Intelligent autonomous robots

The Use of Artificial Evolution in Robotics

Seminar presented at
NCTU
Aug. 20, 2007
SIAR2007

Mattias Wahde

mattias.wahde@chalmers.se

Schedule

Fundamentals of evolutionary algorithms	09.30-10.30
Evolutionary robotics	10.50-11.50
Behavior selection	13.30-14.30
Robots intended for transportation	14.50-16.20

Some abbreviations (see also p. iii)

ANN	=	Artificial neural network
BBR	=	Behavior-based robotics
EA	=	Evolutionary algorithm
ER	=	Evolutionary robotics
GA	=	Genetic algorithm
RNN	=	Recurrent neural network

Introduction to evolutionary robotics (ER)

- Subfield of robotics, in which evolutionary algorithms (EAs) are used for generating robotic brains (or bodies, or both).
- Method: **behavior-based robotics (BBR)**.
- In BBR, robotic brains are built in a bottom-up fashion (more about this later).

pp. 1-3

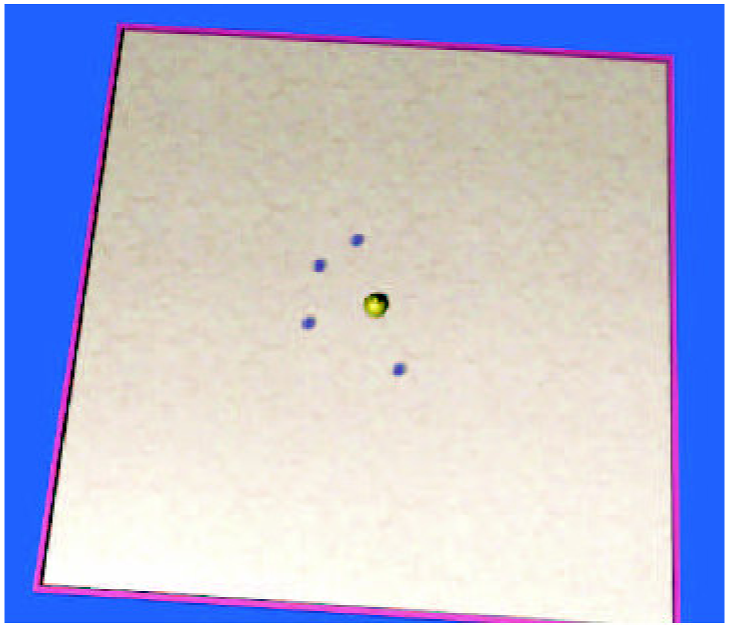
- ER Deals mostly with **autonomous robots**, i.e. robots that *move freely in unstructured environments, without human supervision.*

- **Unstructured environments:** Environments that chang rapidly and in an unpredictable way (maps unreliable...)

pp. 1-3

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Example: Cleaning behavior

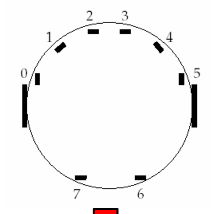


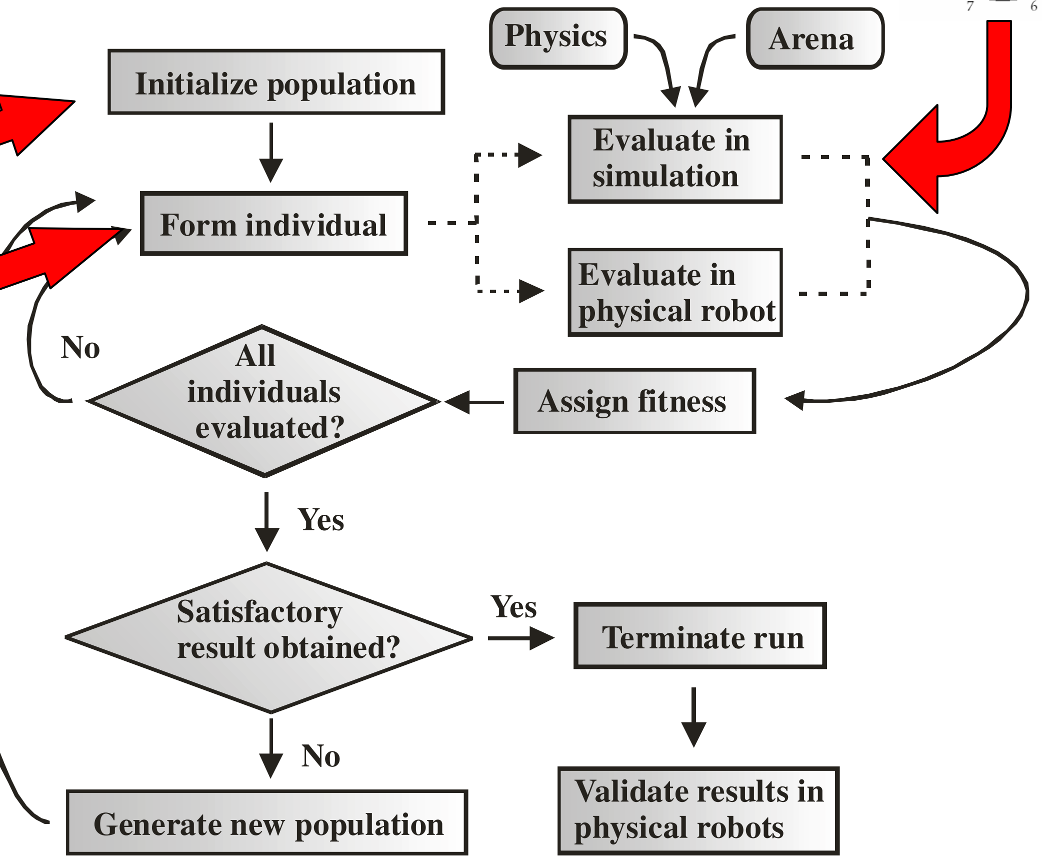
Reference [137]

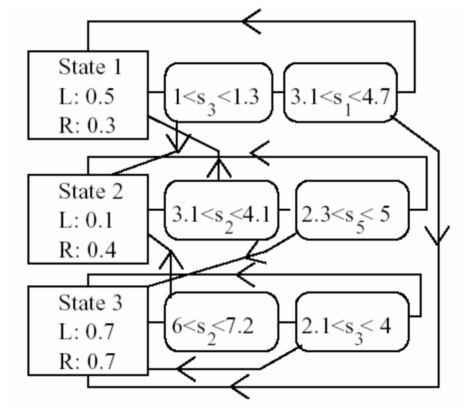
pp. 3-5

Procedure

Randomly







Brain (example)

pp. 3-5

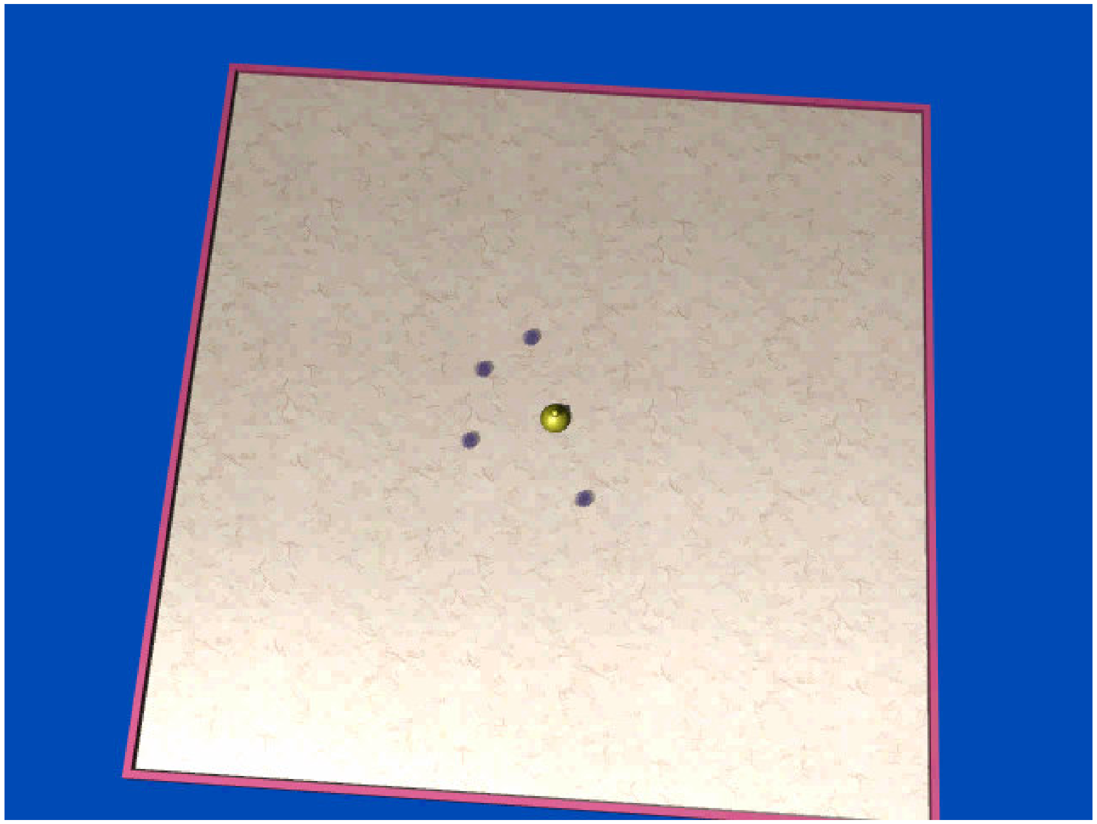
Fitness measure

The mean squared distance (measured from the center of the arena) at the end of the evaluation.

NOTE: No direct credit assignment for individual actions!

pp. 3-5

Results (simulation)



pp. 3-5

Validation in a physical robot

An essential step!

pp. 3-5

Optimization methods

- Taxonomy:
  - Deterministic algorithms (essentially classical optimization algorithms)
  - Stochastic algorithms (evolutionary algorithms, ant colony optimization, particle swarm optimization)

- Classical optimization algorithms (e.g. gradient descent, Newton's method, interior point methods etc.) are applicable in many problems in science and engineering

pp. 38-39

Optimization methods

- Classical optimization methods are particularly useful in **convex problems**.
- However, there are many problems for which classical optimization methods are not suitable, e.g. problems in which..
  - ..the value of the objective function (the quantity being optimized) can only be obtained as a result of e.g. a time-consuming simulation.
  - ..the number of variables is itself variable during optimization (e.g. in the optimization of recurrent neural networks for autonomous robots).
- In these cases, stochastic optimization methods are more appropriate.

pp. 38-39

Stochastic optimization

- Many different classes of algorithms exist, e.g.
  - Evolutionary algorithms
  - Particle swarm algorithms
  - Ant colony optimization
  - Tabu search
  - Simulated annealing

  etc etc.
- Here, we shall only consider evolutionary algorithms.
- PS.. I'm writing a book, "Stochastic optimization algorithms" (completed around Jan. 2008..)

pp. 38-39

Basics of evolutionary algorithms

EAs are methods for search and optimization, inspired by darwinian evolution.

Central concepts: gradual, hereditary change as a response to challenges (to survival) from the environment.

pp. 6-21

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Example

pp. 6-21

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

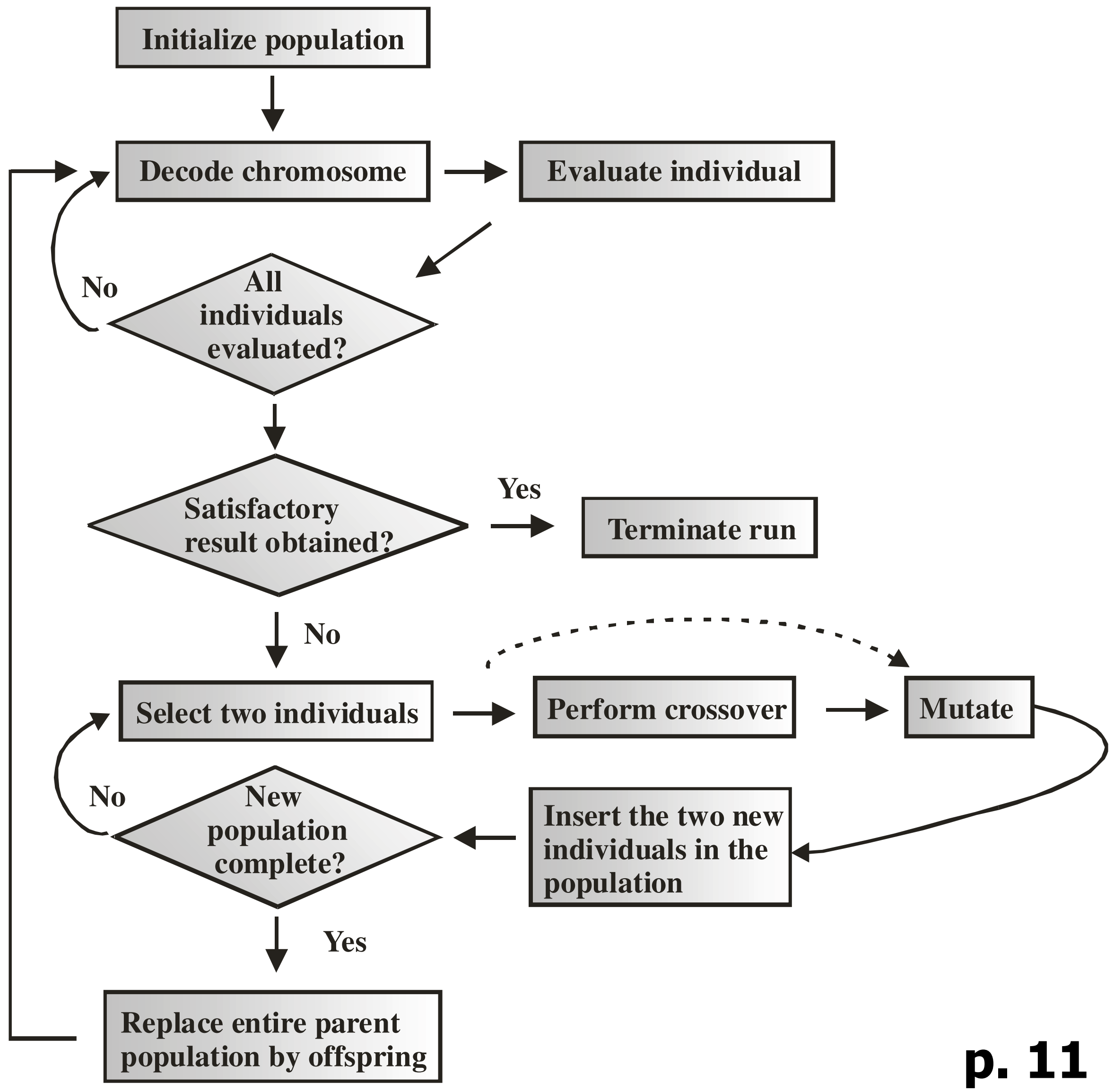
Fundamental concepts

- Genome
- Populations
- Species
- Fitness
- Selection
- Reproduction

pp. 6-21

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Basic flow of an EA



Detailed description

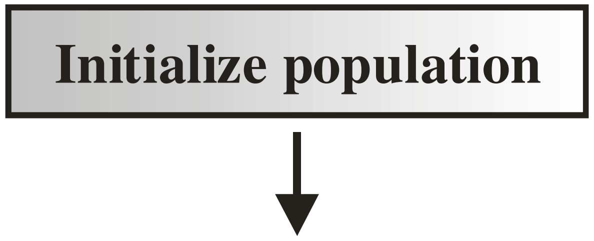
- Consider the case of function maximization, applied to the function

$$\psi_n(x_1, x_2, \ldots, x_n) = \frac{1}{2} + \frac{1}{2n} \exp\left(-\alpha \sum_{i=1}^{n} x_i^2\right) \sum_{i=1}^{n} \cos\left(\beta\sqrt{i}x_i \sum_{j=1}^{i} j x_j\right)$$

- (Maximum = 1, at $x_1 = x_2 = \ldots = x_n = 0$).

p. 12-13

Step 1

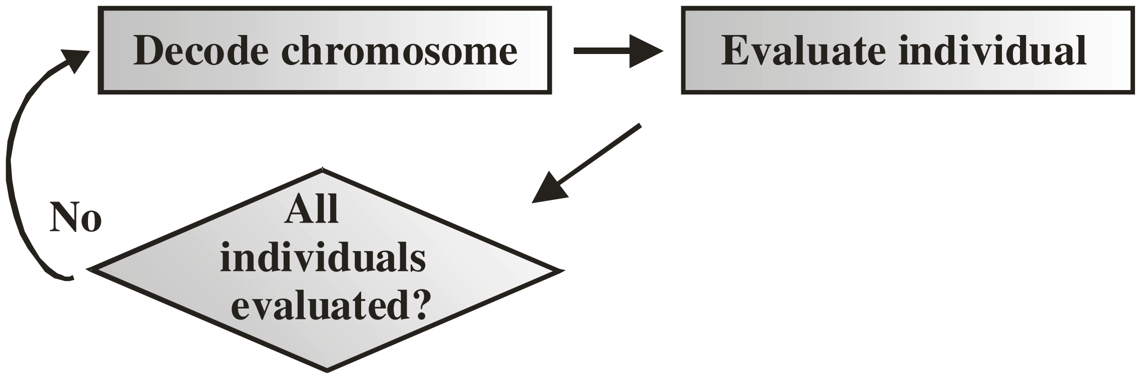


- Identify **variables** (in this case: x_i, i=1,...,n).
- Select an encoding scheme (e.g. binary)

$$\underbrace{0\,1\,0\,1\,1\,0\,1\,1\,1\,0\,1\,0\,1\,1\,1\,0\,1\,1\,0\,0\,0\,1\,0\,1\,0\,1\,1}_{x_1}\underbrace{0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,0\,1\,1\,1\,0\,0\,0\,1\,0\,1\,0\,0\,1\,0\,1\,1\,0\,1}_{x_2}\ldots$$

- Initialize N such **chromosomes** randomly.
- Each entry is called a **gene**.

Step 2

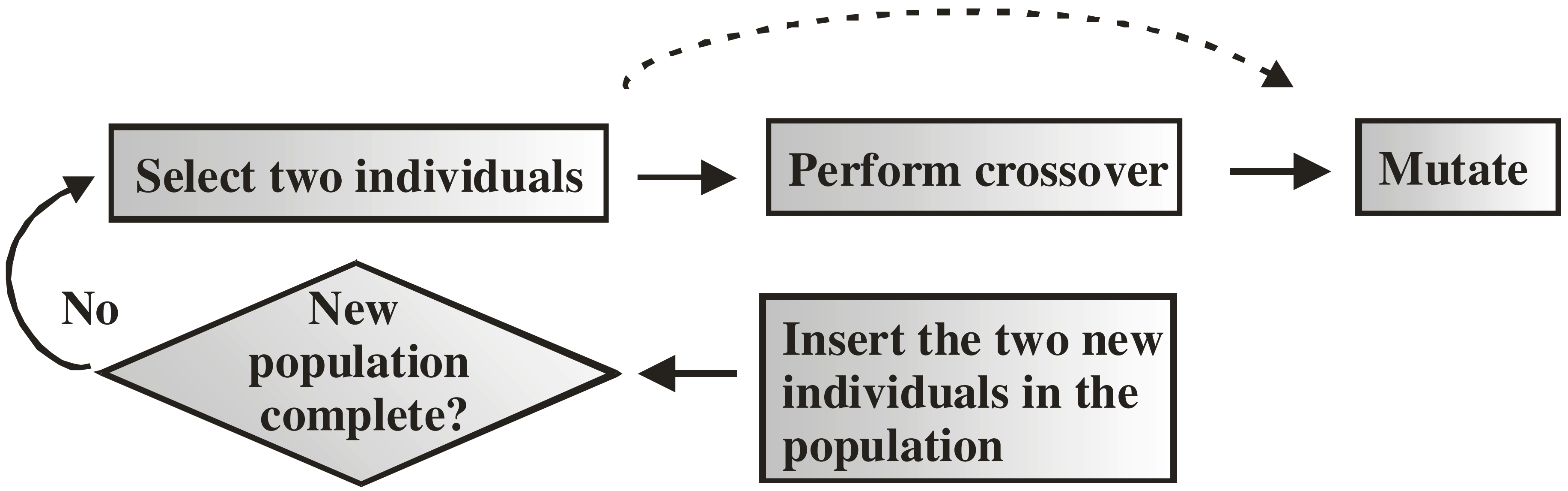


- Decode chromosome => **individual**, in this case the variables x_i.

- Evaluate the individual: requires **fitness measure**. In this case, use simply the function value $f(x_1,x_2, \ldots ,x_n)$.

p. 14-15

Step 3: selection and reproduction



p. 15-19

Step 3.1

Select two individuals

- Select individuals in proportion to fitness.

- Two common methods: **roulette-wheel selection** and **tournament selection**

p. 15-19

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

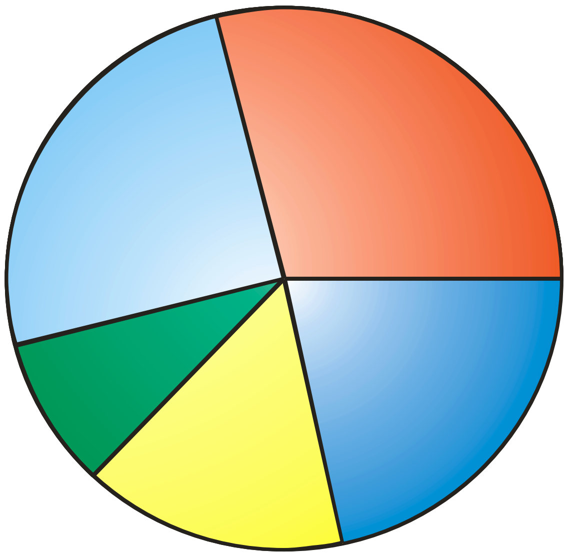
Roulette-wheel selection

- Each individual occupies a space, proportional to its fitness, on a roulette-wheel:

- In practice, select smallest j such that

$$\frac{\sum_{i=1}^{j} f_i}{\sum_{i=1}^{N} f_i} > r,$$

(f_i = fitness of individual i)



p. 15-19

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Tournament selection

- Choose m individuals randomly from the population. Next, select the best one with probability p_{tour}.(Otherwise pick one of the others, randomly).

