypasses decomposing the rules/mechanisms for the target behavior (which may not even be possible)
- relies on the evaluation of the system as a whole
- can deal with the richness/complexity of the dynamic system, involving not only a multiple-agent scenario but also a possible physical link between the agents
- is easy to implement

How can we use this for our Swarm Bot?

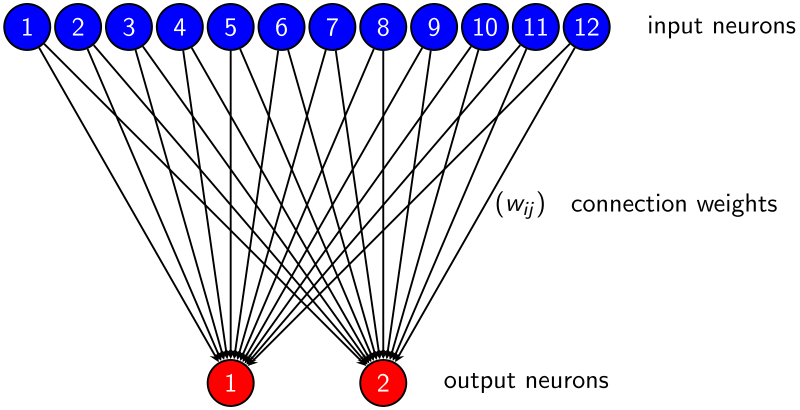
How can we use this for our Swarm Bot?

- connect sensory input and motor output through an artificial neural network

How can we use this for our Swarm Bot?

- connect sensory input and motor output through an artificial neural network
- determine the details of this network using artificial evolution

Single Layer Perceptron



Properties

- based on neural connections in the nervous system

Properties

- based on neural connections in the nervous system
- neurons are arranged in layers

Properties

- based on neural connections in the nervous system
- neurons are arranged in layers
- neurons outputs of one layer are connected to all neuron inputs of the subsequent layer

Properties

- based on neural connections in the nervous system
- neurons are arranged in layers
- neurons outputs of one layer are connected to all neuron inputs of the subsequent layer
- connections are *weighted*

Properties

- based on neural connections in the nervous system
- neurons are arranged in layers
- neurons outputs of one layer are connected to all neuron inputs of the subsequent layer
- connections are *weighted*
- network functionality depends on:

Properties

- based on neural connections in the nervous system
- neurons are arranged in layers
- neurons outputs of one layer are connected to all neuron inputs of the subsequent layer
- connections are *weighted*
- network functionality depends on:
  - network topology

Properties

- based on neural connections in the nervous system
- neurons are arranged in layers
- neurons outputs of one layer are connected to all neuron inputs of the subsequent layer
- connections are *weighted*
- network functionality depends on:
  - network topology
  - connection weights

One neuron

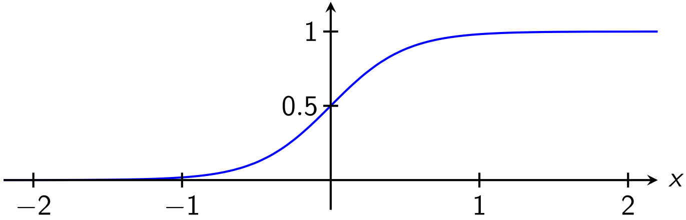
$$\text{input} \xrightarrow{\textit{weight}} \boxed{\text{activation function}} \longrightarrow \text{output}$$

One neuron

$$\text{input} \xrightarrow{\textit{weight}} \boxed{\text{activation function}} \longrightarrow \text{output}$$

Used activation function (sigmoid function):

$$f(x) = \frac{1}{1 + e^{-ax}}$$



One neuron

Neuron output depends on:

One neuron

Neuron output depends on:

1. output of previous neurons

One neuron

Neuron output depends on:

1. output of previous neurons
2. connection weights

One neuron

Neuron output depends on:

1. output of previous neurons
2. connection weights
3. activation function

One neuron

Neuron output depends on:

1. output of previous neurons
2. connection weights
3. activation function

In the following experiment, S-Bot controllers only differ in (2).

