ALPEN-ADRIA UNIVERSITÄT KLAGENFURT

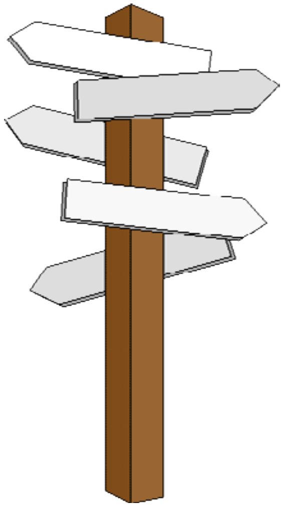
Lakeside Labs

Evolutionary Design of Self-Organizing Systems

Lakeside Research Days 2010
July 14, 2009

Wilfried Elmenreich and István Fehérvári
Mobile Systems Group/Lakeside Labs
Institute for Networked and Embedded Systems
Alpen-Adria Universität Klagenfurt

Overview



- Research problem
- Possible approaches
- The evolutionary approach
- A tool for evolving SOS
- Application examples
- Challenges

Research Problem: Design of Emergence

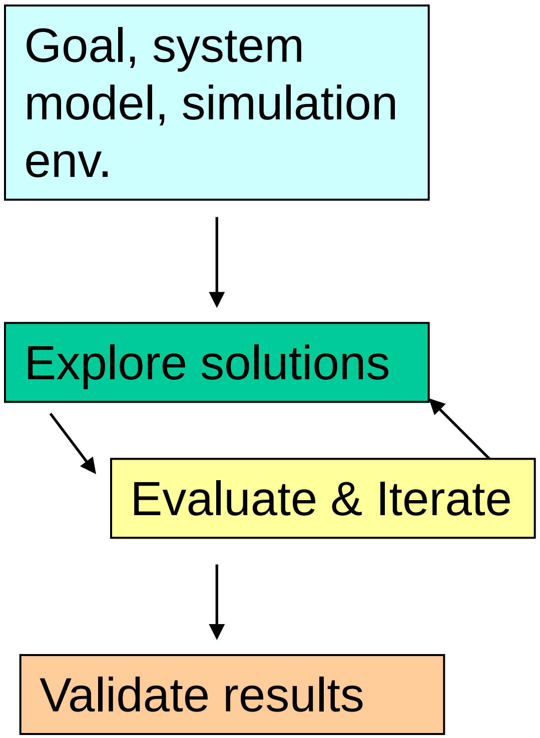
- The good news: after defining the agents, there might be a simple interaction rule that does the task, eg.,
  - bring a system into synchronization
  - coordinate cells to emerge a pattern
  - let robots play soccer
- Difficult to predict behavior for a given ruleset
  - Complex systems often behave counter-intuitive behavior (experiment on slime mold simulation by Mitch Resnick, MIT)
  - Even after the simulation had been shown, the effect of a paramter change was mispredicted
- *How to design local rules achieving the desired global properties?*

Possible Approaches

- Design rules manually
  – use notion of „friction“ as utility function (cf. C. Gershenson, Design and Control of Self-Organizing Systems, 2007)
- Derive local rules from a „Laplacian Demon“
  – Auer, Wüchner, DeMeer, A Method to Derive Local Interaction Strategies for Improving Cooperation in Self-Organizing Systems)
- *Derive rules using an automated, self-organizing search algorithm*

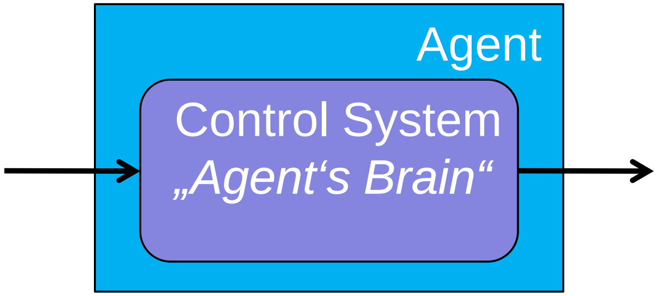
The Evolutionary Approach

- Methodology:
  - Simulation of target system as playground
  - Evolvable model of local behavior (eg., (fuzzy) rules, ANN)
  - Define goal via fitness function (eg., maximize throughput in a network)
  - Run evolutionary algorithm to derive local rules that fulfill the given goal