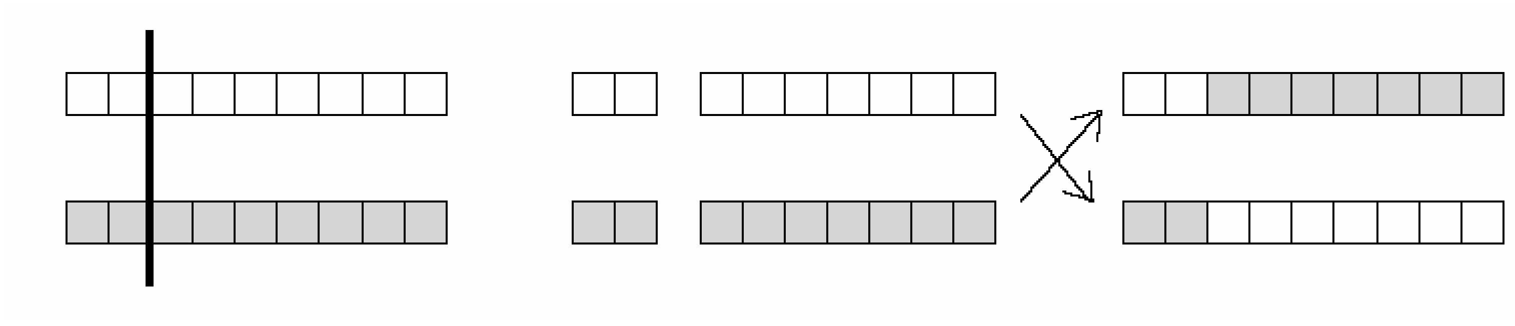
p. 15-19

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Step 3.2

Perform crossover

- Crossover:



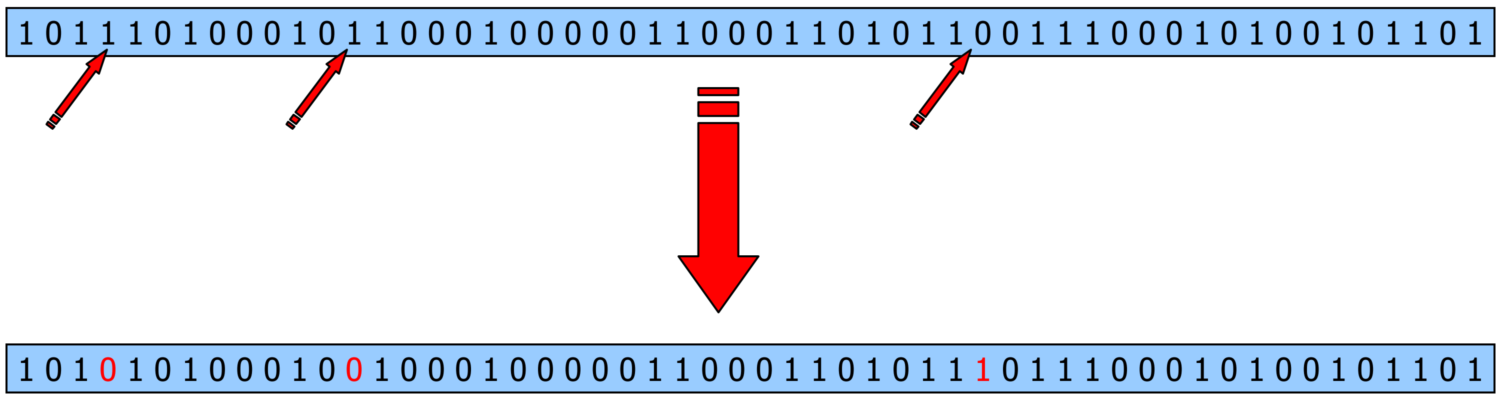
- Randomly selected crossover point.
- Carried out only with probability p_c.

p. 15-19

Step 3.3

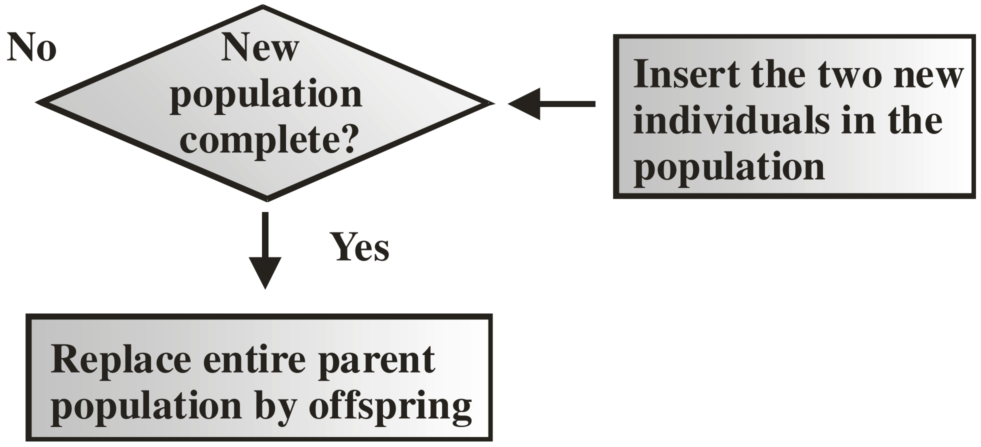
Mutation

- Randomly change a few genes (probability p_{mut} per gene):



p. 15-19

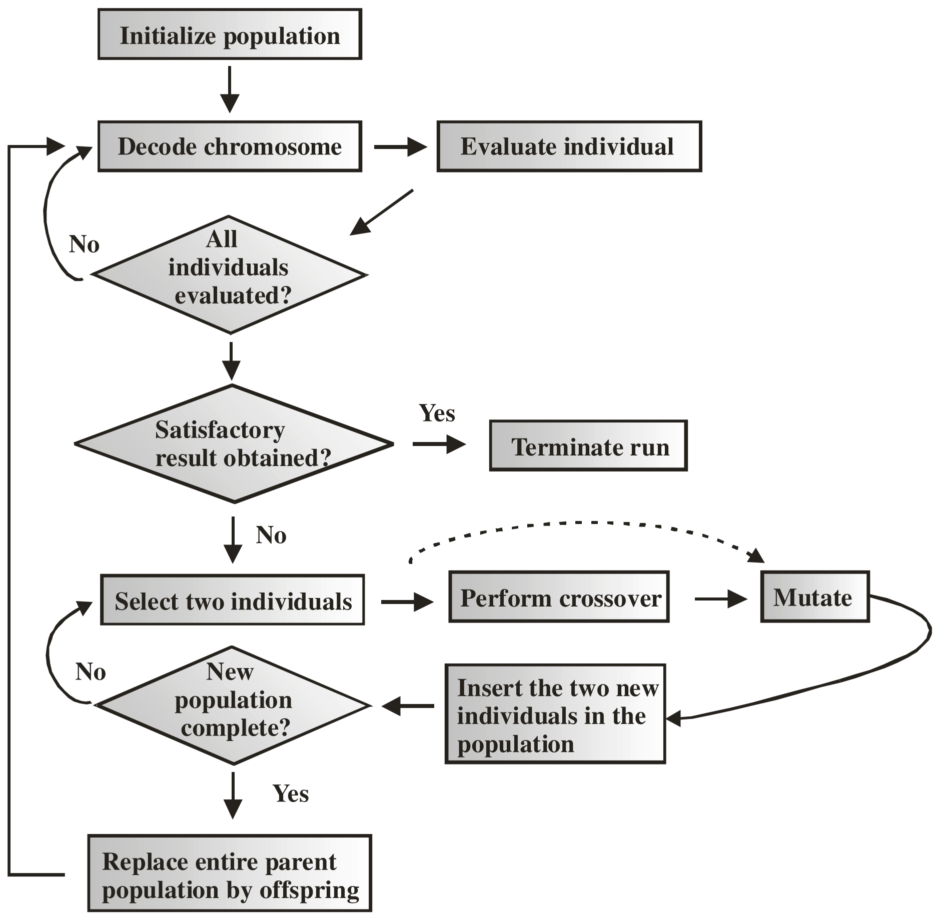
- The procedure is repeated until N new individuals have been formed:


No
New population complete?
Insert the two new individuals in the population
Yes
Replace entire parent population by offspring

- The parent population is then discarded, and the evaluation of the first generation is complete.

p. 15-19

- The process of evaluation, selection, and reproduction is repeated until a satisfactory solution has been found.



Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Important aspects of EAs

- **Flexibility** (no gradient, credit for individual actions etc. needed).
- **Efficiency** (can cope with very large and complex search spaces).
- **Versatility** (applicable in almost all problems involving optimization).

Some difficulties

- Selecting a representation (i.e. whether or not to use chromosomes at all etc.)
- Selecting a fitness measure
- Premature convergence
- Convergence to suboptimal solutions may occur (applies to all stochastic optimization methods)

Biological evolution vs. EAs

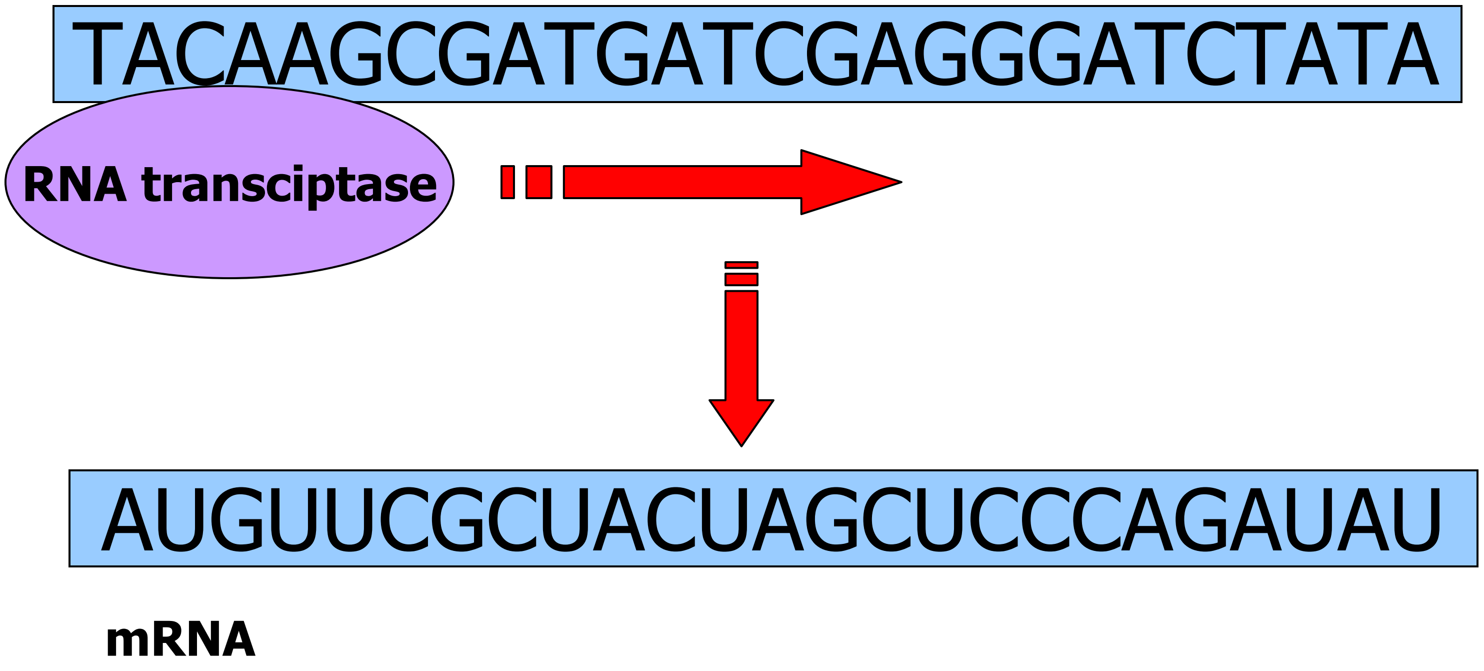
- EAs are strongly simplified compared to biological evolution:

- In EAs there is rarely any **gene regulation**, cell division etc. Chromosomes are used as lookup tables.

p. 8-10

Genetic information in biology

Transcription

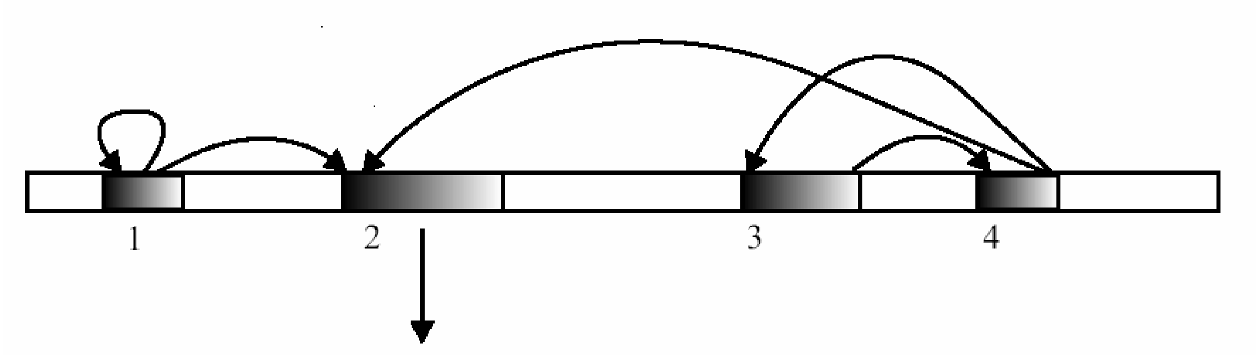


Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

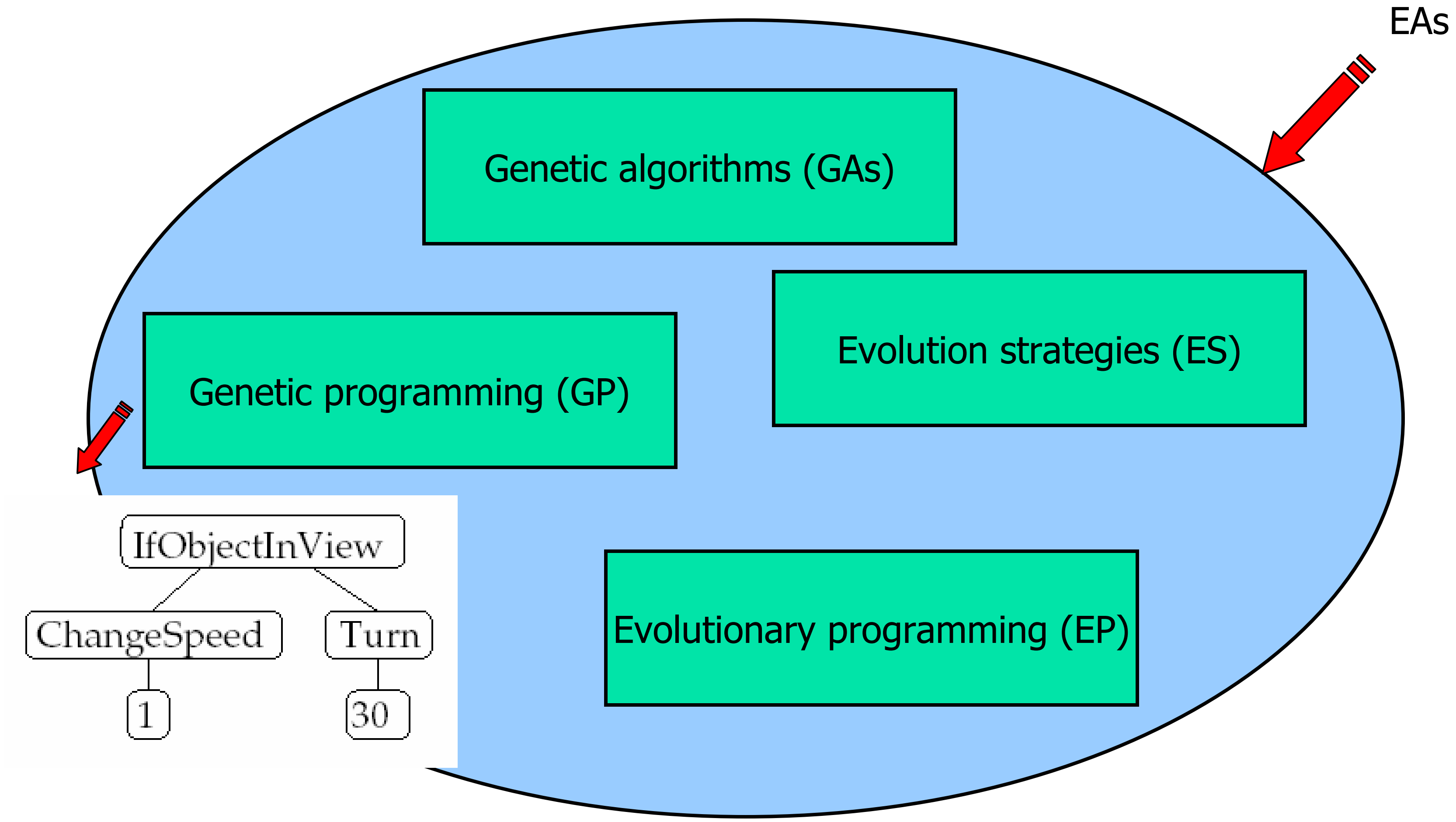
Translation

$$\underbrace{\text{AUG}}_{\textbf{start}}\underbrace{\text{UUC}}_{\textbf{Glu}}\underbrace{\text{FCU}}_{\textbf{Ser}}\text{ACUAGCUCCCAGAUAU}\quad \textbf{etc..}$$

- The amino acid sequence forms a protein.
- Gene regulation:

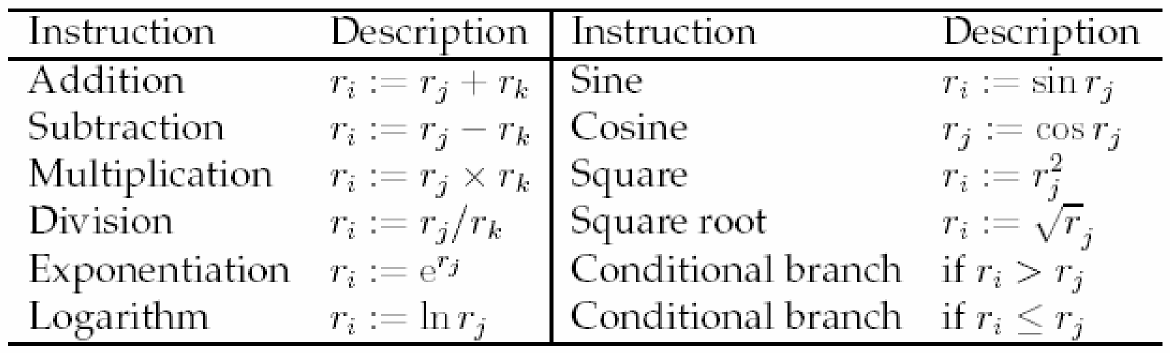


Examples of EAs



(Linear) genetic programming

- Used for evolving programs consisting of linear sequences of basic instructions.
- Applicable in cases where the target structure is unknown.