Abstract

- also based on biology

Abstract

- also based on biology
- uses *selection*, *reproduction* and *mutation*

Abstract

- also based on biology
- uses *selection*, *reproduction* and *mutation*
- optimizes on the basis of a given *fitness function*

Abstract

- also based on biology
- uses *selection*, *reproduction* and *mutation*
- optimizes on the basis of a given *fitness function*
- process:

Abstract

- also based on biology
- uses *selection*, *reproduction* and *mutation*
- optimizes on the basis of a given *fitness function*
- process:
    1. create initial population

Abstract

- also based on biology
- uses *selection*, *reproduction* and *mutation*
- optimizes on the basis of a given *fitness function*
- process:
    1. create initial population
    2. calculate the fitness of each individual in the current population

Abstract

- also based on biology
- uses *selection*, *reproduction* and *mutation*
- optimizes on the basis of a given *fitness function*
- process:
    1. create initial population
    2. calculate the fitness of each individual in the current population
    3. select a certain amount of well-fit individuals for reproduction

Abstract

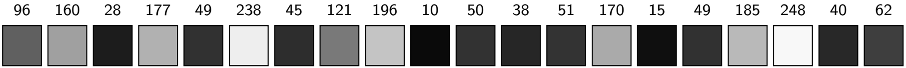
- also based on biology
- uses *selection*, *reproduction* and *mutation*
- optimizes on the basis of a given *fitness function*
- process:
  1. create initial population
  2. calculate the fitness of each individual in the current population
  3. select a certain amount of well-fit individuals for reproduction
  4. breed next generation using reproduction and mutation

Abstract

- also based on biology
- uses *selection*, *reproduction* and *mutation*
- optimizes on the basis of a given *fitness function*
- process:
  1. create initial population
  2. calculate the fitness of each individual in the current population
  3. select a certain amount of well-fit individuals for reproduction
  4. breed next generation using reproduction and mutation
  5. repeat 2-4

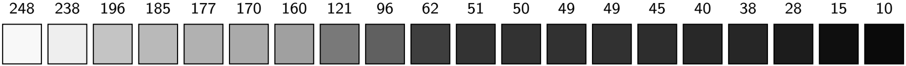
Example

Initialization



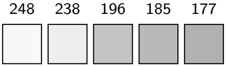
Example

Ordered by fitness



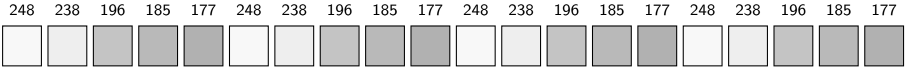
Example

Selection



Example

Reproduction



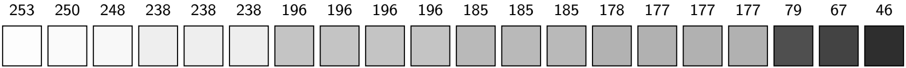
Example

Mutation

248	238	196	185	177	250	67	196	185	79	178	238	196	185	177	46	238	196	253	177

Example

Ordered by fitness again



Process

Five steps:

Process

Five steps:

1. The real robot is defined, including real hardware.

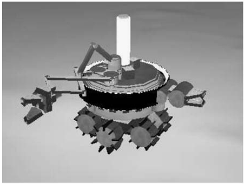
Process

Five steps:

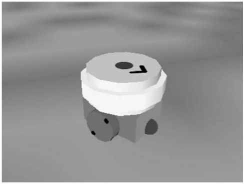
1. The real robot is defined, including real hardware.
2. A simulator is developed which can model the robot at different detail levels.

Process

(a)



(b)



(c)

Process

Five steps:

1. The real robot is defined, including real hardware.
2. A simulator is developed which can model the robot at different detail levels.
3. A simplified model is chosen, that permits to run evolutionary experiments in a reasonable amount of time.

Process

Five steps:

1. The real robot is defined, including real hardware.
2. A simulator is developed which can model the robot at different detail levels.
3. A simplified model is chosen, that permits to run evolutionary experiments in a reasonable amount of time.
4. Successful controllers from step 3 are improved using a detailed model.

Process

Five steps:

1. The real robot is defined, including real hardware.
2. A simulator is developed which can model the robot at different detail levels.
3. A simplified model is chosen, that permits to run evolutionary experiments in a reasonable amount of time.
4. Successful controllers from step 3 are improved using a detailed model.
5. Successful controllers from step 4 are improved on the real hardware.

Process

Five steps:

1. The real robot is defined, including real hardware.
2. A simulator is developed which can model the robot at different detail levels.
3. A simplified model is chosen, that permits to run evolutionary experiments in a reasonable amount of time.
4. Successful controllers from step 3 are improved using a detailed model.
5. Successful controllers from step 4 are improved on the real hardware.