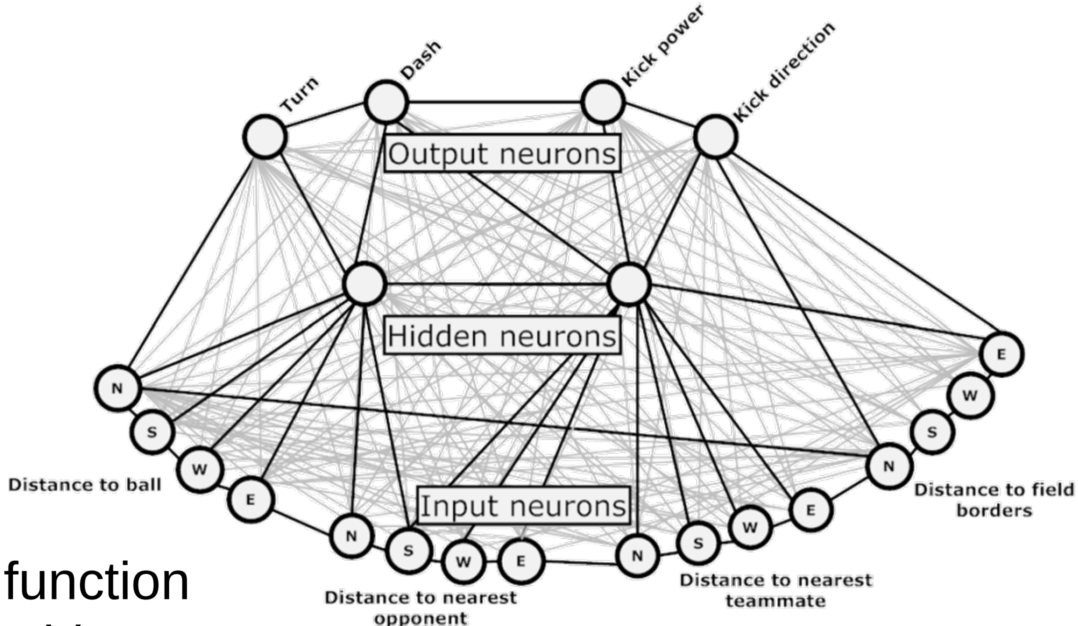
Control system to be evolved

- Controls the agents of the SOS
- Processes inputs (from sensors) and produces output (to actuators)
- Must be evolvable
  - Mutation
  - Recombination