Instruction	Description	Instruction	Description
Addition	$r_i := r_j + r_k$	Sine	$r_i := \sin r_j$
Subtraction	$r_i := r_j - r_k$	Cosine	$r_j := \cos r_j$
Multiplication	$r_i := r_j \times r_k$	Square	$r_i := r_j^2$
Division	$r_i := r_j / r_k$	Square root	$r_i := \sqrt{r_j}$
Exponentiation	$r_i := e^{r_j}$	Conditional branch	if $r_i > r_j$
Logarithm	$r_i := \ln r_j$	Conditional branch	if $r_i \leq r_j$

Table 2.1: *Examples of typical LGP operators. Note that the operands can be either variable registers or constant registers.*

pp. 22-25

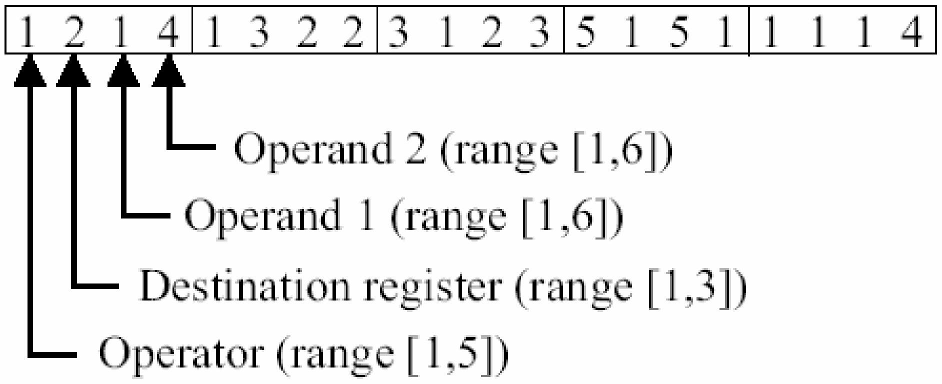
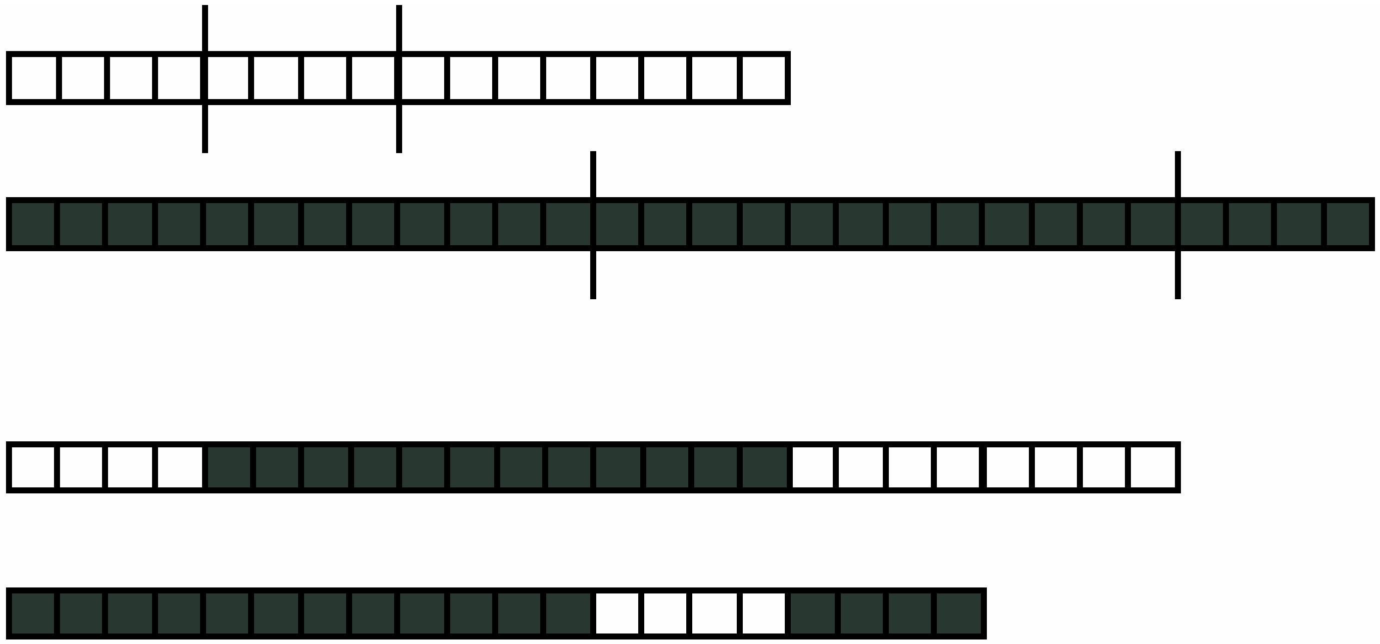


Figure 2.9: *An example of an LGP chromosome.*

Genes	Instruction	Result
1, 2, 1, 4	$r_2 := r_1 + c_1$	$r_1 = 1, r_2 = 2, r_3 = 0$
1, 3, 2, 2	$r_3 := r_2 + r_2$	$r_1 = 1, r_2 = 2, r_3 = 4$
3, 1, 2, 3	$r_1 := r_2 \times r_3$	$r_1 = 8, r_2 = 2, r_3 = 4$
5, 1, 5, 1	if $(r_1 > c_2)$	$r_1 = 8, r_2 = 2, r_3 = 4$
1, 1, 1, 4	$r_1 := r_1 + c_1$	$r_1 = 9, r_2 = 2, r_3 = 4$

Table 2.2: *Evaluation of the chromosome shown in Fig. 2.9, in a case where the input register r_1 was initially assigned the value 1. The variable registers r_2 and r_3 were both set to 0, and the constant registers were set as $c_1 = 1, c_2 = 3, c_3 = 10$. The first instruction (top line) is decoded from the first four genes in the chromosome etc. The resulting output was taken as the value contained in r_1 at the end of the calculation.*

pp. 22-25



Optimization of structures of variable size

- Particularly relevant for ER: **Artificial neural networks** (ANN).
- Distributed systems for computation, (loosely) based on biological neural networks.
- Two kinds: **feedforward networks** (FFNN) and **recurrent networks** (RNN).
- There exists many training methods for neural networks, e.g. backpropagation.

Recurrent neural networks (RNNs)

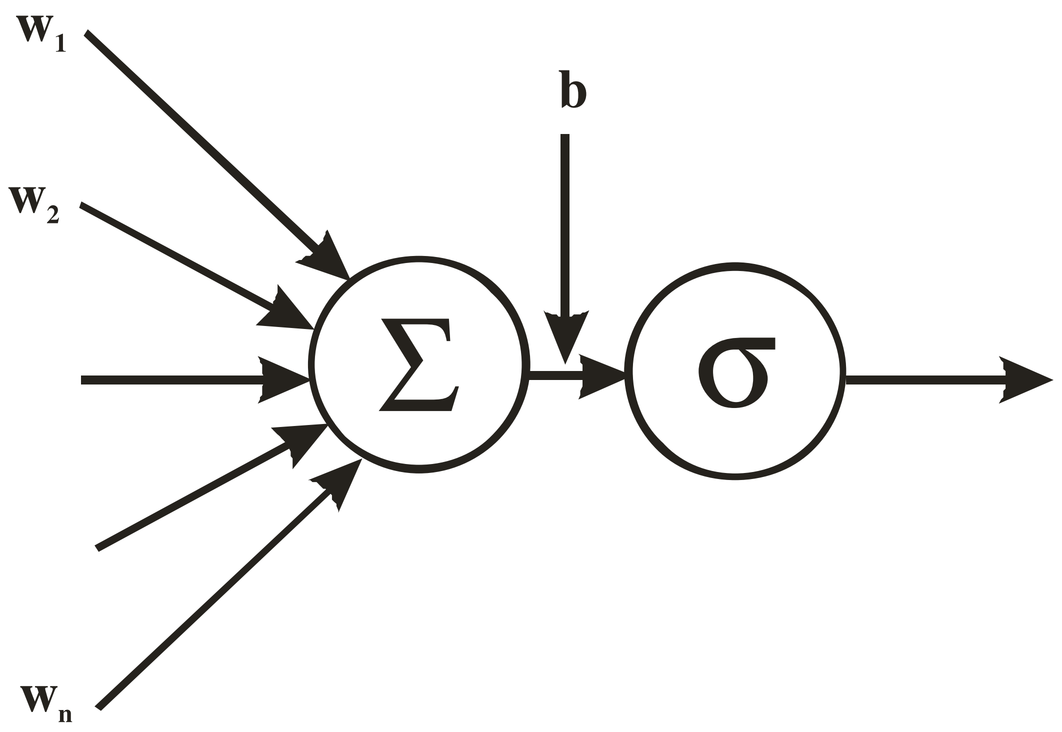
- For autonomous robots, few dedicated neural-network methods are applicable: no input-output-examples (only an overall evaluation).

- Alternative: use stochastic optimization (e.g. EAs)

pp. 94-97

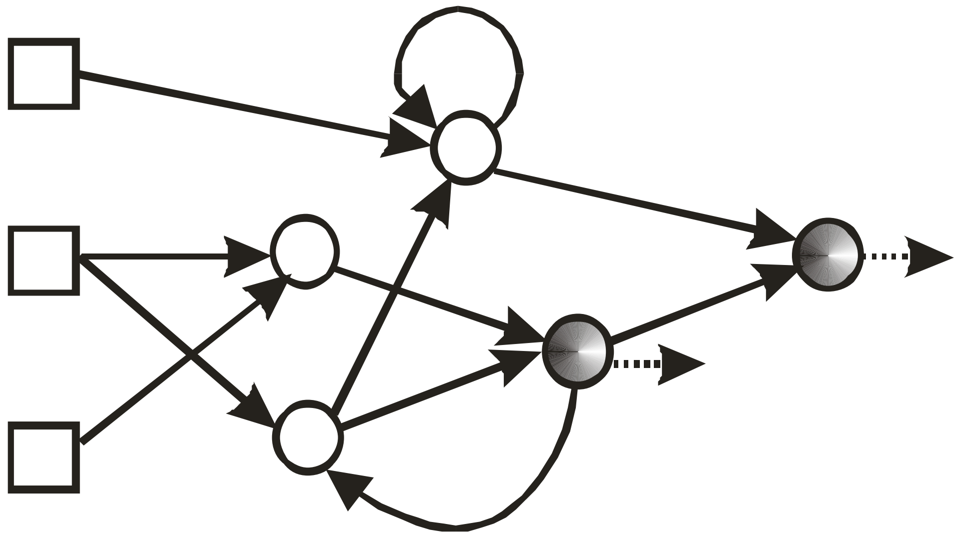
Recurrent neural networks (RNNs)

- Neuron model:



pp. 94-97

Recurrent neural networks (RNNs)

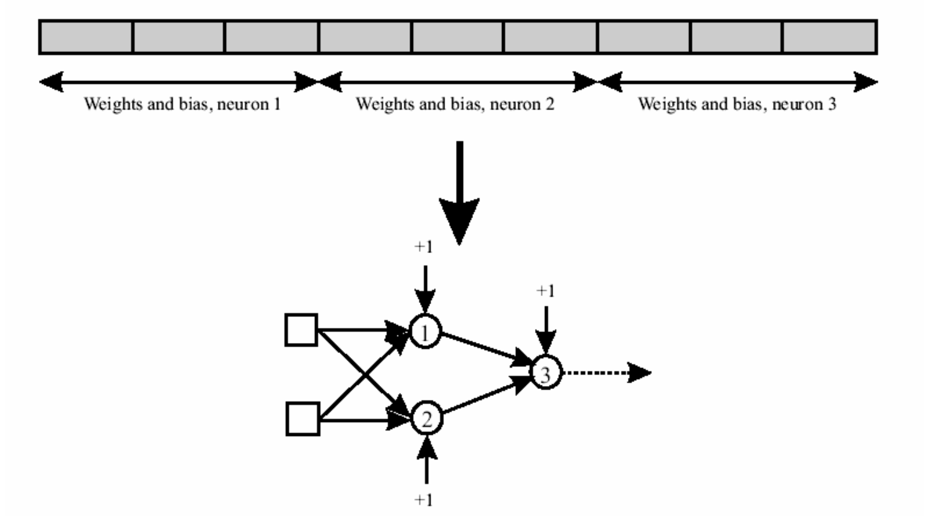


- Network equations (see Appendix A):

$$\tau_i \dot{x}_i(t) + x_i(t) = \sigma\left(\sum_j w_{ij} x_j(t) + \sum_j w^{\mathrm{I}}_{ij} I_j(t) + b_i\right)$$

Evolution of RNNs

- For a given architecture, it is easy to implement an EA that optimizes the network parameters:

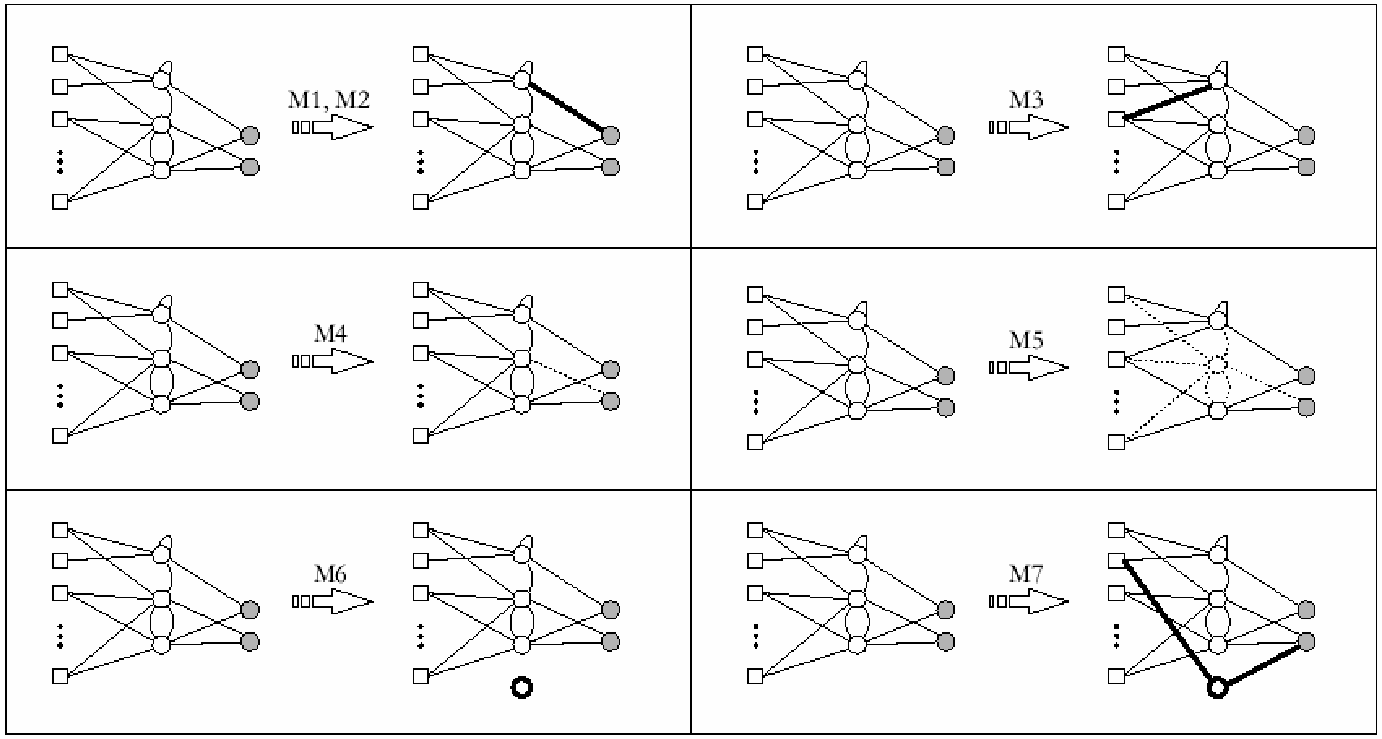

Weights and bias, neuron 1
Weights and bias, neuron 2
Weights and bias, neuron 3
+1
+1
+1
1
2
3

Evolution of RNNs

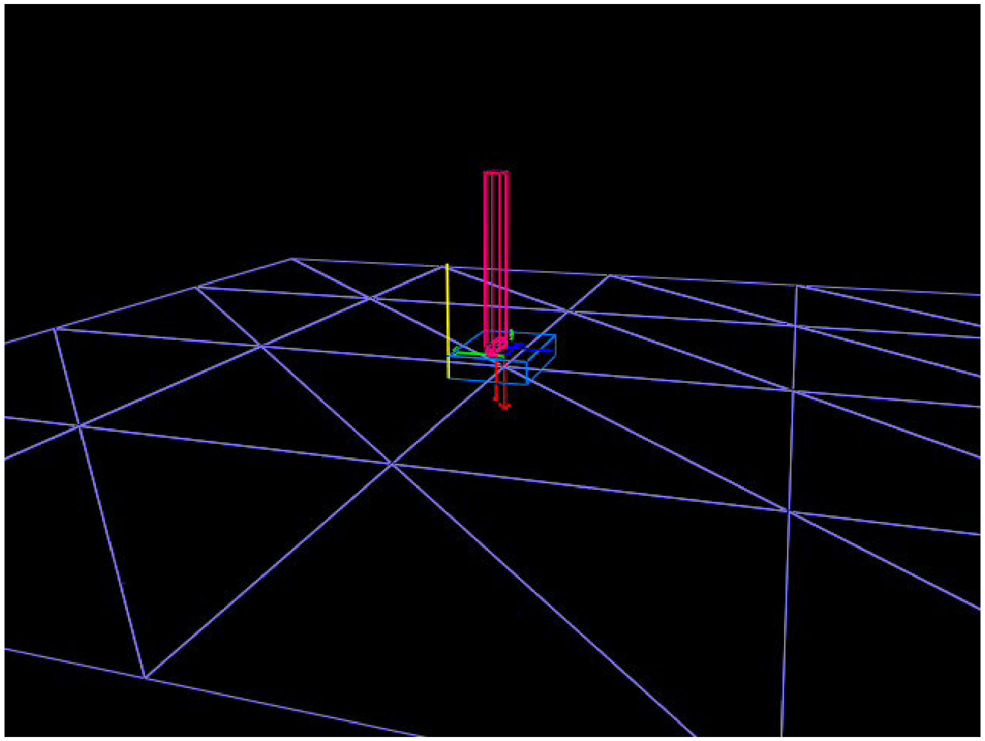
- Often difficult to specify architecture a priori, particularly in ER.
- Skip the step of chromosomal encoding, and let the EA operate directly on the RNNs.
- Crossover often has a negative effect on RNNs (distributed computation).
- Allow variations in network architecture.

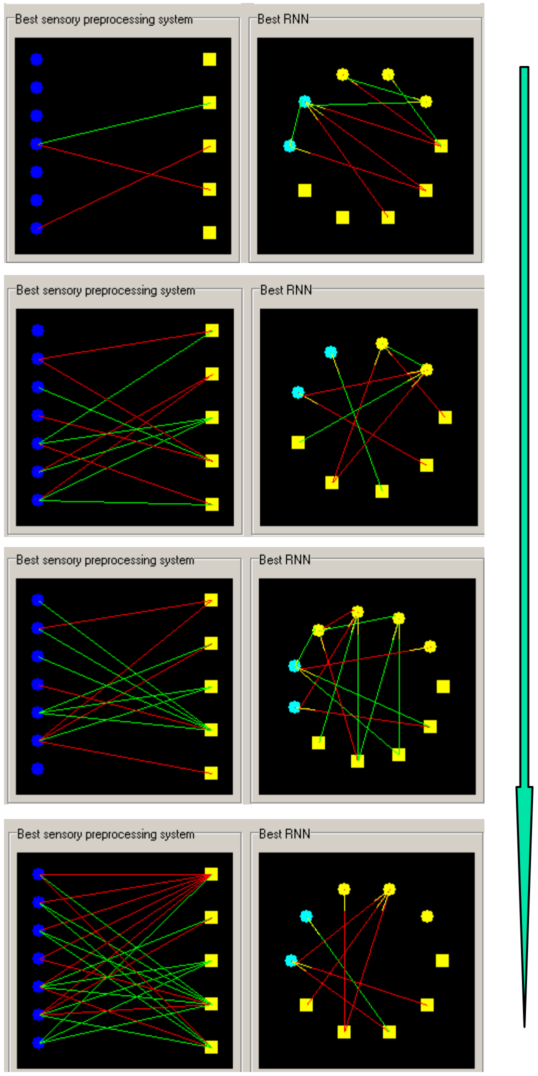
Evolution of RNNs

- Mutation operators:



Example: One-legged hopping



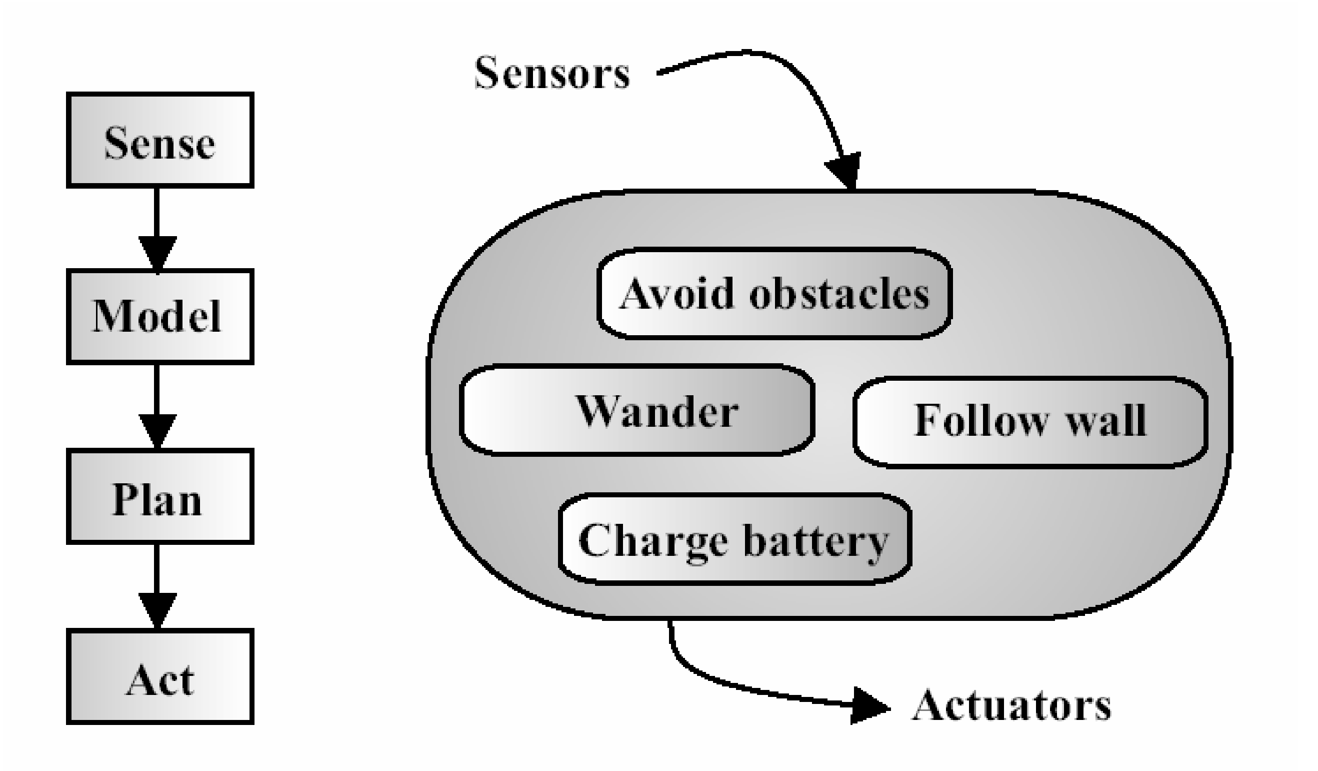

Best sensory preprocessing system
Best RNN
Best sensory preprocessing system
Best RNN
Best sensory preprocessing system
Best RNN
Best sensory preprocessing system
Best RNN

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Evolutionary robotics

- ER: generating robotic brains using EAs.
- Note: *robotic brain* rather than *control system*.
- ER is normally used in connection with BBR.