Here: only 1-3.

Setup

- simplified simulation model used, no turret, no grippers

Setup

- simplified simulation model used, no turret, no grippers
- basically just a set of wheels with a speaker

Setup

- simplified simulation model used, no turret, no grippers
- basically just a set of wheels with a speaker
- the speaker always emits a sound, which can be sensed by other S-Bots using three sound sensors up to a distance of 75 cm

Setup

- simplified simulation model used, no turret, no grippers
- basically just a set of wheels with a speaker
- the speaker always emits a sound, which can be sensed by other S-Bots using three sound sensors up to a distance of 75 cm
- detection of neighbors or objects is done using 8 proximity sensors

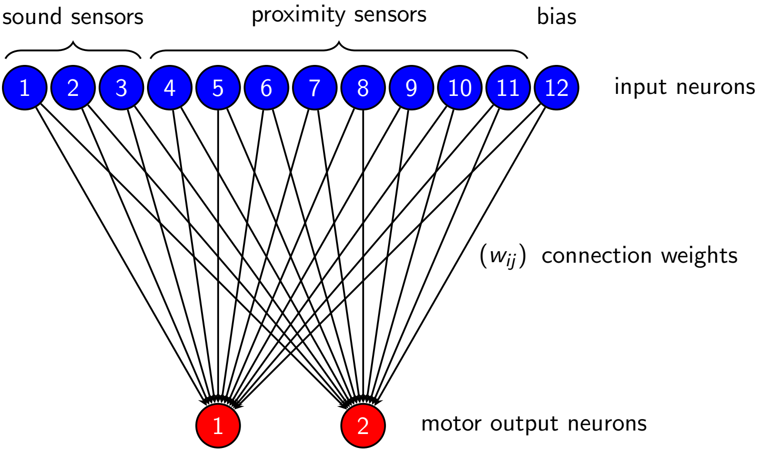
Setup

- simplified simulation model used, no turret, no grippers
- basically just a set of wheels with a speaker
- the speaker always emits a sound, which can be sensed by other S-Bots using three sound sensors up to a distance of 75 cm
- detection of neighbors or objects is done using 8 proximity sensors
- noise is simulated by a uniformly distributed random signal within $\pm5\%$ of the sensors saturation value

Setup

- simplified simulation model used, no turret, no grippers
- basically just a set of wheels with a speaker
- the speaker always emits a sound, which can be sensed by other S-Bots using three sound sensors up to a distance of 75 cm
- detection of neighbors or objects is done using 8 proximity sensors
- noise is simulated by a uniformly distributed random signal within $\pm 5\%$ of the sensors saturation value
- global area is 3×3 meters, bigger than the perceptual range of the S-Bots

Network structure



The evolutionary algorithm

- weights range in [-10, $+10$] and are represented by 8 bits

The evolutionary algorithm

- weights range in [-10, +10] and are represented by 8 bits
- each genotype consists of (12x2) x 8 = 192 bits.

The evolutionary algorithm

- weights range in [-10, +10] and are represented by 8 bits
- each genotype consists of (12x2) x 8 = 192 bits.
- start with 100 random genotypes. Each is tested over 8 epochs

The evolutionary algorithm

- weights range in [-10, +10] and are represented by 8 bits
- each genotype consists of (12x2) x 8 = 192 bits.
- start with 100 random genotypes. Each is tested over 8 epochs
- one epoch means: random number, positions and orientation of sbots

The evolutionary algorithm

- weights range in [-10, +10] and are represented by 8 bits
- each genotype consists of (12x2) x 8 = 192 bits.
- start with 100 random genotypes. Each is tested over 8 epochs
- one epoch means: random number, positions and orientation of sbots
- the top 20 genotypes produce 5 offsprings (flipping bits)

The evolutionary algorithm

- weights range in [-10, +10] and are represented by 8 bits
- each genotype consists of (12x2) x 8 = 192 bits.
- start with 100 random genotypes. Each is tested over 8 epochs
- one epoch means: random number, positions and orientation of sbots
- the top 20 genotypes produce 5 offsprings (flipping bits)
- each evolutionary run lasts 100 generations