- You cannot easily do this with an algorithm represented in C code…

Artificial Neural Networks

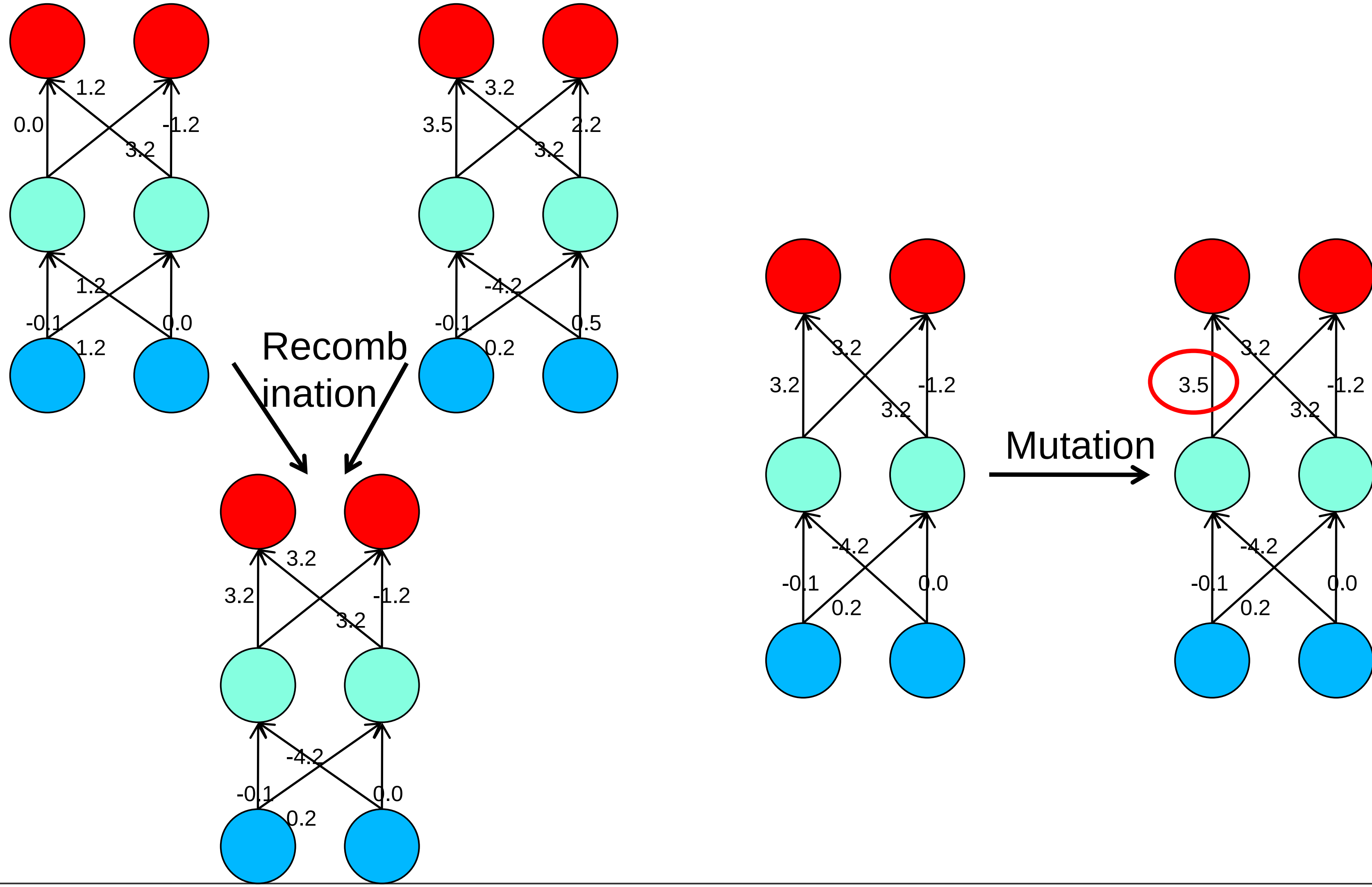


- Each neuron sums up the weighted outputs of the other connected neurons

$$o_i(k+1) = F(\sum_{j=0}^{n} w_{ji} o_j(k) + b_i)$$

- The output of the neuron is the result of an activation function (eg step, sigmoid) applied to this sum
- Neural network are distinguished by their connection structure
  - Feed forward connections (layered)
  - Recursive (Ouput neurons feed back to input layer)
  - Fully meshed

Evolving Neural Networks



Generalization property

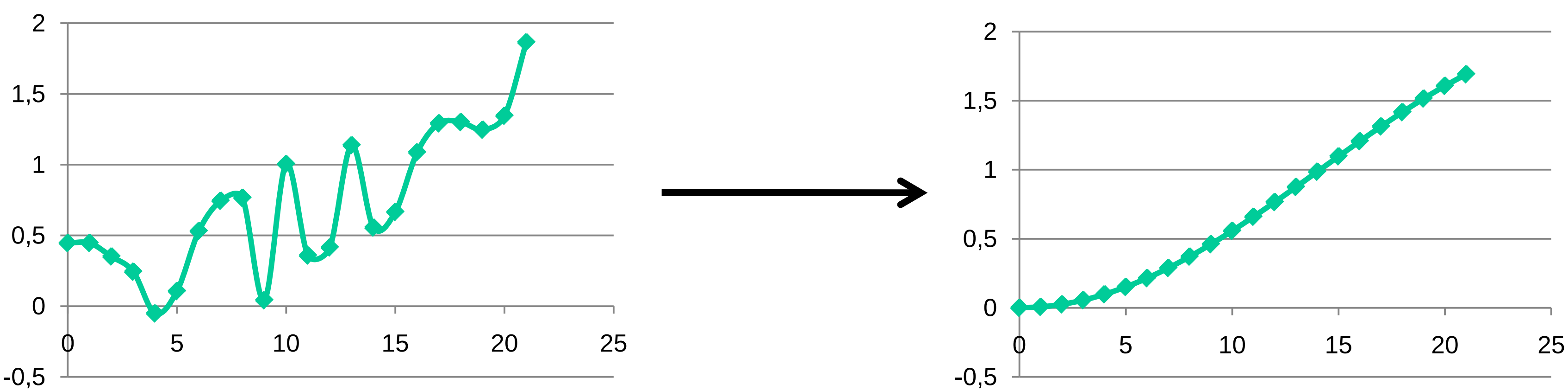
- For example, an ANN learned to recognize a pattern:

A

- Then, it will likely also recognize

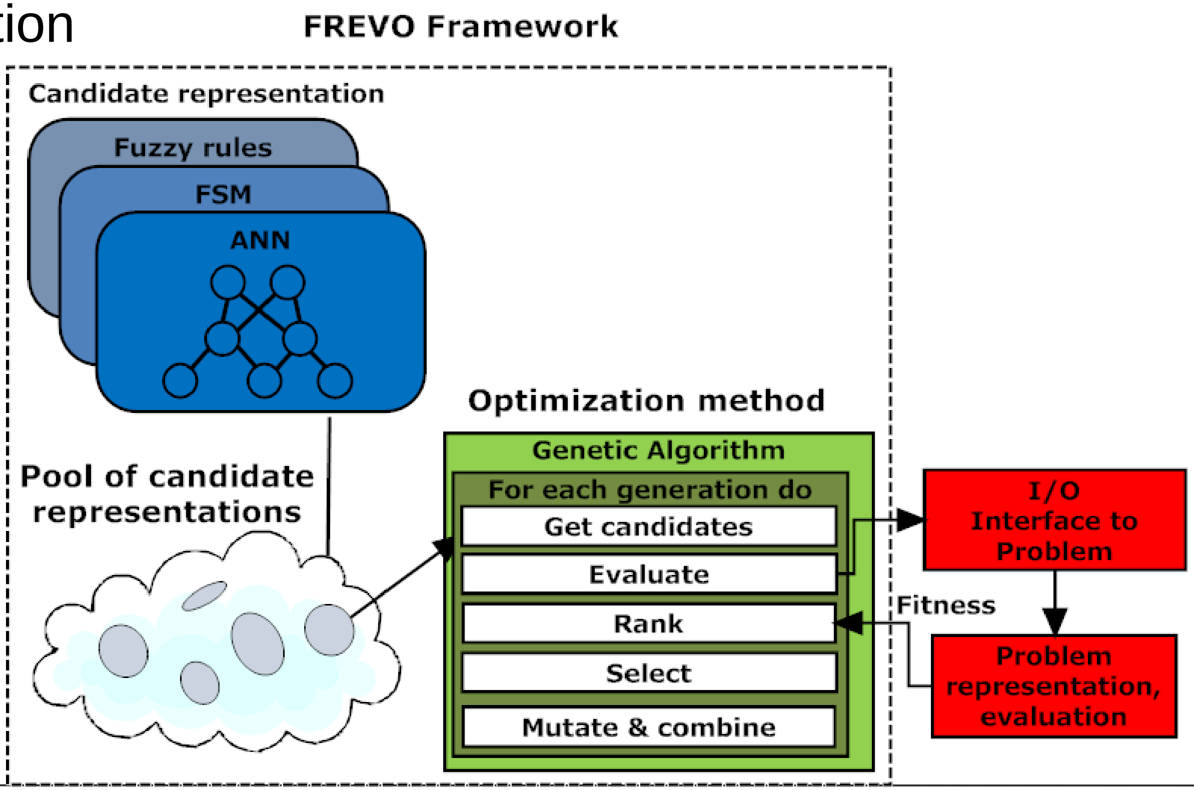
A A A

- This helps also in smoothing out the fitness landscape:



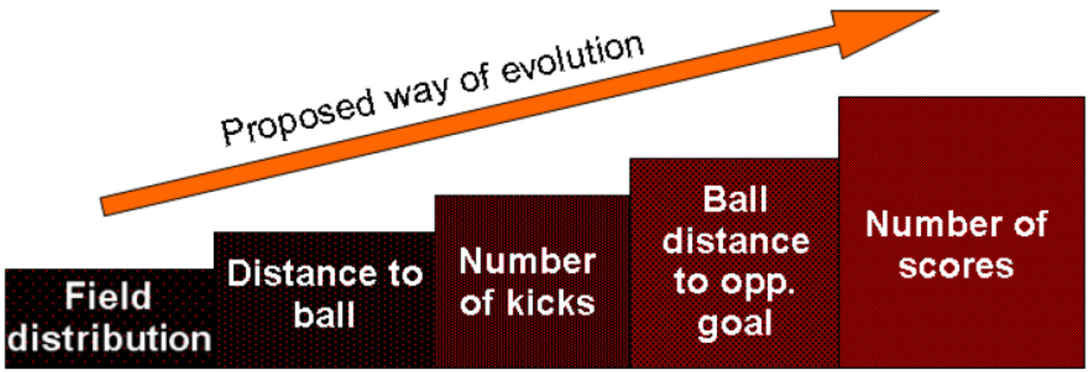
A Java Tool for Designing SOS

- FREVO (Framework for Evolutionary Design)
- Defines flexible components for
  - Controller representation
  - Problem specification
  - Optimizer