BBR vs. Classical AI



Classical AI

- ~1950s -
- Focused on high-level reasoning (i.e. human-level intelligence).
- Uses complex world models.
- Successful in subfields of robotics (e.g. navigation, image recognition etc.).
- Not well suited for generating autonomous robots capable of handling rapidly changing environments.

Behavior-based robotics (BBR)

- ~1980s -
- Builds robotic control systems (**robotic brains**) in a bottom-up fashion, starting from simple behaviors.
- Examples of behaviors: *avoid obstacles*, *follow wall*, *charge batteries* etc.
- More generous definition of intelligence, e.g. *the ability to survive and to strive to reach other goals, in an unstructured environment.*
- Rooted in ethology.
- Well suited for unstructured environments.

p. 40-42

Behavior-based robotics (BBR)

- In practice, robotic brains often merge the two approaches (classical AI and BBR).
  - BBR: Basic, survival-related behaviors
  - Classical AI: Reasoning and other complex behaviors.
- Critique of BBR: "*Only applicable to toy problems*". True?
- ...So far, yes (to some extent). Can the BBR-approach be extended, to generate more complex robotic brains?...
- ...Probably, yes, but that will involve solving the problem of **behavior selection**, i.e. activation of behaviors at the correct moment.

Behavior-based robotics (BBR)

- Two important problems in ER:

(1) Evolution of elementary behaviors.

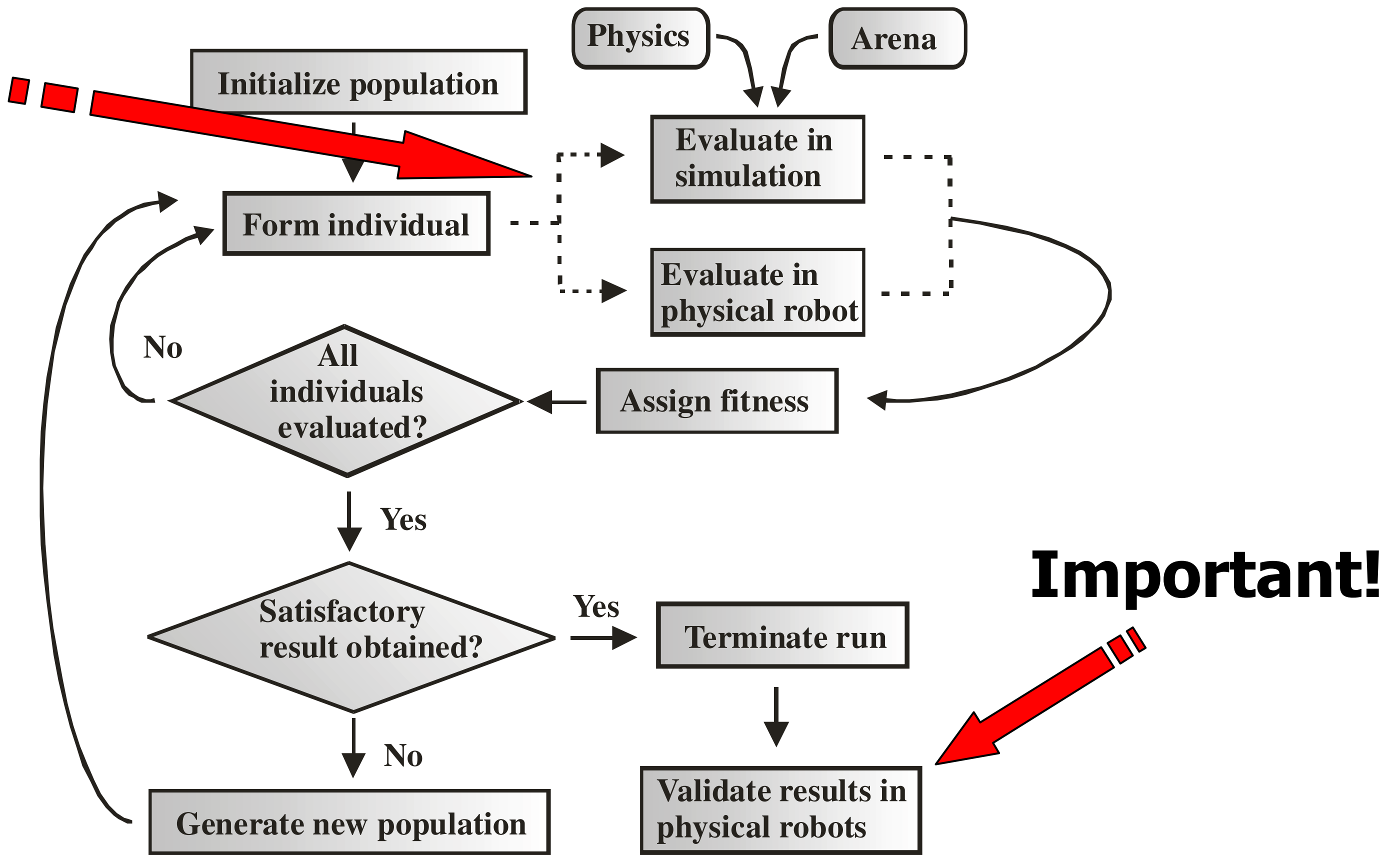
(2) Evolution of complete robotic brains, either by directly evolving complex behaviors, or through **behavioral organization**.

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Behavior-based robotics (BBR)

- Note that the definition of **behaviors** and **actions** is somewhat fuzzy.
- Normally, a behavior is a sequence of actions.
- In some cases, the distinction between behaviors is not altogether clear, either.

Basic flow of an ER investigation



Simulations vs. evolution in hardware

- Simulations are generally faster than evolution in hardware.
- Problem: **Reality gap** [53], since

(1) Difficult to capture all aspects of reality in a simulation
(2) Real environments are noisy, on all levels.
(3) Differences exist even between supposedly identical sensors.

Evolution in hardware

- Problems:

  (1) Time-consuming.

  (2) Power supply. Powered floor possible [33], but slow charging. Another possibility: use capacitors [102].

  (3) Many applications require continuous monitoring.

Evolution in hardware

- Most common approach: use a single robot, upload each brain, and evaluated consecutively [34], [94].
- Alternative: **Embodied evolution** [33], [140]. Exchange of genetic material between physical robots, using IR transmission.

Improving simulations

- General guidelines [59]:

(1) Base simulations on empirical data rather than e.g. artificial sensor models.

(2) Include the correct amount of noise.

(3) Use a noise-tolerant representation, e.g. ANNs.

Evolution of single behaviors

A few examples:

- Wandering,
- (Simple) navigation,
- Walking,
- Box-pushing,
- Motion of a robotic arm

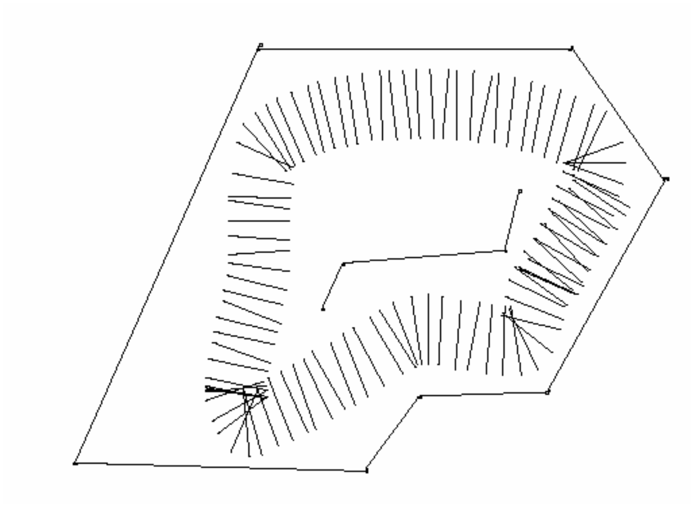
Wandering

- Floreano and Mondada [34].
- Evolution in hardware (Khepera).
- Simple, ANNs as robotic brain. Some recurrent couplings in output layer.
- Fitness measure:

$$\Phi = V\left(1 - \sqrt{|\Delta v|}\right)(1 - i)$$

p. 47-48

Wandering



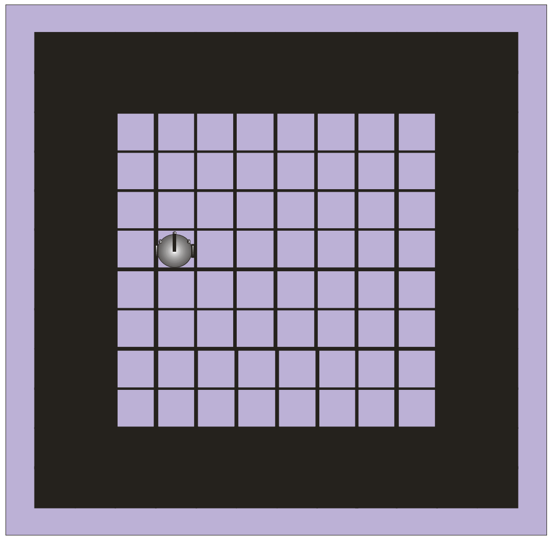
- Long generation time (40 min.) due to evolution in hardware.

Wandering

- Jakobi et al. [59] considered a similar problem.
- They found that the factor (1-i) was unnecessary in a cluttered environment.
- **Implicit fitness measure**.
- Note: limitations on fitness measures if evolution in hardware is used.

Wandering

- Miglino et al. [87] evolved wandering behavior.
- Robots were placed in a 26x26 grid, with the central 20x20 squares white and the rest black.

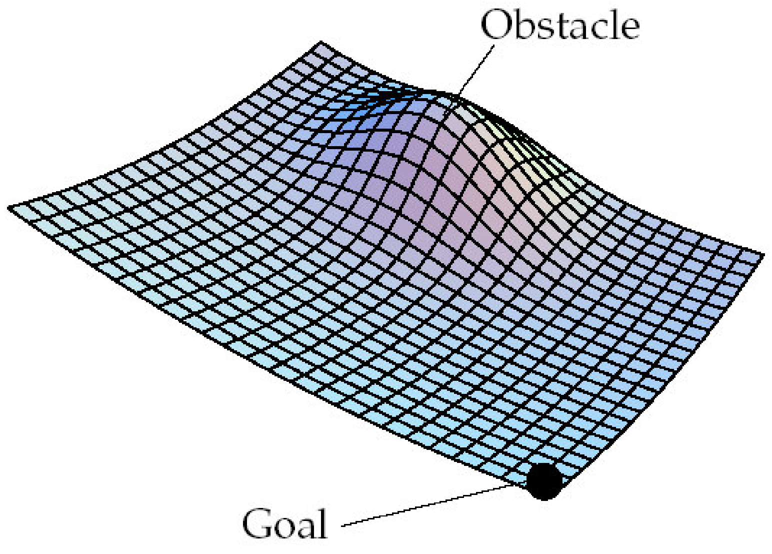


Wandering

- Adaptation to particular conditions were found (common ER problem).
- Solution: use multiple starting configurations.
- Validation in physical robots: results were improved when noise was added to simulations (more about this later...)

Navigation

- Savage et al. [117] used an EA to evolve **potential field navigation**.



pp. 52-56

Navigation

- In potential fields navigation, goals are represented by attractive potentials, and obstacles are represented by repulsive potentials. Examples:

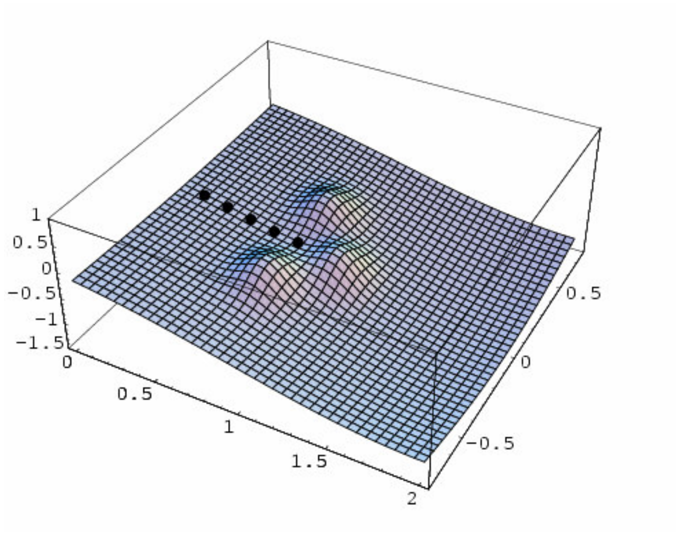
$$\Phi^{\mathrm{g}} = k_{\mathrm{g}} \left(\mathbf{x} - \mathbf{x}_{\mathrm{g}}\right)^2$$

$$\Phi^{\mathrm{o}} = k_{\mathrm{o}} e^{-\frac{(\mathbf{x}-\mathbf{x}_{\mathrm{o}})^2}{w_{\mathrm{o}}^2}},$$

Navigation

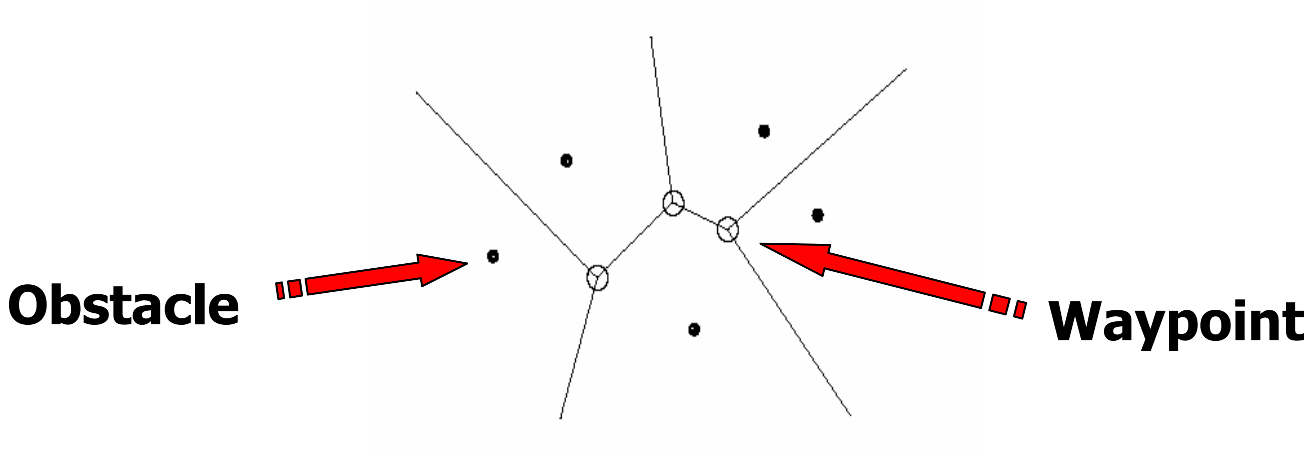
$$\hat{\mathbf{r}} = -\frac{\nabla\Phi}{|\nabla\Phi|}.$$

- The robot follows the gradient of the field
- Problem: local optima optima in the field:



Navigation

- Local optima can be circumnavigated by the introduction of waypoints, e.g. small attractive potentials.
- Problem: how should waypoints be placed.
- Possible solution: Voronoi diagrams.



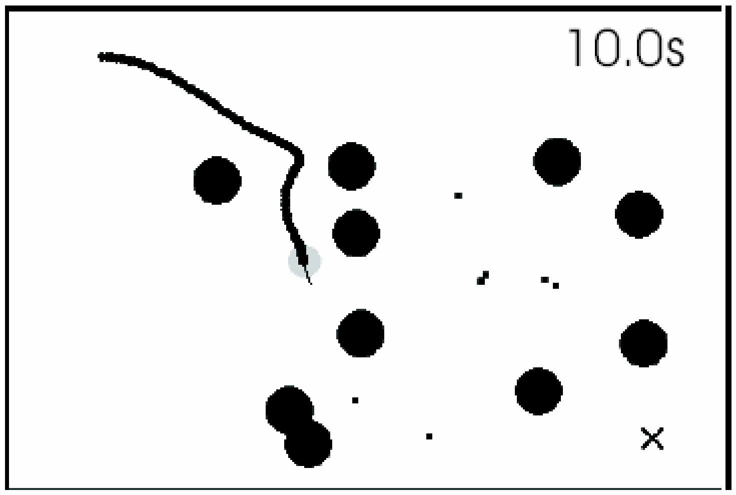
Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Navigation

- Additional problems: depth and width of potentials (goals, obstacles, waypoints), waypoint removal, robot speed.
- These parameters were evolved in [117].
- Goal: navigate from point A to point B, as fast as possible, without collisions.

Navigation

- 20 different arenas were used.



Navigation

- Fitness on a given arena:

$$f_i = \frac{T_{\max}}{T} \mathrm{e}^{-\frac{d}{D}},$$

- Combined fitness measure: two versions

$$f^{(1)} = \frac{1}{N_{\mathrm{e}}} \sum_{i=1}^{N_{\mathrm{e}}} f_i, \qquad f^{(2)} = \min_i f_i,$$

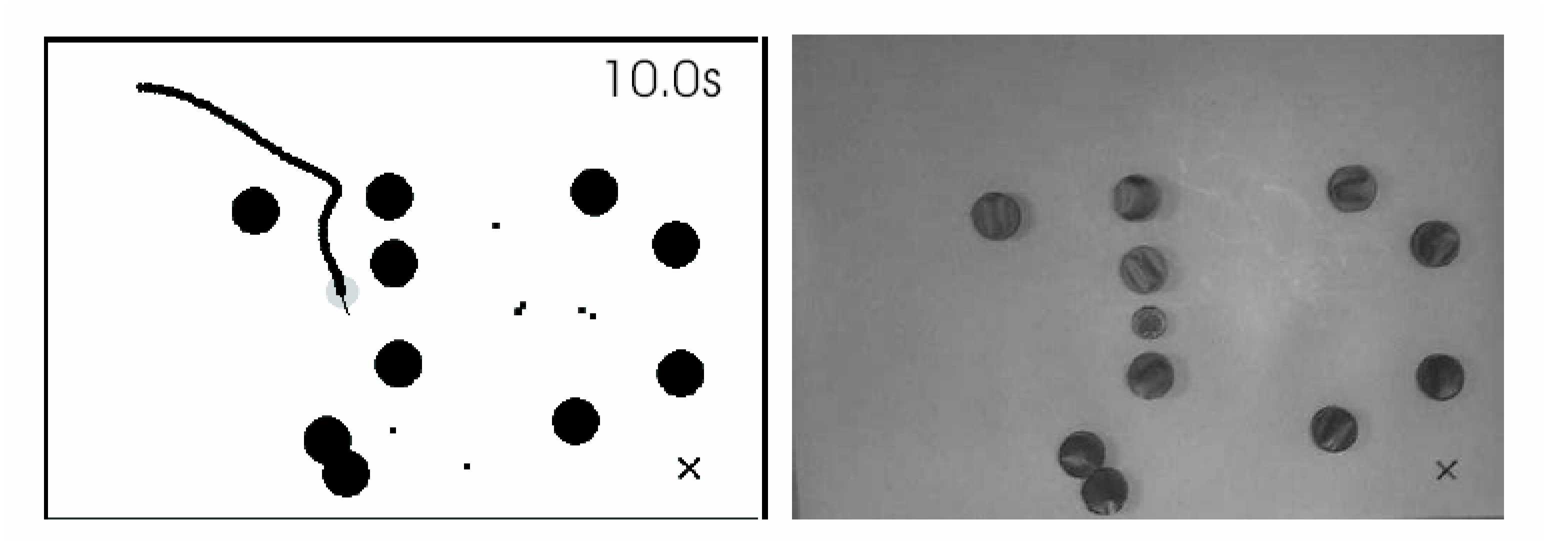
Navigation

- Results:
- Focusing on the *worst* evaluation gives more robust results.

Run #	No. of reached goals	Fitness type
1	20	f_{I}
2	18	f_{I}
3	14	f_{I}
4	19	f_{I}
5	20	f_{II}
6	20	f_{II}
7	20	f_{II}
8	20	f_{II}

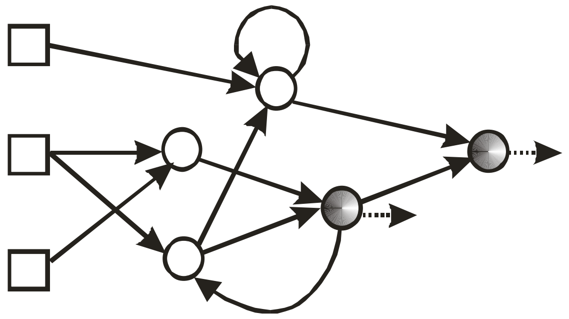
Navigation

- Validation on Khepera robot: qualitatively similar results, but problems with positioning (solution: lower speed).