The evolutionary algorithm

- weights range in [-10, +10] and are represented by 8 bits
- each genotype consists of (12x2) x 8 = 192 bits.
- start with 100 random genotypes. Each is tested over 8 epochs
- one epoch means: random number, positions and orientation of sbots
- the top 20 genotypes produce 5 offsprings (flipping bits)
- each evolutionary run lasts 100 generations
- tested for 20 evolutionary runs

The evolutionary algorithm

- fitness function accounts for number of S-Bots used

The evolutionary algorithm

- fitness function accounts for number of S-Bots used
- fitness of a genotype is the average fitness of all epochs

The evolutionary algorithm

- fitness function accounts for number of S-Bots used
- fitness of a genotype is the average fitness of all epochs
- genotype is evaluated for its aggregation and motion quality:

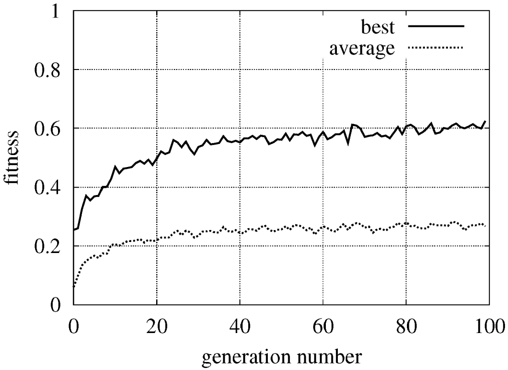
The evolutionary algorithm

- fitness function accounts for number of S-Bots used
- fitness of a genotype is the average fitness of all epochs
- genotype is evaluated for its aggregation and motion quality:
    - does the genotype minimize the distance between sbot and the center of the group?

The evolutionary algorithm

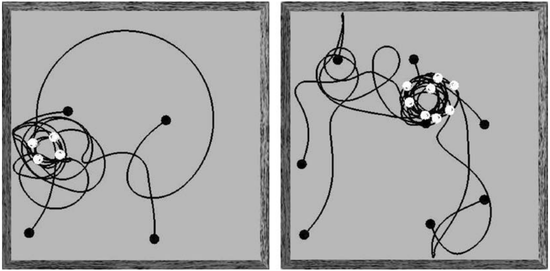
- fitness function accounts for number of S-Bots used
- fitness of a genotype is the average fitness of all epochs
- genotype is evaluated for its aggregation and motion quality:
  - does the genotype minimize the distance between sbot and the center of the group?
  - do the wheels of the sbot turn in the same direction?

The evolutionary algorithm



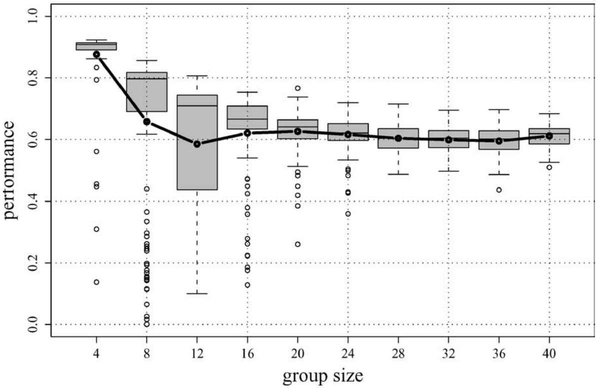
- Aggregation perfomance averaged over 20 evolutionary runs

Observations



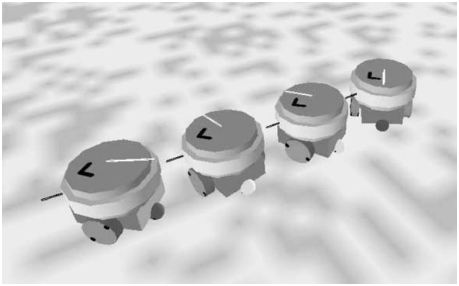
- Aggregation behavior.
  Following behavior of small groups emerges, if the group size gets larger, chaos increases.

Scalability



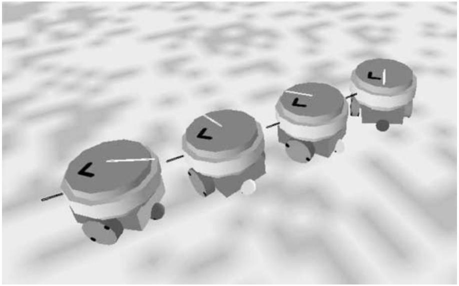
- Best genotype of each run tested against increasing group sizes.