Giving FREVO a Problem

- Basically, we need a simulation of the problem
- Interface for sensor/actuator connections to the agents
- Feedback from a simulation run -> fitness value



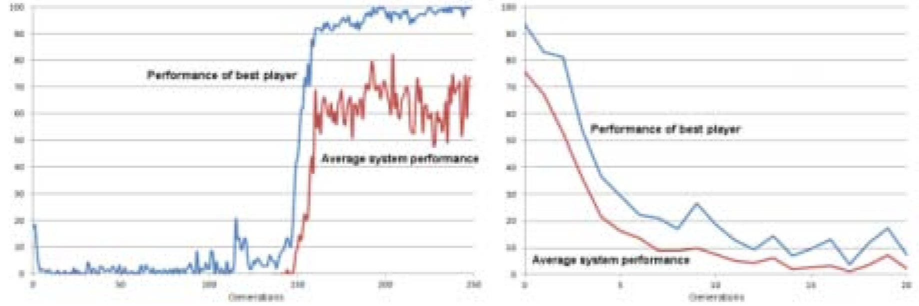
Multi-level fitness function for a robot soccer simulation

- Finally, compute for some hundred generations (can take days)

Example Applications of FREVO

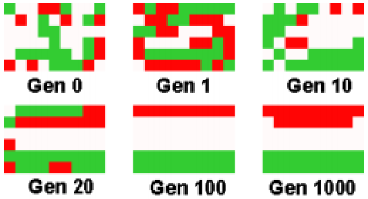
- Evolution of cooperative behavior in simulated robot soccer

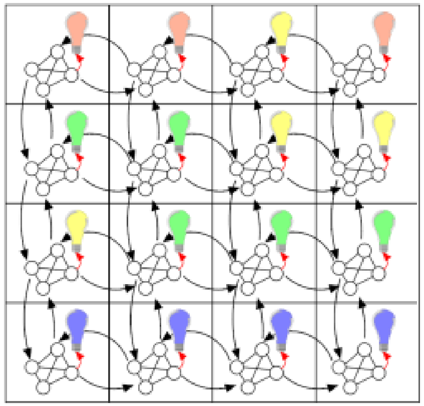
- Study on evolution of cooperative behavior



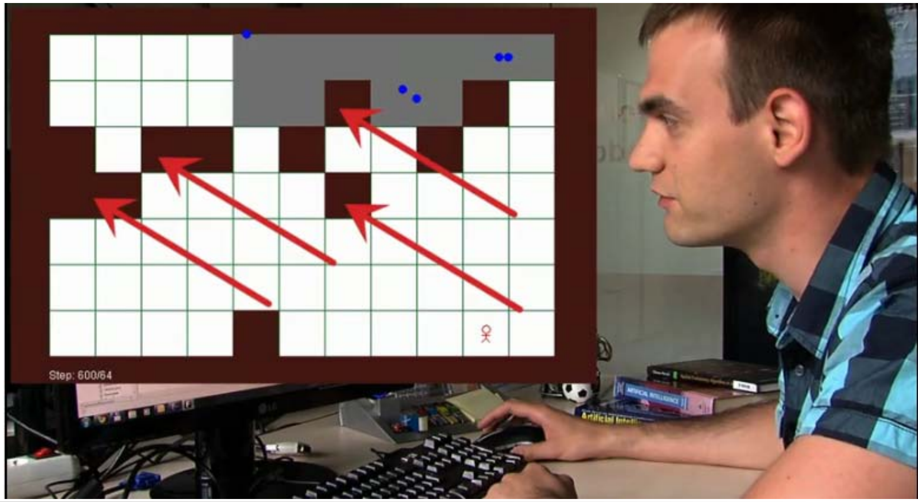
Example Applications of FREVO (cnt.)

- Case study on self-organizing cellular automata patterns





- Algorithm for coordinating microcopters in a search mission



Challenges

- Accurate modeling of target system
- Typically no immediate feedback possible
  - reinforcement learning instead of supervised learning
- Step from simulation to real system
- Creating trust in solutions
  - Validation/verification methods

…brief demonstration of

FREVO Framework for evolutionary design

Web links:

- FREVO tool download (open source): **www.frevotool.tk**
- Project DEMESOS (Design Methods for Self-Organizing Systems): **www.demesos.tk**