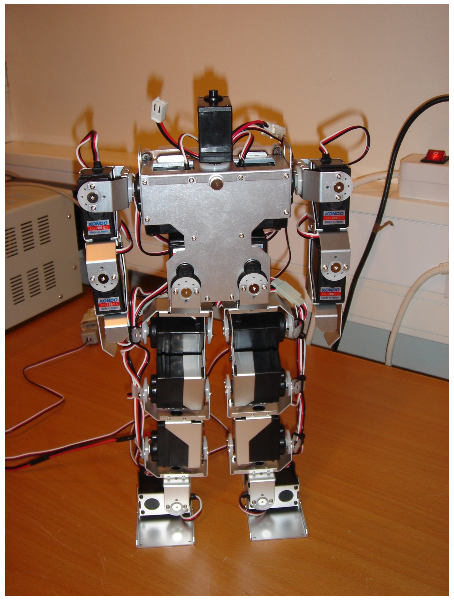


Navigation

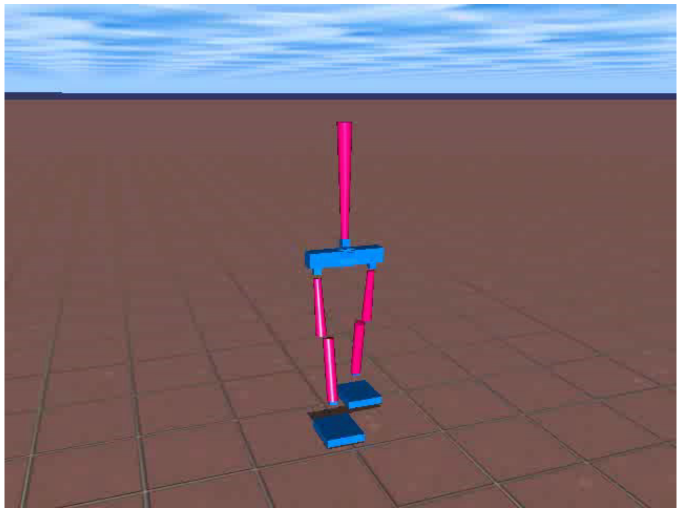
- Evolution of basic navigation, using RNNs as robotic brains: ER Simulator, v1.0
- **Note!** Strongly simplified: single, deterministic evaluation used.
- Only intended as a demo.



Humanoid robots



- A special case of autonomous robots.
- Humanoid shape =>
  - Easy adaptation to environments designed for people.
  - Capability of walking in stairs etc.
  - Easier to accept, on a social level (less intimidating etc.)
  - ...but more difficult to control (not statically balanced etc.)



Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Optimization of bipedal gaits

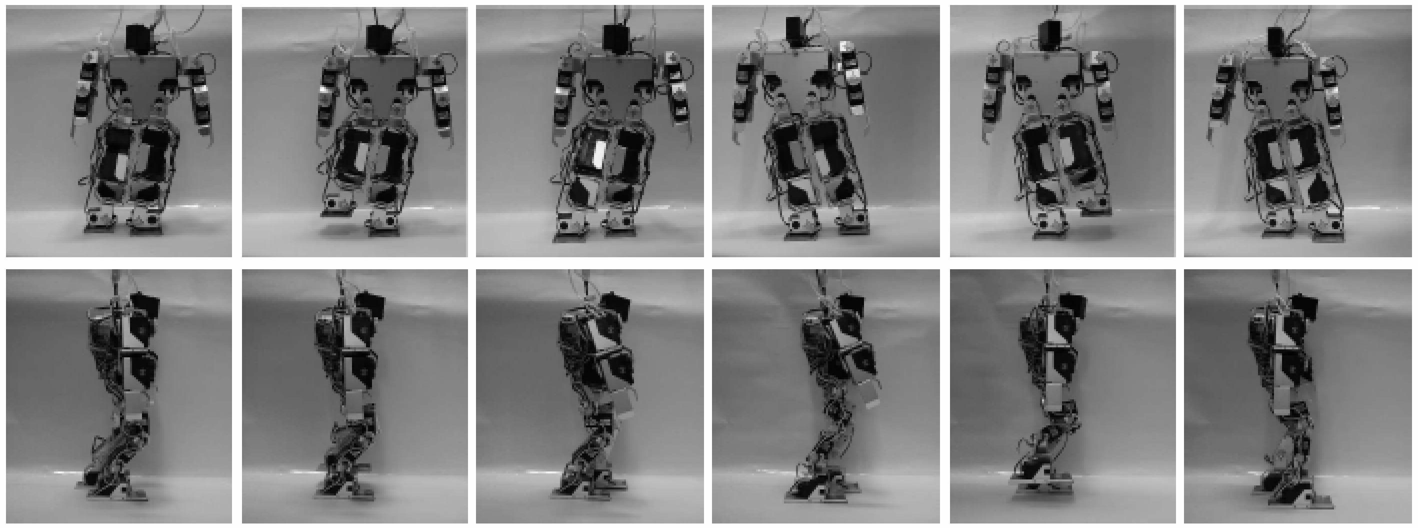


Figure 3.26: *A Kondo robot in action. This particular gait is statically stable and consists of six states, shown from the front and from the side.*

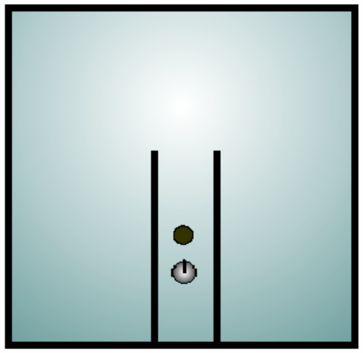
p. 83-84

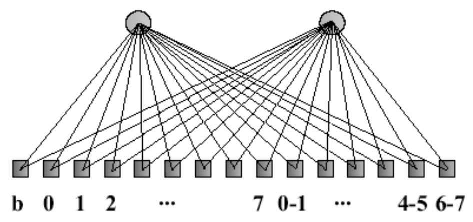
Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Box-pushing

- Sprinkhuizen-Kuyper et al. [120] considered different fitness measures for box-pushing.
- **External** vs. **internal**: the latter dependent only on information available to the robot.
- **Local** vs. **Global**: the latter takes into account only the difference between the final state and the initial state.

Box-pushing





- Best results were obtained with a global external fitness measure.

$$f_{\mathrm{GE}} = d(B_T, B_0) - \frac{1}{2} d(B_T, R_T)$$

- Thus, the EA did best when it was least constrained.

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Motion of a robotic arm

- An example involving a stationary robot

p. 62-63

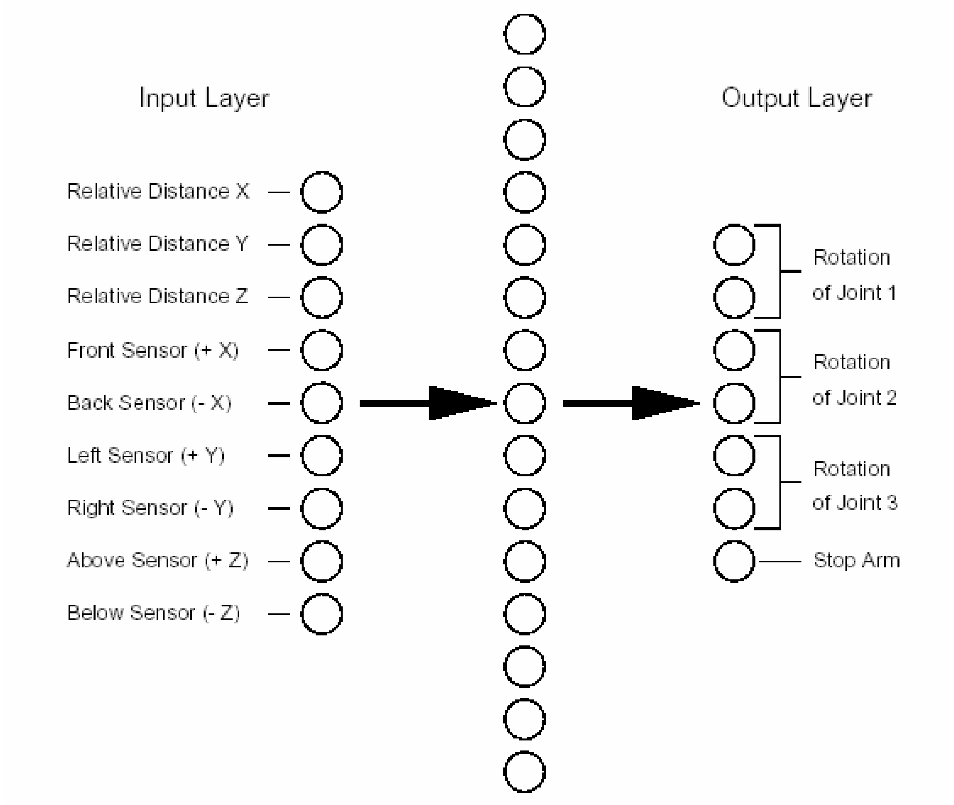
Motion of a robotic arm

- Moriarty and Miikkulainen [91] evolved obstacle avoidance for an OSCAR-6 robotic arm.
- **Supervised training methods** (such as backpropagation) are difficult to apply in an arena with obstacles (no evident error signal for each move).

Motion of a robotic arm

- Moriarty and Miikkulainen evolved two neural networks, a primary network and a secondary network.
- The primary network handled general approach to the target position, whereas the secondary network handled fine-tuned movements close to the target.

Motion of a robotic arm



Motion of a robotic arm

- The evaluation used a global, external fitness measure (percentage of original distance covered).
- 400 training cases (obstacle configuations) and 50 validation cases were used.
- The results obtained were within industry standards (< 1cm deviation).

Evolution of complex behaviors and behavioral organization (selection)

- In the architectures used in BBR, a complete robotic brain is built up from a set of simple behaviors.
- The goal is to arrive at robots capable of complex behaviors.

pp. 63-79

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

- Essentially three different approaches have been tested:

  (1) Evolution of complex behavior without explicit behavioral selection.

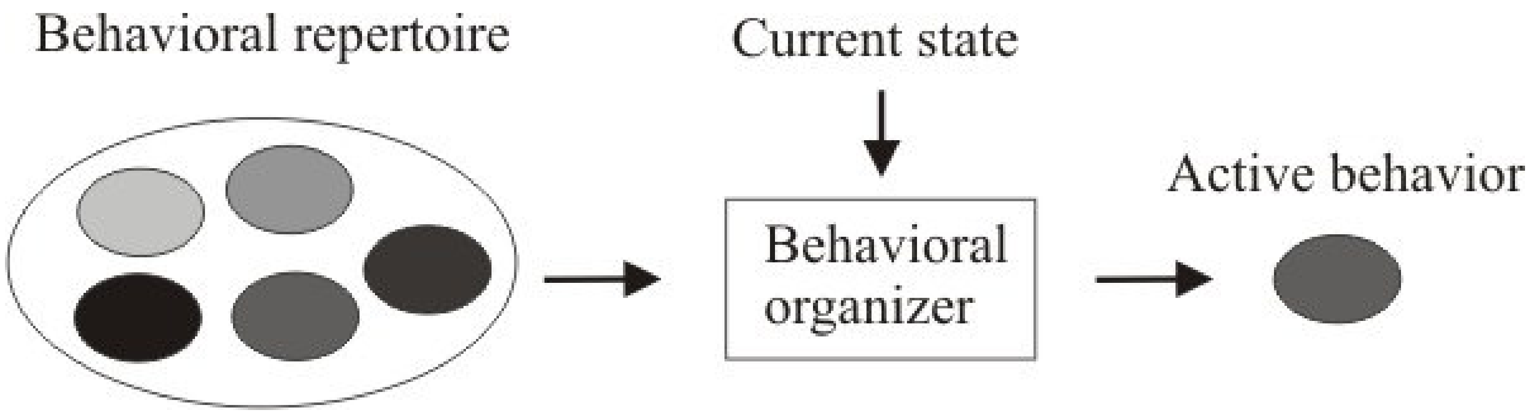
  (2) Evolution of a behavioral repertoire of simple behaviors, which are then combined using some (non-evolutionary) method for behavioral organization.

  (3) Evolving the behavioral organizer.

Behavioral organization

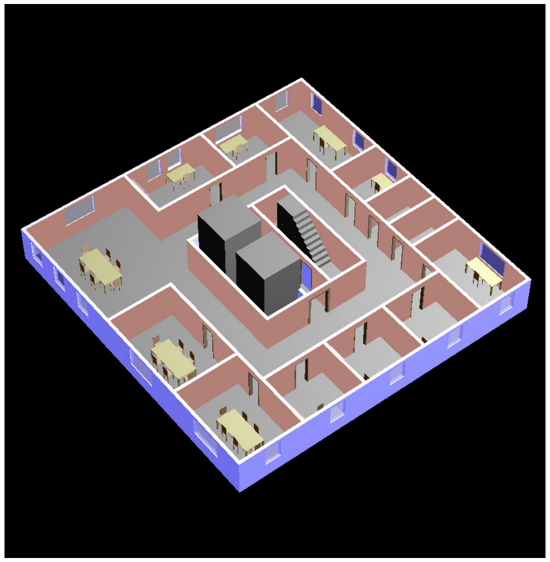
- The issue of **behavioral organization** (also known as **behavioral** selection or **action selection** has been studied extensively in ethology.
- Two main methods: **arbitration methods** (only one behavior active at each instant), and **cooperative methods**.
- Examples: Subsumption, DAMN, Potential fields navigation etc.

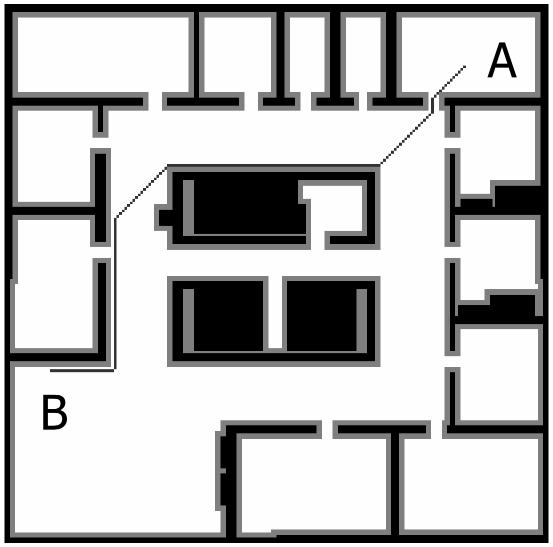
The problem of behavior organization



Behavior selection

- Example: a delivery task
  - Need to reach the goal as fast as possible, but...
  - ...must avoid collisions.
  - ...must avoid running out of energy.
  - ...must select an appropriate path – perhaps the shortest path is blocked?
  - ...what if there are *two* goal positions? The order in which they ought to be reached may depend on events that occur during navigation.
  - etc. etc.





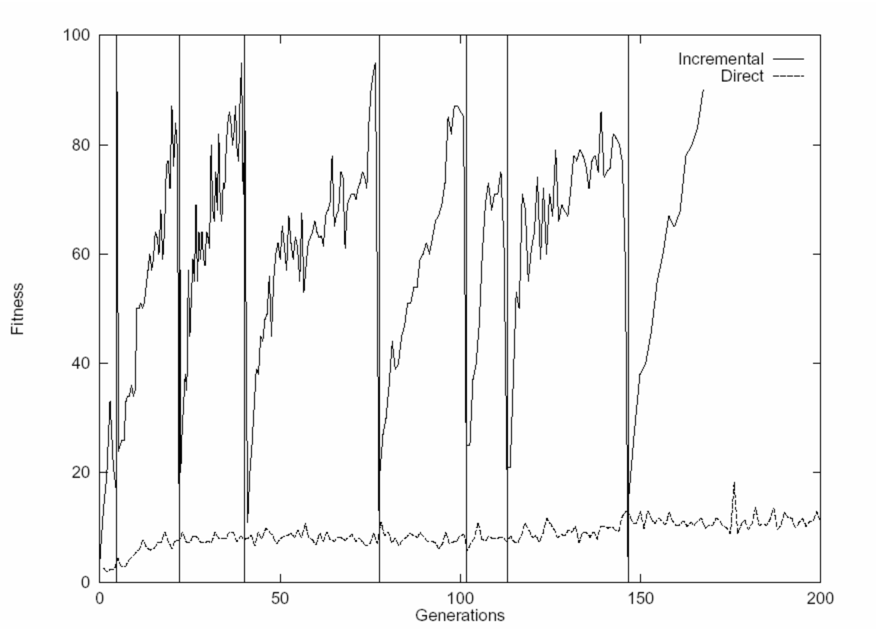
Evolution of complex behavior without explicit behavioral selection.

- Incremental evolution (Gomez and Miikkulainen [44]): make the task gradually more complex.
- Applied to the problem of prey capture.

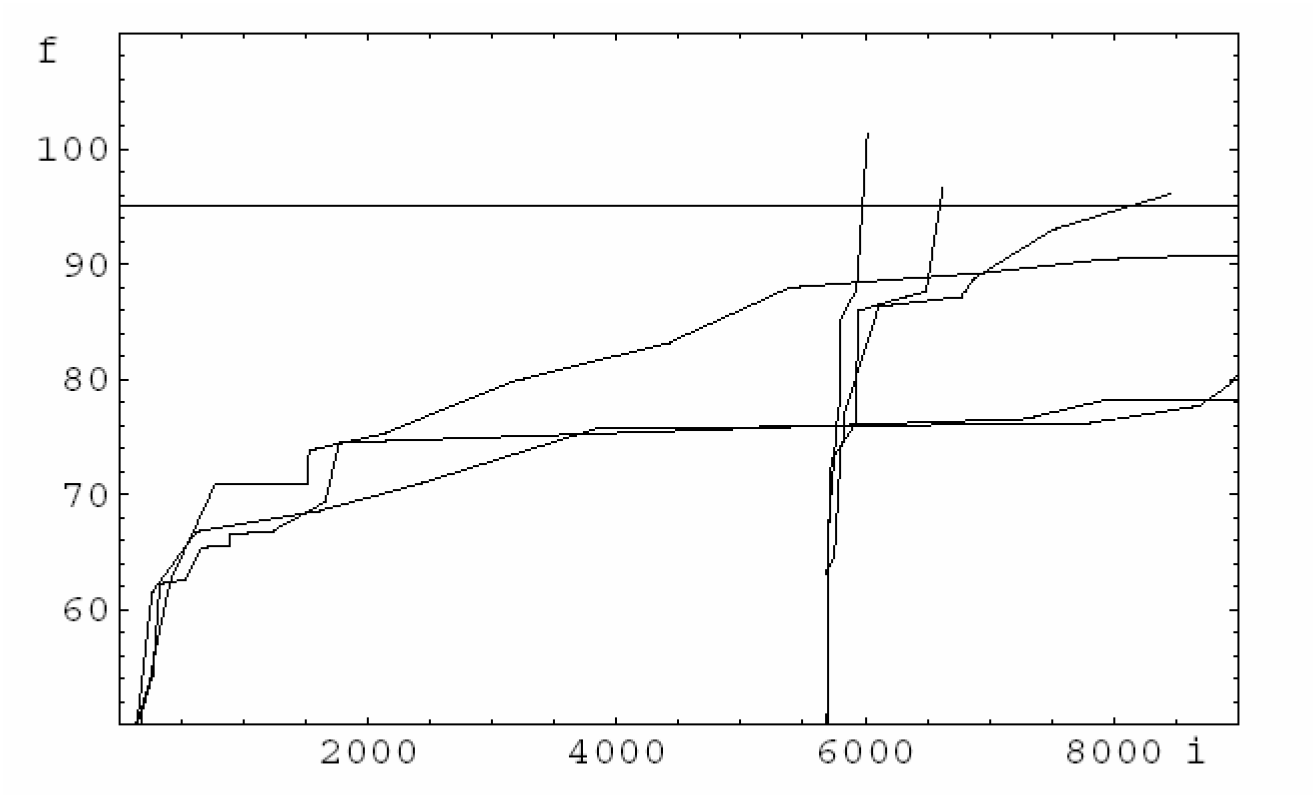
$$E^{s}_{n} \longleftarrow \text{Speed of prey}$$

$$\text{Head start of prey} \nearrow n$$

- $E^{1.0}_4$ was difficult to generate by direct evolution.
- However, the sequence $E^{0.0}_0$, $E^{0.2}_0$ ... $E^{1.0}_4$ was readily evolved.



- Similar results were obtained by Wahde and Sandholt [137], who evolved robotic brains for floor sweeping and obstacle avoidance.



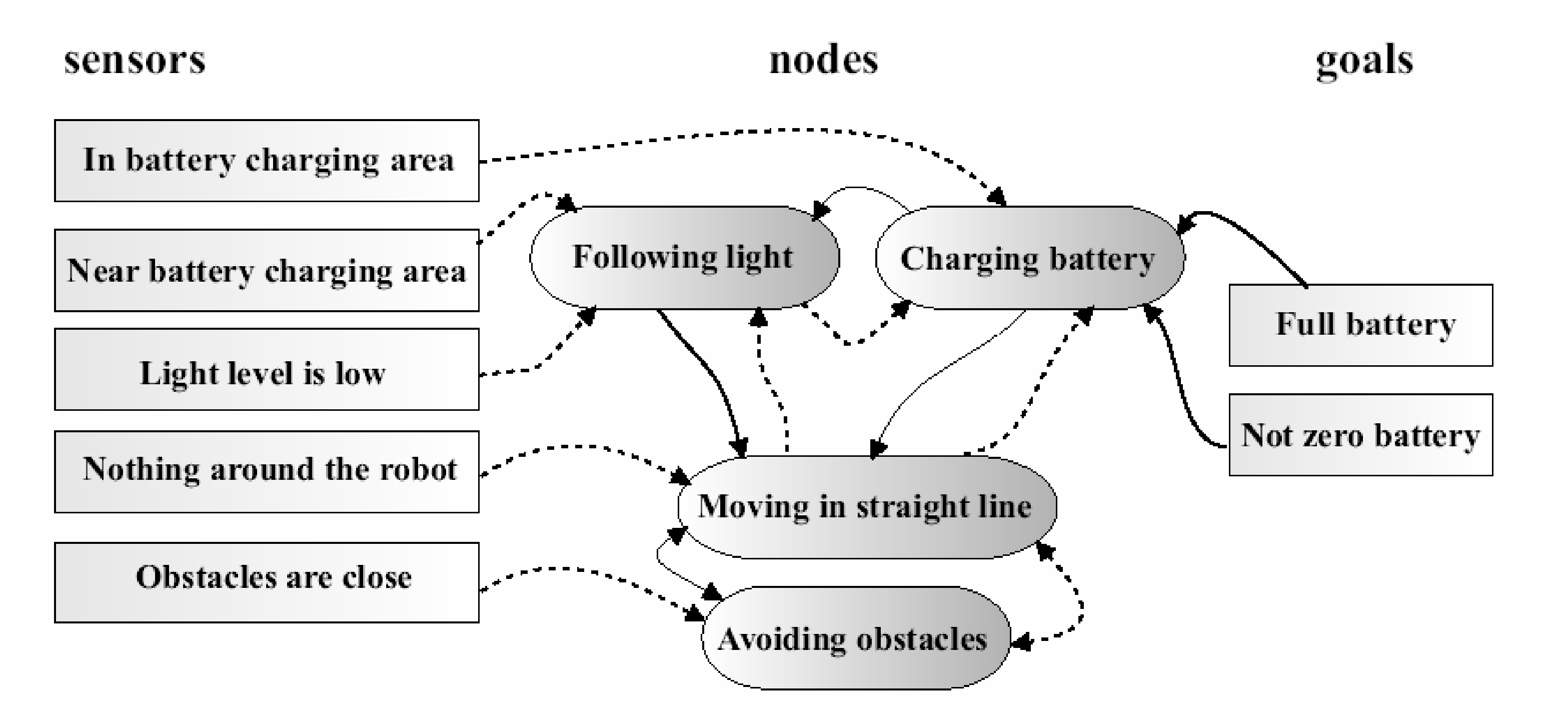
- Floreano and Mondada [34] evolved homing navigation (to an energy source, indicated by a light).
- It was found that good results could be found by making the fitness measure *simpler*, thus allowing the EA to explore more freely.
- From [34] it is also evident that analyzing robotic brains in the form of ANNs is very difficult.