Coordinated motion



- problem: S-Bot start with different orientations

Coordinated motion



- problem: S-Bot start with different orientations
- try to solve this problem and to evolve coordinated movement using only local information

Simulation

- no sound, S-Bots already connected via rigid links representing the grippers

Simulation

- no sound, S-Bots already connected via rigid links representing the grippers
- each S-Bot now has:

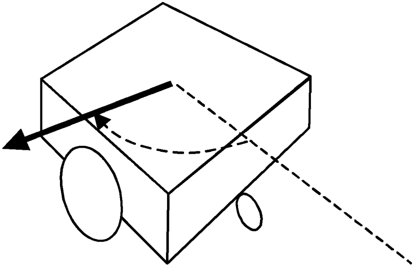
Simulation

- no sound, S-Bots already connected via rigid links representing the grippers
- each S-Bot now has:
  - a turret that can rotate with respect to its chassis

Simulation

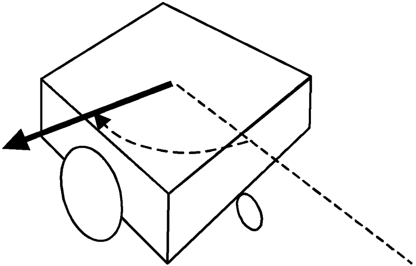
- no sound, S-Bots already connected via rigid links representing the grippers
- each S-Bot now has:
  - a turret that can rotate with respect to its chassis
  - a traction sensor indicating the angle between its turret and the chassis and the force of the traction

Simulation



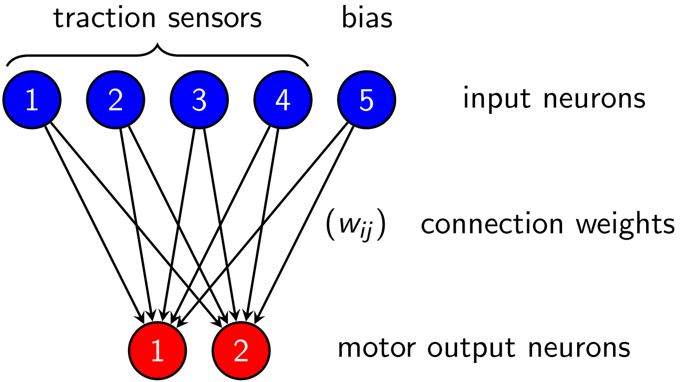
- traction sensor provides an average direction towards which the group is trying to move as a whole

Simulation



- traction sensor provides an average direction towards which the group is trying to move as a whole
- measure the mismatch between the group's direction and the S-Bot's chassis direction

Network structure



The evolutionary algorithm

- sensor inputs is the cosine of the angle diff' between each sensor's prefered direction and the group traction

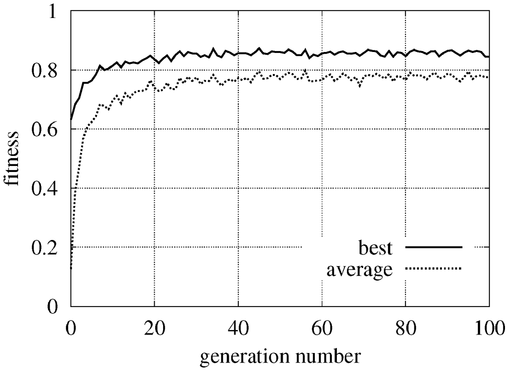
The evolutionary algorithm

- sensor inputs is the cosine of the angle diff' between each sensor's prefered direction and the group traction
- evolution basically the same as used with aggregation

The evolutionary algorithm

- sensor inputs is the cosine of the angle diff' between each sensor's prefered direction and the group traction
- evolution basically the same as used with aggregation
- measure the fitness for each epoche as the Euclidean distance between start and endpoint of the group

The evolutionary algorithm



- Coordinated movement performance for 20 runs à 100 generations.