Evolution of a behavioral repertoire

- Kim and Cho [65] evolved four behaviors for a task similar to that considered by Floreano and Mondada.

- Behaviors: *Move in a straight line*, *follow light*, *avoid obstacles*, and *charge batteries.*

- Behavioral organization was achieved using Maes' [78] method of **activation networks**.



- The robotic brain obtained by Kim and Cho [65] was easier to interpret than an ANN.

- However, the method of activation networks (and most other methods for behavioral organization) requires the user to set many parameters *by hand*. Very difficult!

- Alternative: Evolve the behavioral organization system.

Issues

- Achieving complex overall behavior.
- Avoiding manual parameter-tuning.
- Making use of the ethological background of BBR.
  - Animals are equipped with extremely fine-tuned systems for behavior selection.
  - There exists a sequence from the simplest organisms (e.g. bacteria) up to more complex ones.

- These considerations lead us to the **Utility function (UF) method** for behavior selection...

The concept of utility

- **Utility** (formalized by von Neumann and Morgenstern in 1943) has been used frequently in the theory of rational decision-making.
- Given a choice between different actions A_i, a rational agent will selection the action associated with highest utility:

$$A_{i_{\mathrm{sel}}} = \operatorname{argmax} U(A_i)$$

- Thus, essentially, utility acts as a common currency in decision-making.
- The problem is to set appropriate utility functions.

Rational agents and utility

- **Rational agents** display **transitivity of choice**.
- $A_1 > A_2$, $A_2 > A_3 \Rightarrow A_1 > A_3$. (Order of preference).
- Rational agents select actions *as if* they were maximizing a quantity we can call utility:

$$A_{i_{\mathrm{sel}}} = \mathrm{argmax}\; U(A_i)$$

A biological example

- The motion of E. Coli bacteria (towards a food source) can be modelled using two simple behaviors:
  - B_1 Straight-line navigation
  - B_2 Tumbling
- Let B_1 be activated if $U_1 > U_2 = 0$.
- $U_1(t) = V(t) – X(t)$, where $X(t)$ is the food concentration and

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} + aV(t) = bX(t).$$

- In this case, setting values of *a* and *b* fully specifies U_1 and U_2.

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

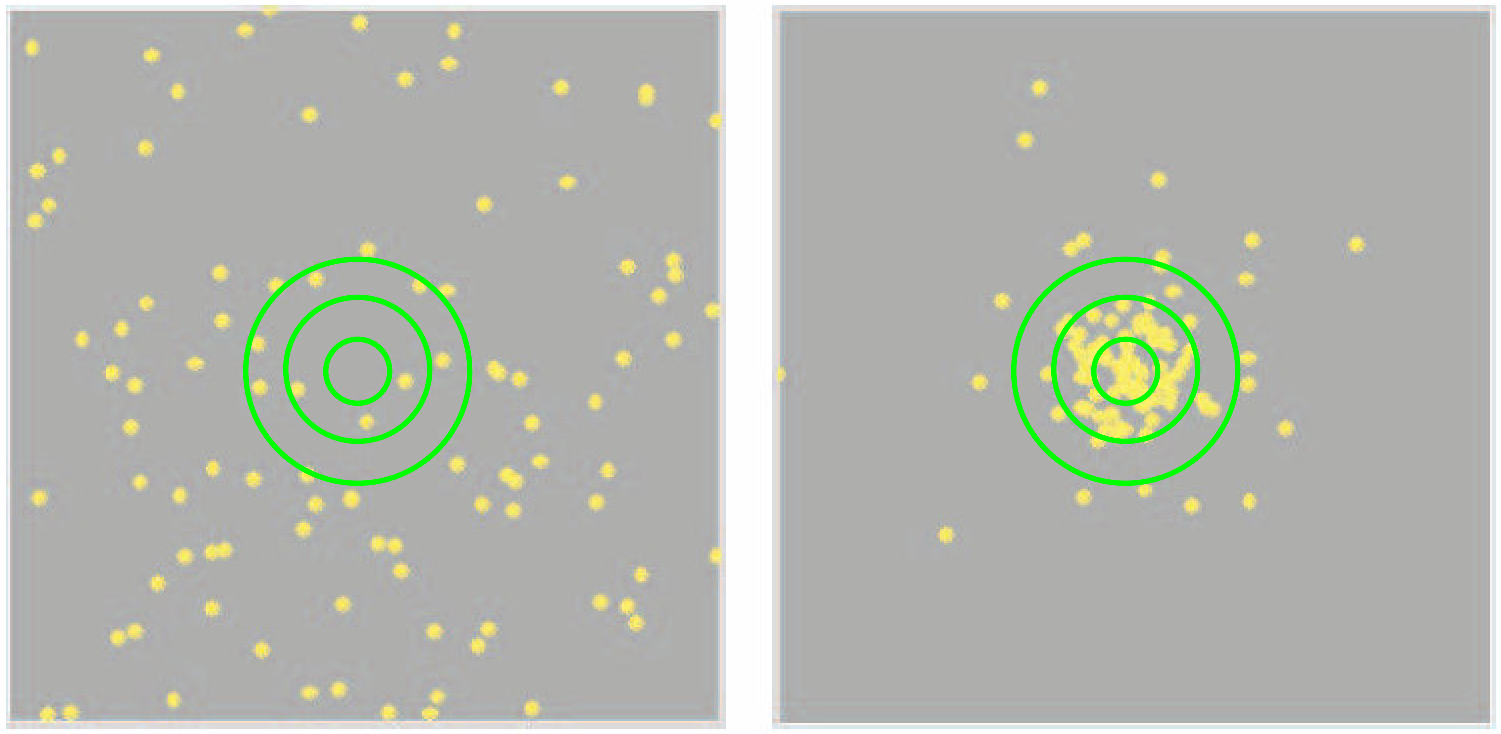


Figure 2.2: *The motion of simulated* E. Coli *bacteria based on the behavior switch defined above. 100 bacteria were simulated, and the parameters a and b were both equal to 1. The attractant had a gaussian distribution, with its peak at the center of the image. The threshold was set to 0. The left panel shows the initial distribution of bacteria, and the right panel shows the distribution after 10 simulated seconds, using a time step of 0.01.*

Evolving behavioral organization: The utility function (UF) method

- Main research topic in my group.
- General reference: Wahde [131] (and this tutorial).
- The utility function method is an arbitration method, based on evolutionary optimization of utility functions.

Example: floor-sweeping

- Robot with two behaviors: *floor-sweeping* and *battery charging*.
- Only *floor-sweeping* gives fitness increase, but *battery charging* is still sometimes necessary – its utility rises as the battery energy falls.

The Utility function method

- A method for behavior selection in autonomous robots, based on optimization of utility functions.
- In the UF method, each behavior B_i is associated with a utility function U_i.
- Behaviors are divided into **task behaviors** (that yield fitness increase) and **auxiliary behaviors** (no fitness increase).

The Utility function method

- The utility functions depend on the state of the robot, defined by external variables (such as sensor readings) and internal variables (corresponding to signalling substances, such as hormones). Example:

$$U_i(s,p) = a_{i,00} + a_{i,10}s + a_{i,01}p + a_{i,20}s^2 + a_{i,11}sp + a_{i,02}p^2,$$

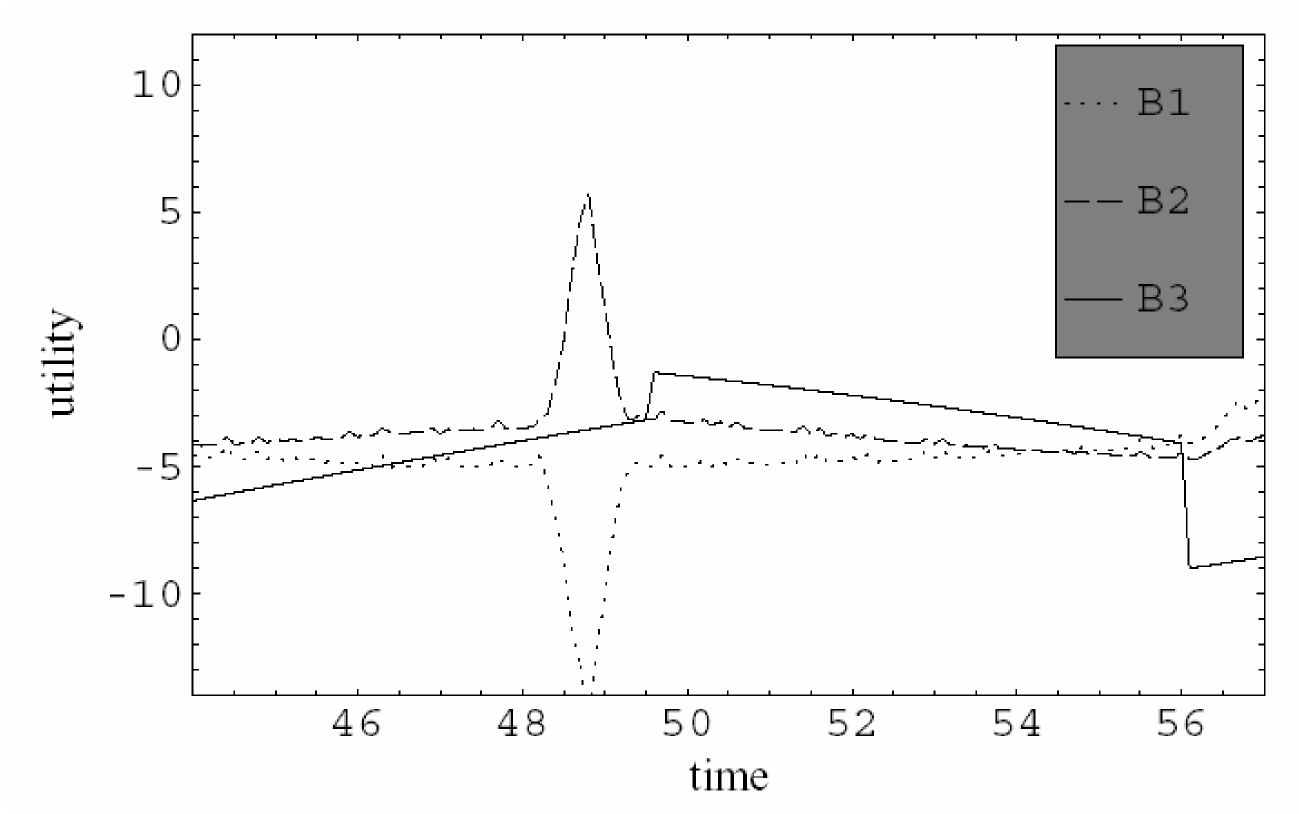
- Expansions (Fourier or polynomial) are normally used.

The Utility function method

- Behavior selection is simple:
  - v1.0 (for navigation tasks): the behavior with highest utility is active.
  - v2.0 (under development): all processes (behaviors) with U > 0 are active. In case of conflicts (e.g. several behaviors attempting to access the same motor) the behavior with highest utility is given command.
- The problem, of course, is to determine the utility functions!

$$U_i(s,p) = a_{i,00} + a_{i,10}s + a_{i,01}p + a_{i,20}s^2 + a_{i,11}sp + a_{i,02}p^2,$$

The utility function method

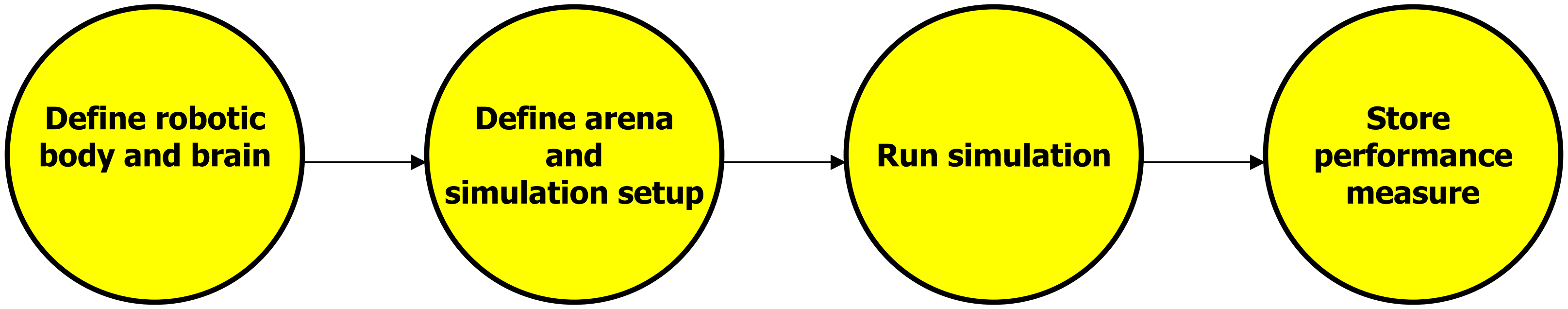


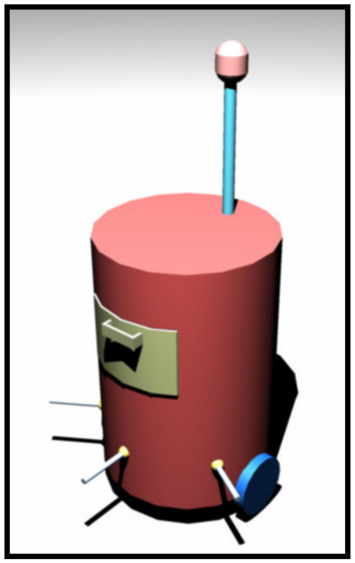
Setting utility functions

- In the UF method, utility functions are optimized using simulations.
- The UF method has been implemented in Delphi.
- The implementation (called UFLibrary) comprises (at present) around 25,000 lines of code.

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Simulation procedure


Define robotic body and brain
Define arena and simulation setup
Run simulation
Store performance measure

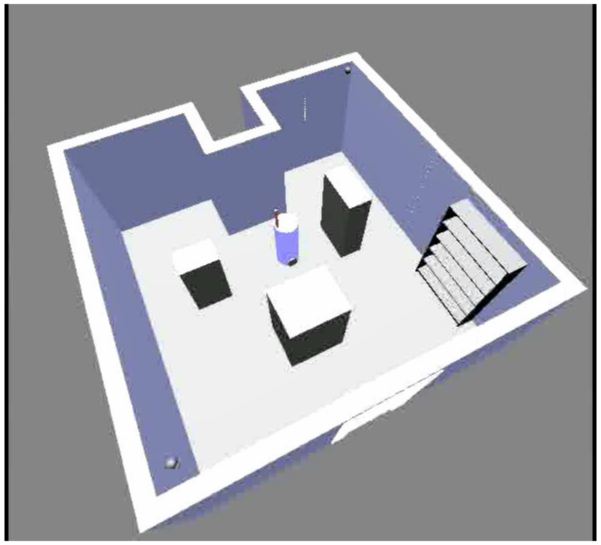


Behaviors (fixed)

- navigation
- obstacle avoidance
- etc. etc.

Utility functions

$$U_i(s,p) = a_{i,00} + a_{i,10}s + \dots$$



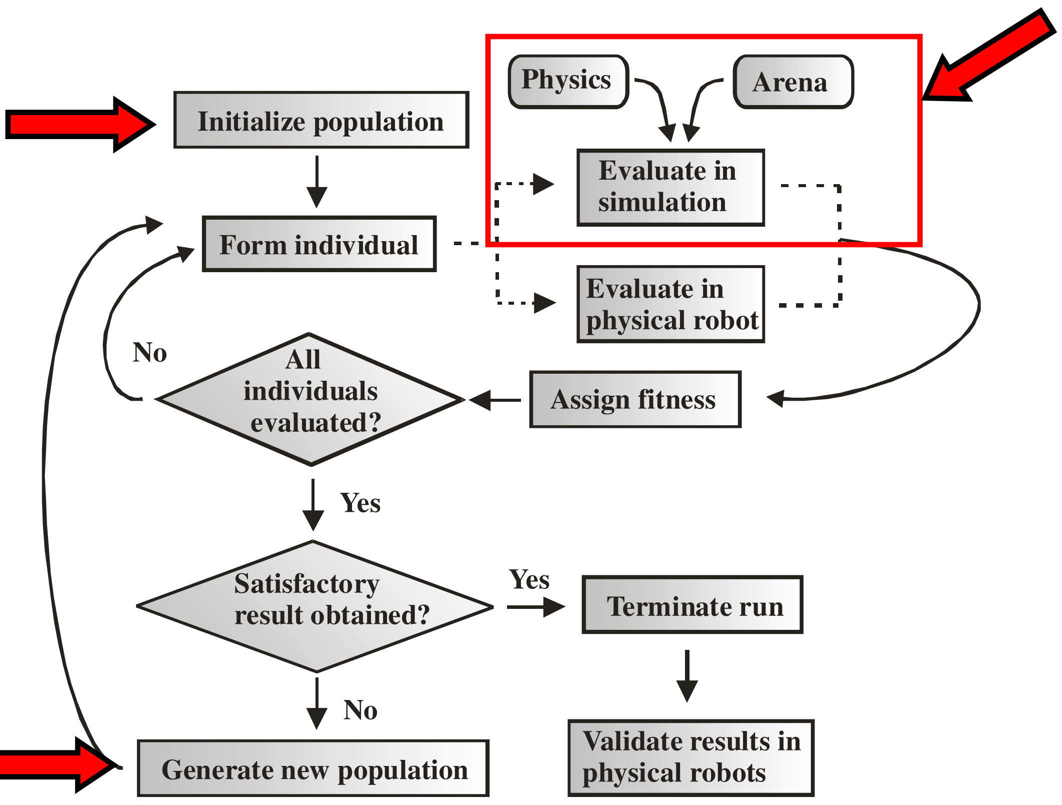
Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Optimization procedure

Random initial utility functions

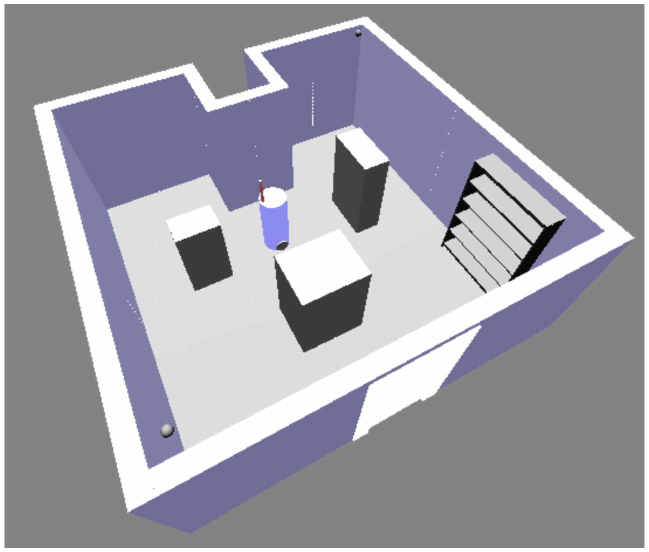
Selection, crossover, mutation

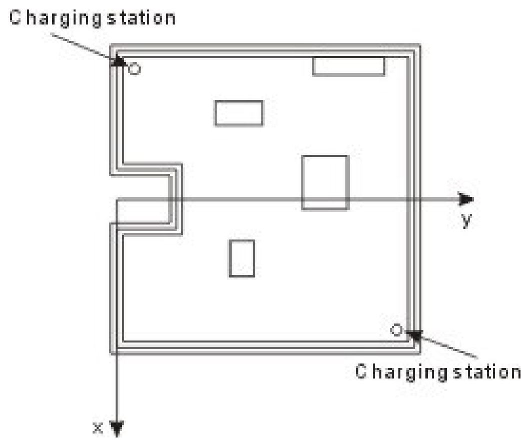
Previous slide!



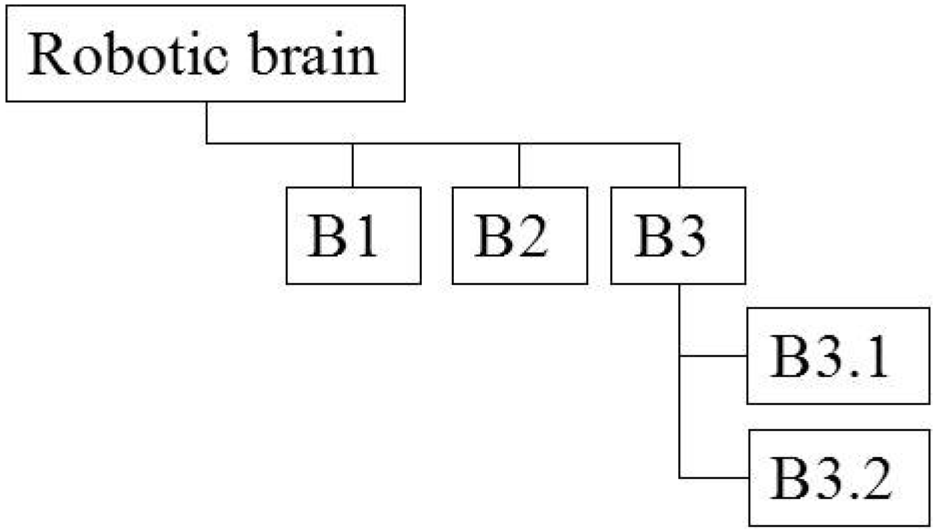
Usage example 1: simple exploration

Task: to explore the arena, while avoiding collisions *and* keeping the battery non-empty.



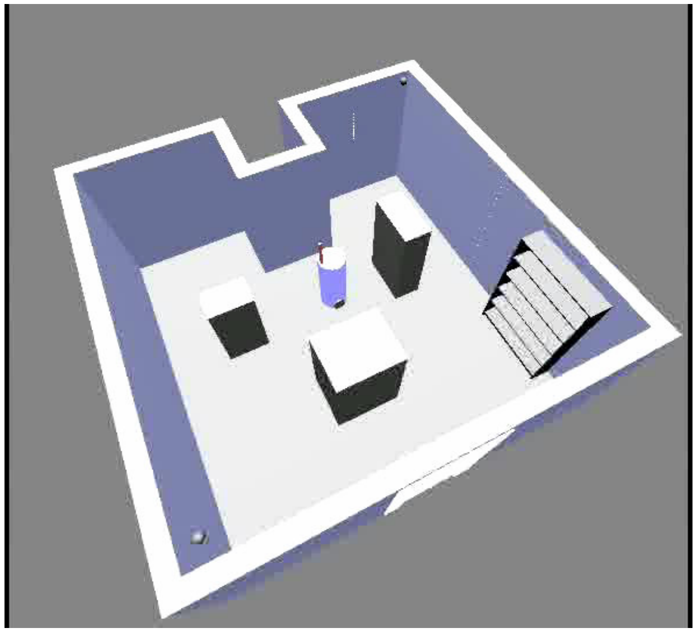


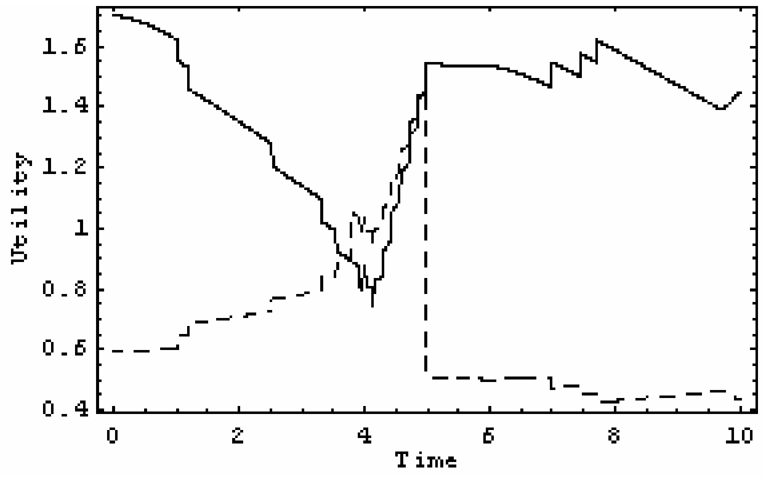
Behavior hierarchy



B1 = navigation, B2 = obstacle avoidance,
B3 = energy maintenance,
B3.1 = locate charging station, B3.2 = charging

Evolved utility functions



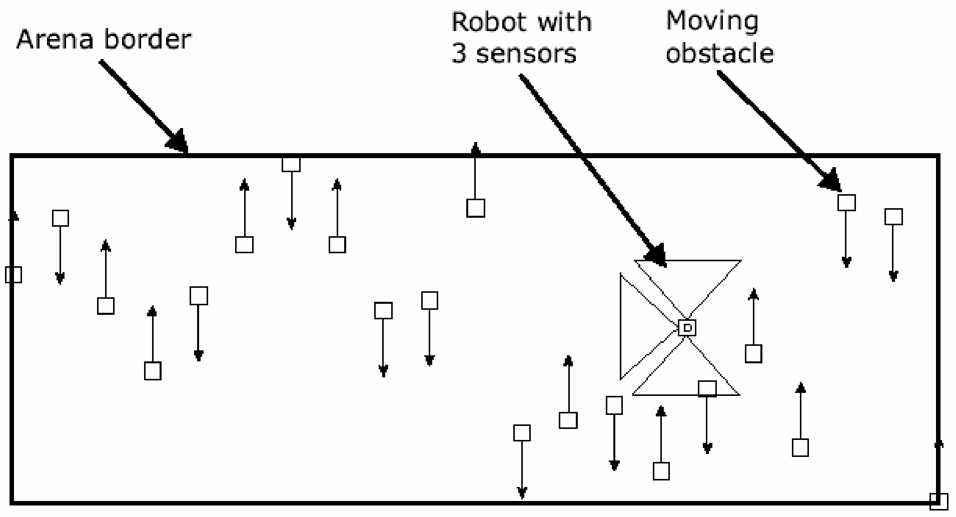


UFlib demo

UFLibDemo.exe

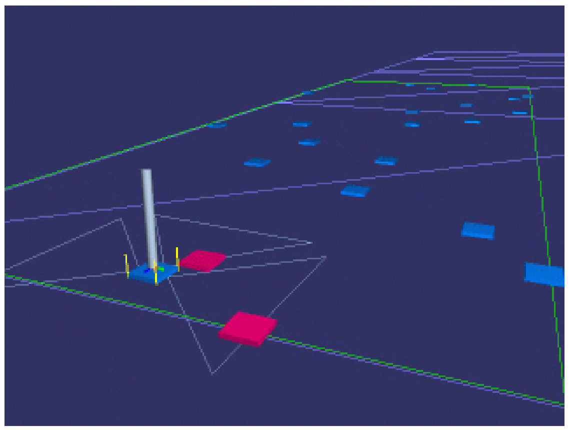
Usage example 2: locomotion

- Pettersson and Wahde [107]* evolved a behavioral organization system for a locomotion task involving four behaviors:



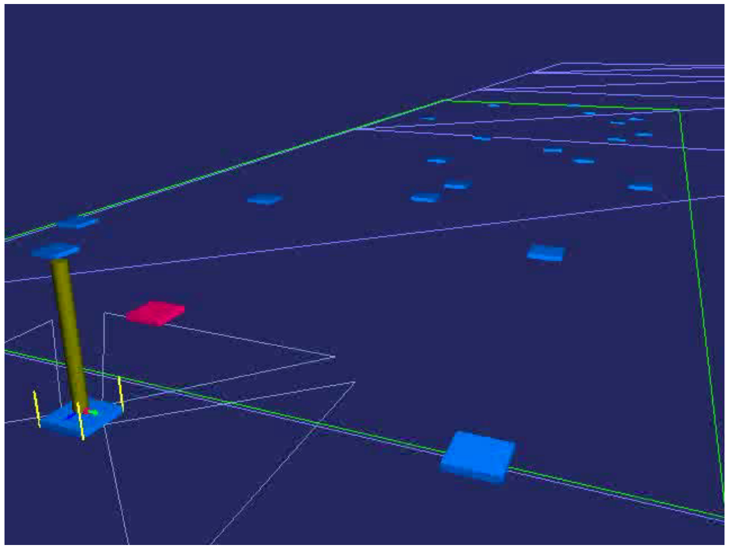
* Pettersson, J. and **Wahde**, M. *Application of the utility function method for behavioral organization in a locomotion task*, IEEE Trans. Ev. Comp., **9(5)**, pp. 506-521, 2005

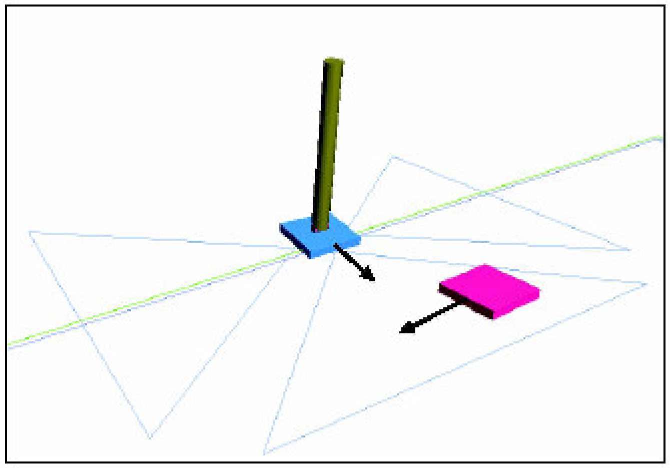
- Behaviors: *move forward*, *move backward*, *stop*, and *charge batteries.*
- In this particular case, the behaviors were also *evolved* (not a requirement).
- Each behavior consisted of an RNN.
- Fitness: distance moved in the initial forward direction.

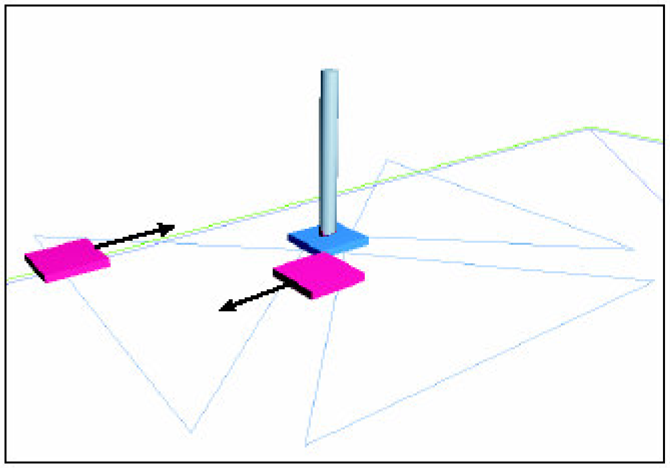


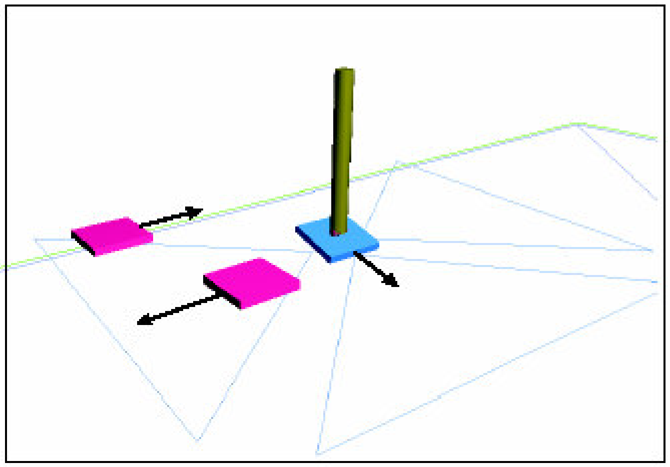
Results

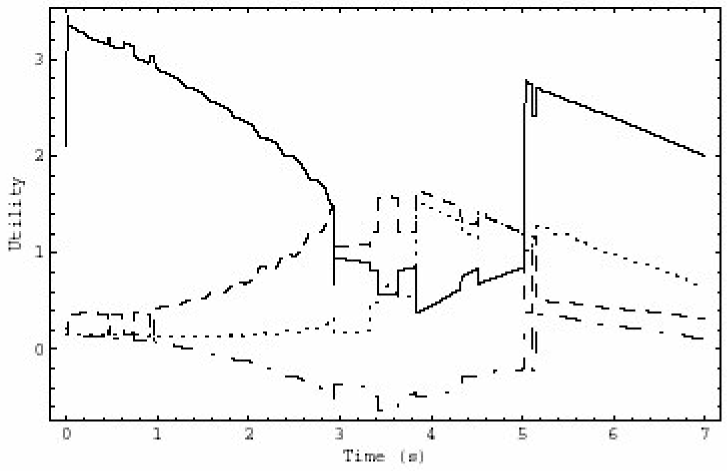
- Successful behavioral organizers were evolved (limitation: quality of behaviors).







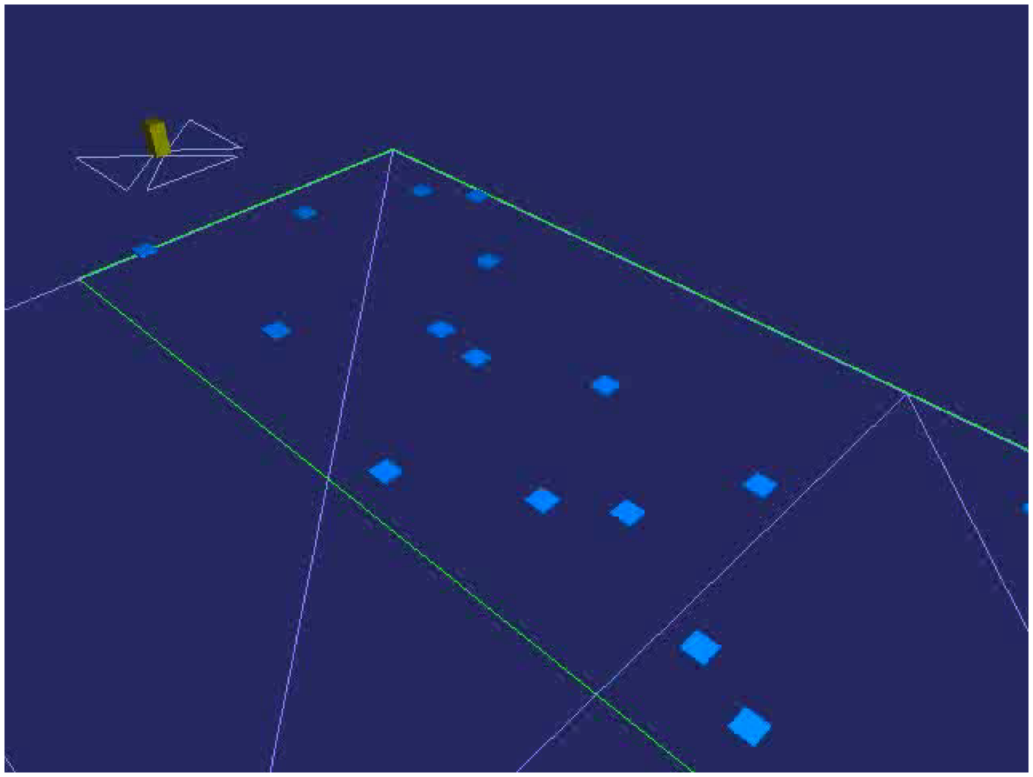




Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

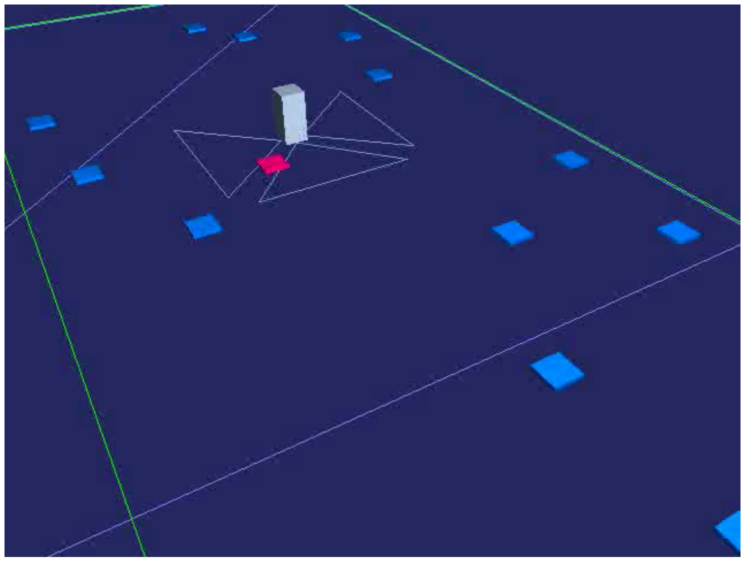
Results

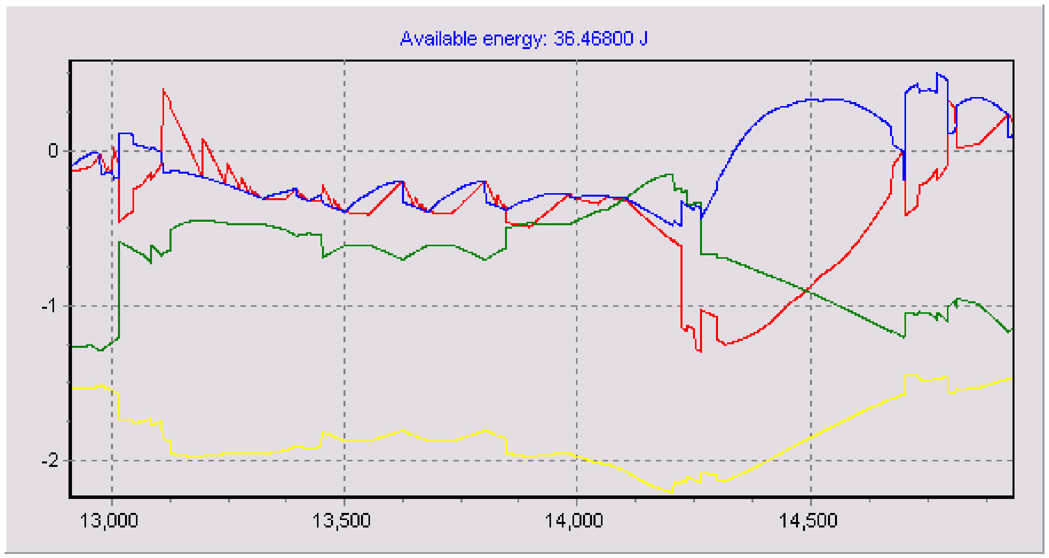
- A simplified model was tried as well:



Results

- Backward motion when needed (despite *negative* fitness contribution).



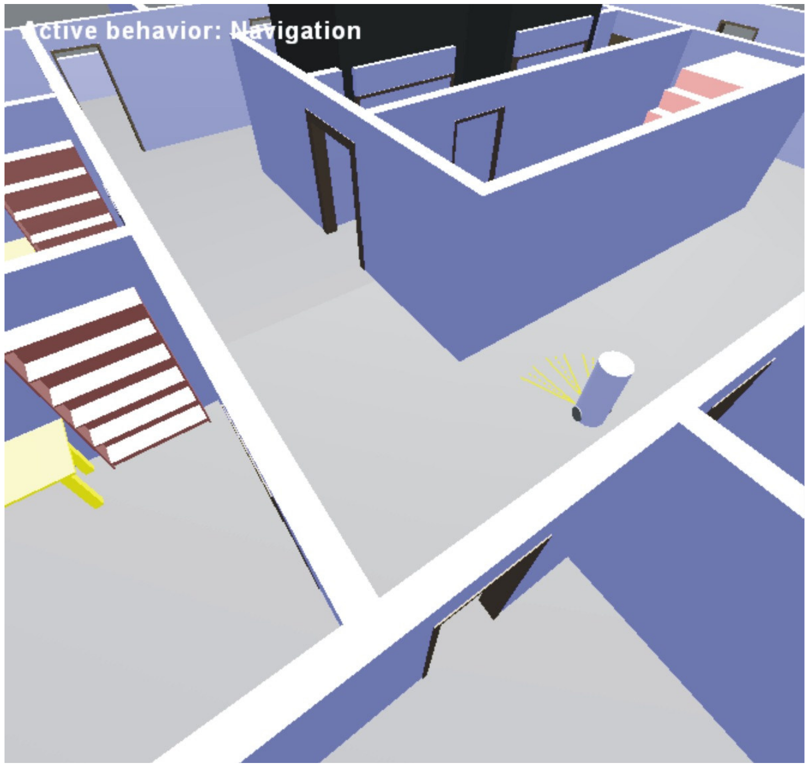


Current projects

- The transportation robot
  - A robot for transportation and delivery in hospitals, factories, supermarkets etc.
- The tour guide robot
  - A robot that will function as a tour guide in museums.
  - This project was started recently

Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

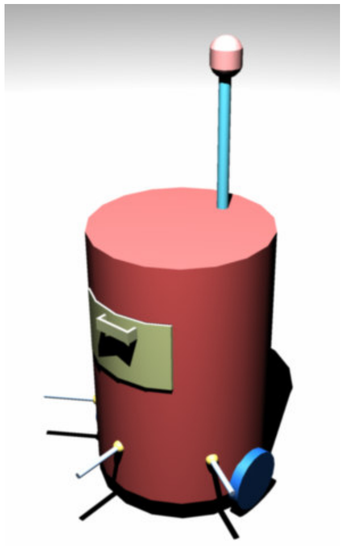
Transportation robot



Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Transportation robot

- An application of the UF method
- Objective: to deliver objects reliably between two arbitrary points in a given arena.
- Sensory modalities:
  - touch sensors
  - laser range finder (4.0m range)
  - wheel encoders