Observed behavior

- each S-Bot makes a move in "its" direction

Observed behavior

- each S-Bot makes a move in "its" direction
- thereby each S-Bot receives a traction direction in correspondence to the majority of orientations among the group

Observed behavior

- each S-Bot makes a move in "its" direction
- thereby each S-Bot receives a traction direction in correspondence to the majority of orientations among the group
- evolved controller pattern:

Observed behavior

- each S-Bot makes a move in "its" direction
- thereby each S-Bot receives a traction direction in correspondence to the majority of orientations among the group
- evolved controller pattern:
  - if orientation is almost identical, traction is near 0, everyone just continues full speed

Observed behavior

- each S-Bot makes a move in "its" direction
- thereby each S-Bot receives a traction direction in correspondence to the majority of orientations among the group
- evolved controller pattern:
  - if orientation is almost identical, traction is near 0, everyone just continues full speed
  - if traction is low, each bot tweaks a bit in the average direction

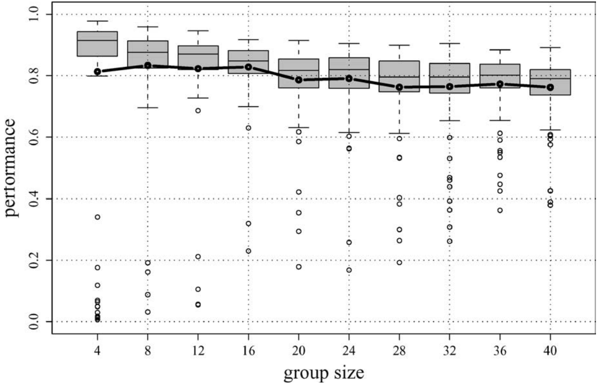
Observed behavior

- each S-Bot makes a move in "its" direction
- thereby each S-Bot receives a traction direction in correspondence to the majority of orientations among the group
- evolved controller pattern:
  - if orientation is almost identical, traction is near 0, everyone just continues full speed
  - if traction is low, each bot tweaks a bit in the average direction
  - if traction is high and orientation is highly misaligned then the S-Bots with higher difference change their orientation more rapidly than the ones who receive a higher traction

Observed behavior

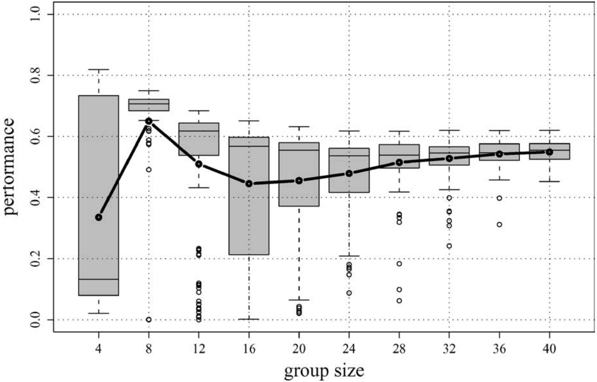
- each S-Bot makes a move in "its" direction
- thereby each S-Bot receives a traction direction in correspondence to the majority of orientations among the group
- evolved controller pattern:
  - if orientation is almost identical, traction is near 0, everyone just continues full speed
  - if traction is low, each bot tweaks a bit in the average direction
  - if traction is high and orientation is highly misaligned then the S-Bots with higher difference change their orientation more rapidly than the ones who receive a higher traction
- side effects: object avoidance and object pulling

Scalability



- Coordinated movement performance for increasing group sizes.

Detailed Model



- Aggregation performance for increasing group sizes (detailed model)

Summary

- multi-stage evolution possible, using the simplified model first to find a suitable solution then continue from there with a detailed model

Summary

- multi-stage evolution possible, using the simplified model first to find a suitable solution then continue from there with a detailed model
- Drawbacks:

Summary

- multi-stage evolution possible, using the simplified model first to find a suitable solution then continue from there with a detailed model
- Drawbacks:
  - the evaluation functions used in this simulation used global information (position of each S-Bot etc.)

Summary

- multi-stage evolution possible, using the simplified model first to find a suitable solution then continue from there with a detailed model
- Drawbacks:
  - the evaluation functions used in this simulation used global information (position of each S-Bot etc.)
  - to evolve these controllers on real hardware evaluation needs to be performed using only local available information

Summary

- multi-stage evolution possible, using the simplified model first to find a suitable solution then continue from there with a detailed model
- Drawbacks:
  - the evaluation functions used in this simulation used global information (position of each S-Bot etc.)
  - to evolve these controllers on real hardware evaluation needs to be performed using only local available information
  - could be bypassed by using a camera in a real setup to obtain positions