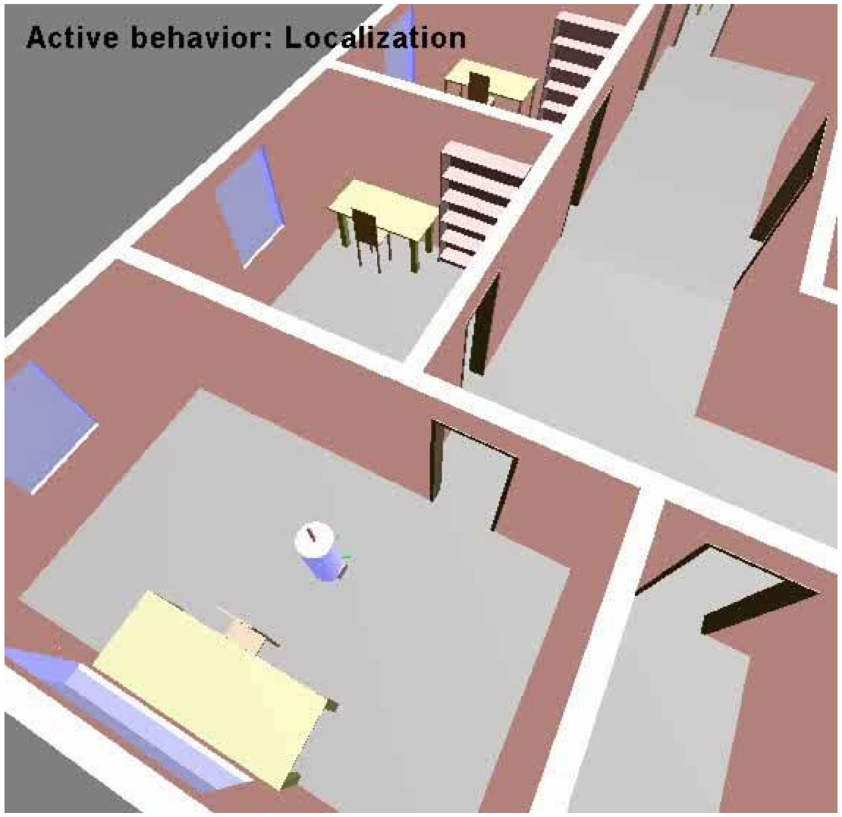


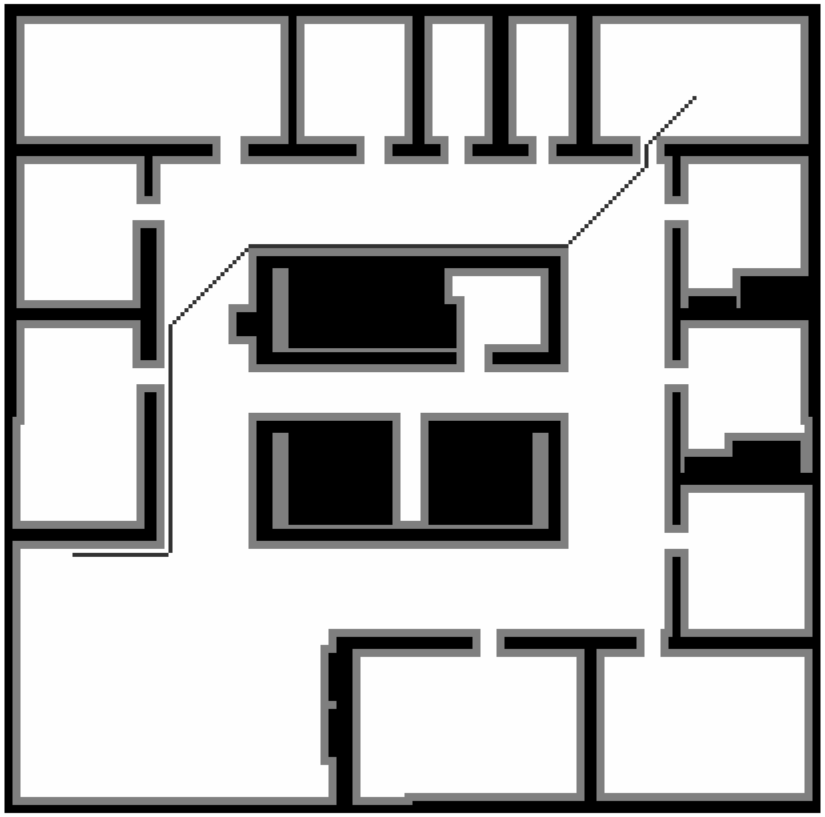
An early prototype..

Similar robots

- TUG, Univ. Of Maryland
- HOSPI (Matsushita)

Transportation robot

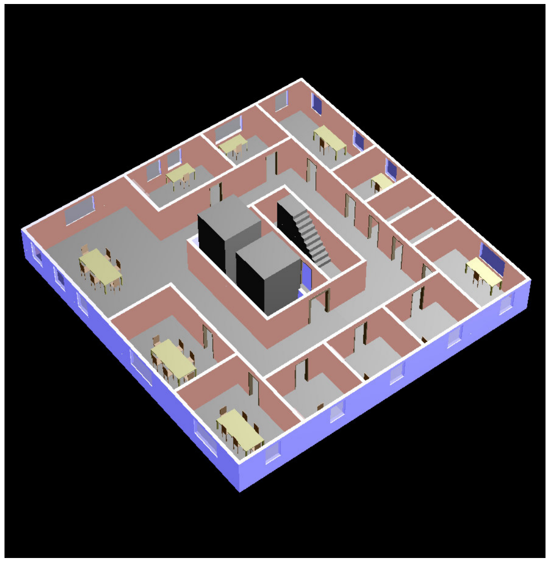


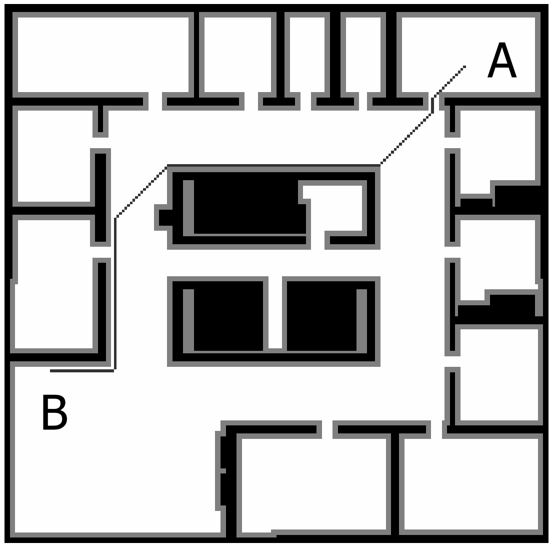


Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Transportation robot

- Example: a delivery task
  - Need to reach the goal as fast as possible, but...
  - ...must avoid collisions.
  - ...must avoid running out of energy.
  - ...must select an appropriate path – perhaps the shortest path is blocked?
  - ...what if there are *two* goal positions? The order in which they ought to be reached may depend on events that occur during navigation.
  - etc. etc.





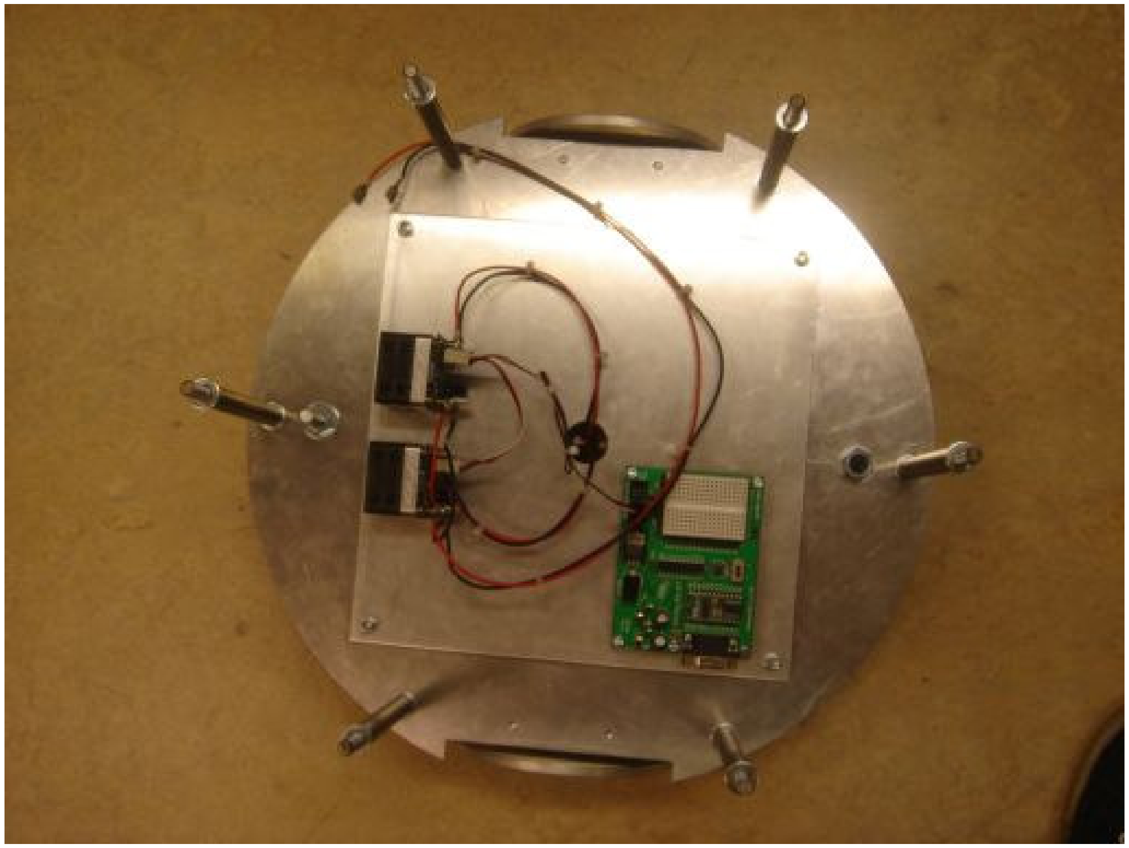
The transportation robot

- Results from simulations

Active behavior: PathNavigation

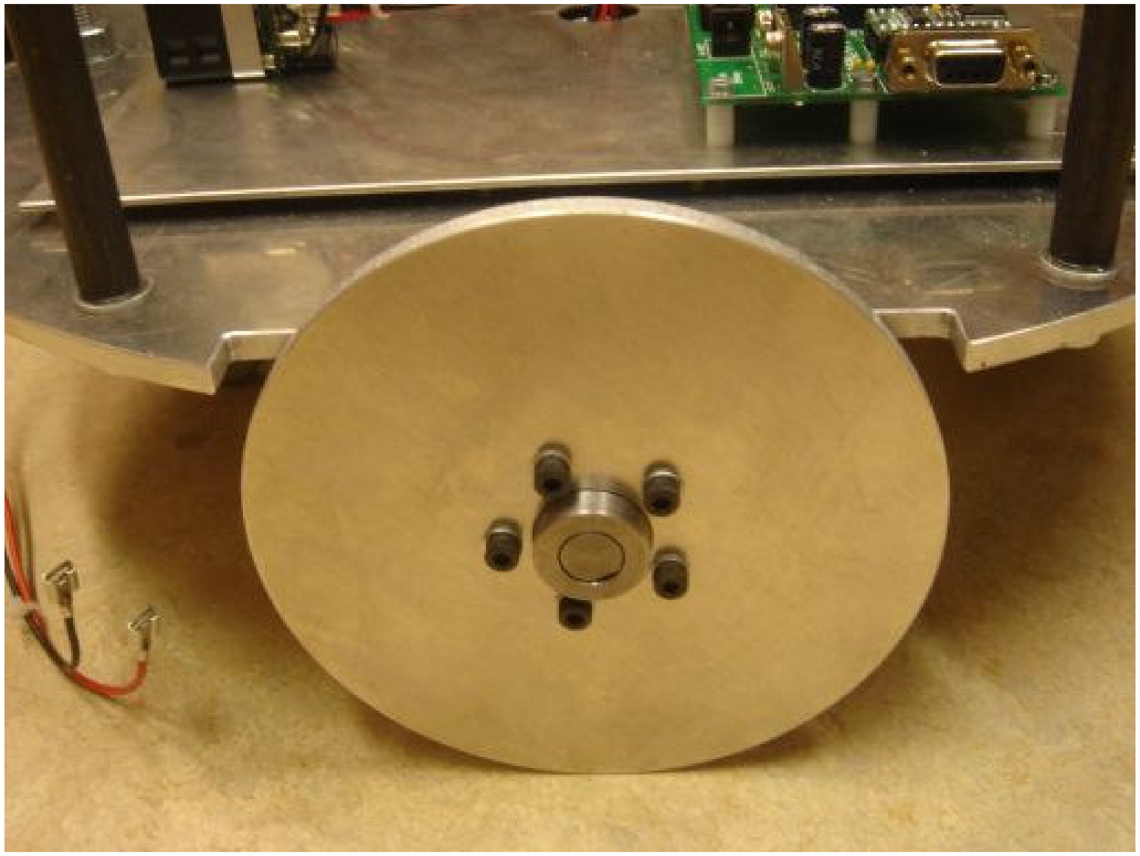
Active behavior: PathNavigation

Prototype (under development)



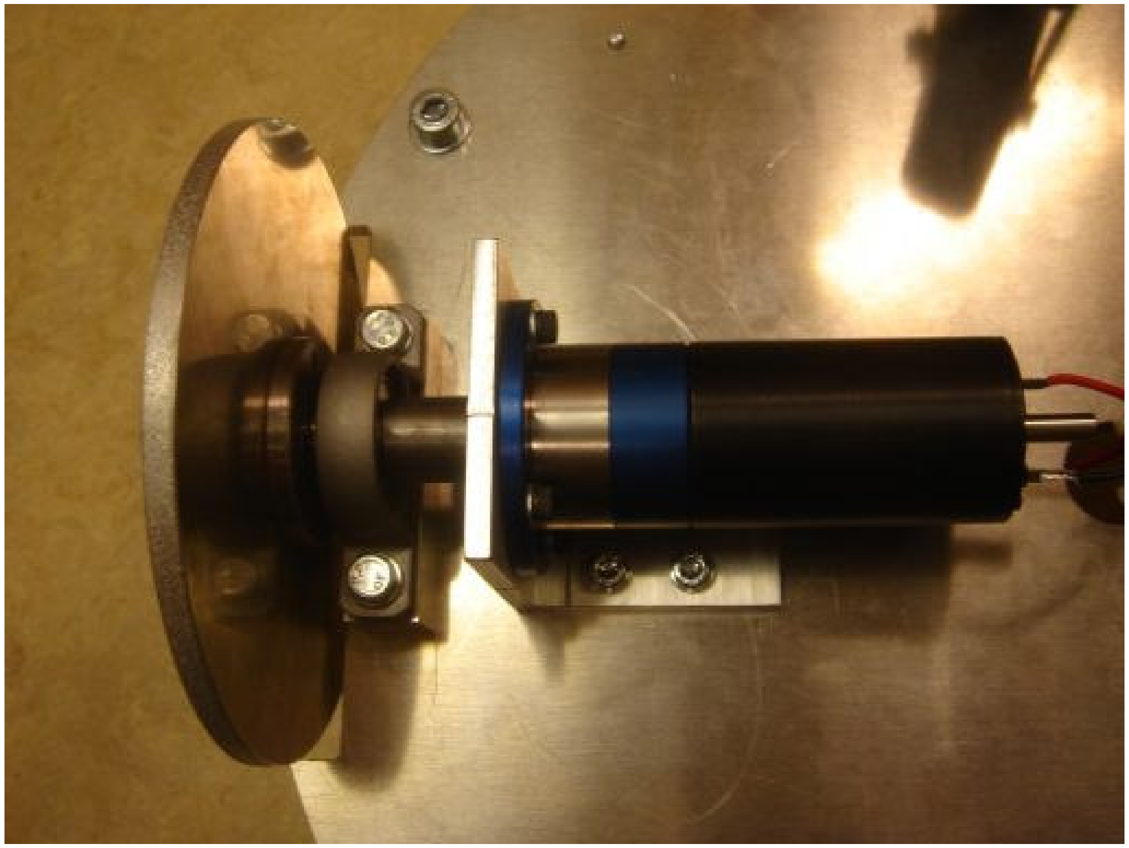
Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Prototype (under development)



Mattias Wahde, PhD, associate professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se www: www.me.chalmers.se/~mwahde

Prototype (under development)



Main points: (UF method)

Behavioral organization is one of the main obstacles in the struggle to achieve complex autonomous robots.

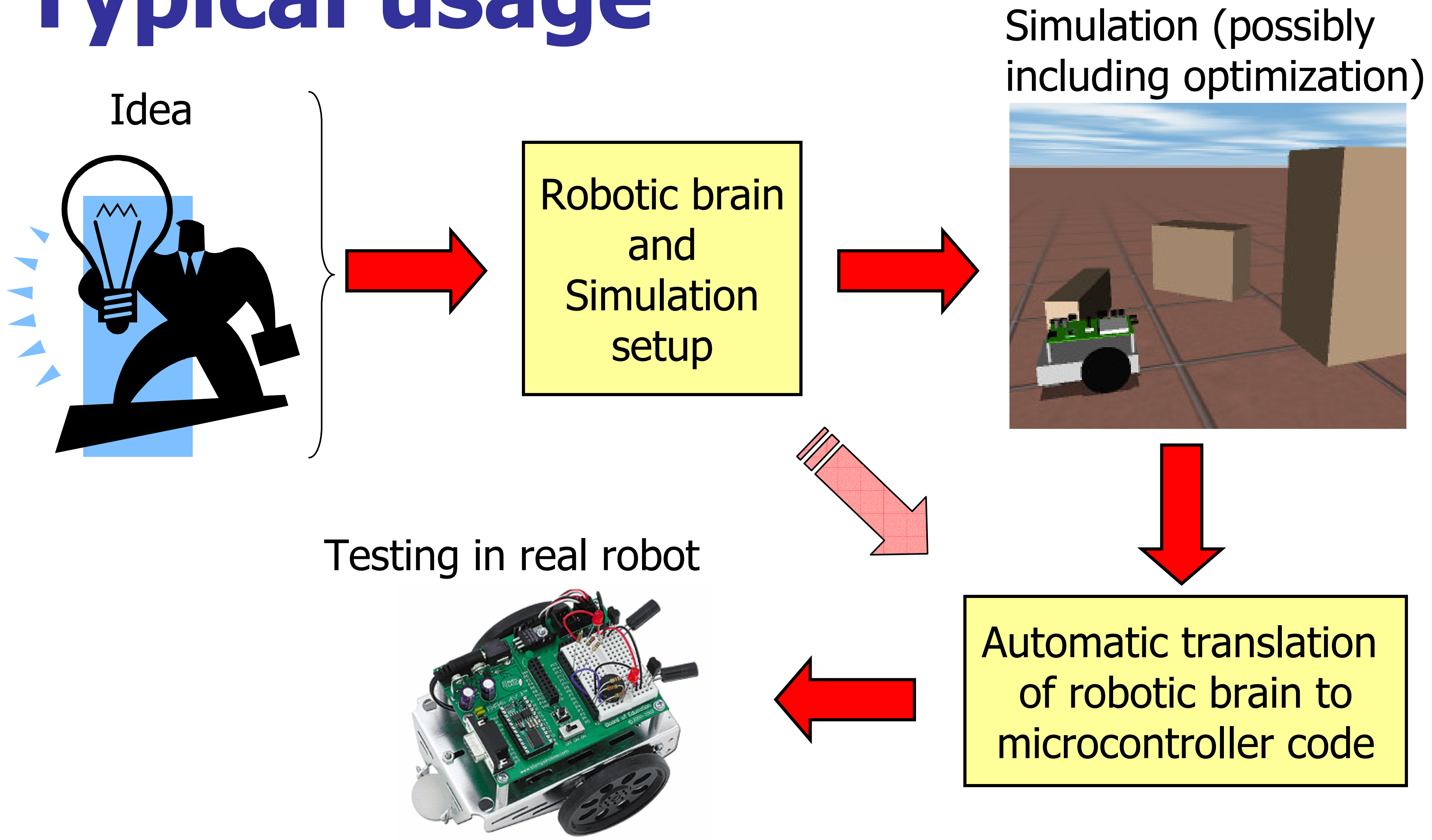
With the utility function method, we are able to generate complex behavioral organizers without hand-coding.

The first version of the utility function method was essentially limited to navigation behaviors (involving only motor behaviors). Currently a new version of the UF method (v2.0) is being developed. In this version, not only motor behaviors but also purely cognitive processes will be allowed.

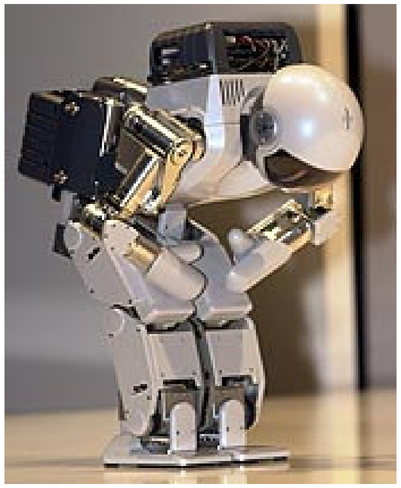
Work in progress: UF, v2.0

- This method comprises three main ideas:
  - Rapid definition of behaviors, within a single framework.
  - Accurate simulation of robots, as well as optimization of behavior selection (and, possibly, individual behaviors).
  - Direct translation of simulation results to microcontroller code.

Typical usage


Idea
Robotic brain and Simulation setup
Simulation (possibly including optimization)
Automatic translation of robotic brain to microcontroller code
Testing in real robot

Thank you very much for your attention!



- Contact information, see below.
- Home page for the Adaptive systems research group:

  http://www.me.chalmers.se/~mwahde/AdaptiveSystems.html

- Regularly updated list of conferences:

  http://www.me.chalmers.se/~mwahde/ConferenceList.php