

Evolutionary Design II

Jason Noble

`jasonn@comp.leeds.ac.uk`

Biosystems group, School of Computing

Last time

- Harnessing evolution in a computer program
- How to construct a genetic algorithm
- Potential difficulties, alternative approaches
- Some history
- Applications: what are evolutionary algorithms good for?

This time

- Search spaces, fitness landscapes, and other metaphors...
- What are the preconditions for a GA to work well?
- Evolutionary robotics: using GAs to design robot brains and bodies

Fitness landscapes

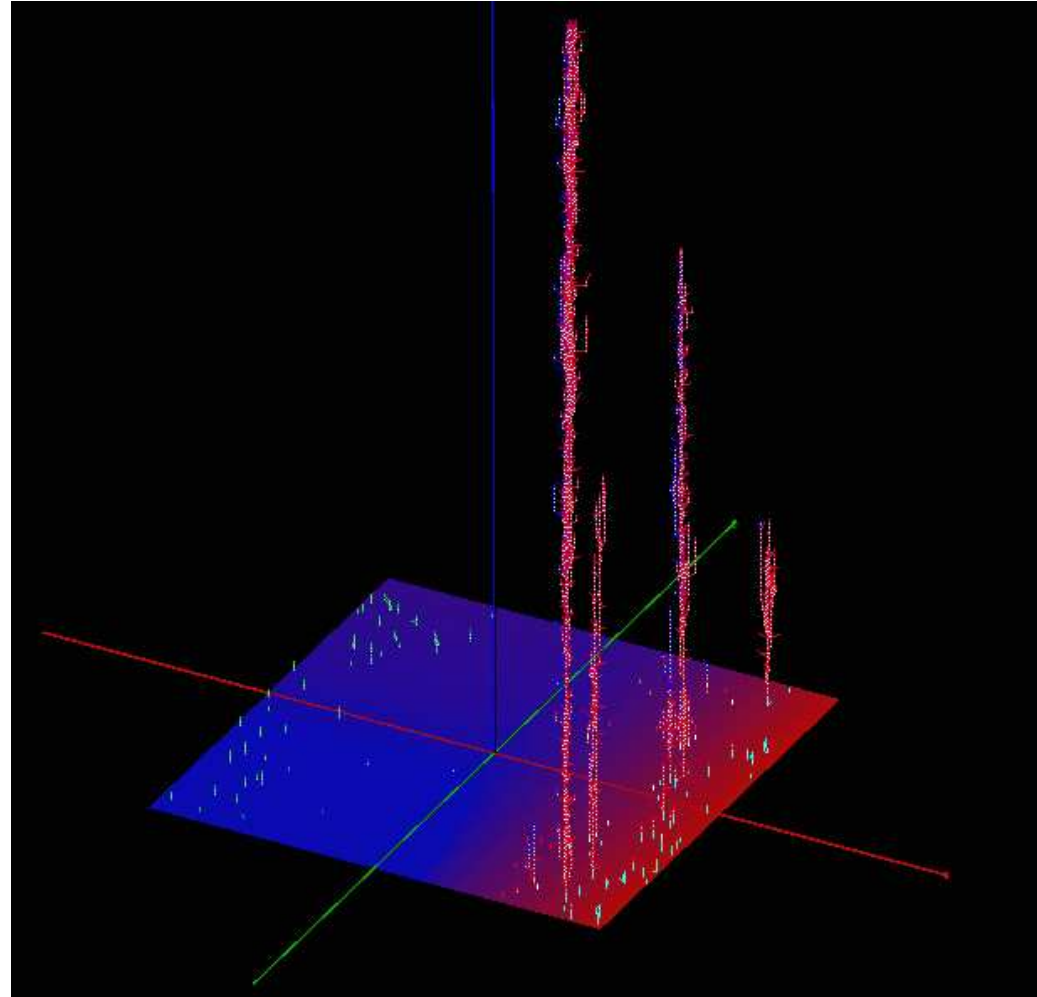


- A metaphor for thinking about what GAs do.
- Peaks on the landscape correspond to regions of high fitness.
- Remember that search is local: you can't just look around for the highest point.
- Metaphor suggests a potential problem: a GA may get stuck on a *local optimum*.

Premature convergence?

- GAs usually seeded with a randomly generated initial population.
- In the metaphor: individuals scattered all over the landscape.
- Very rapidly, the population *converges*, as individuals in low-fitness regions fail to reproduce.
- Average fitness goes up, but population diversity goes down: a good or a bad thing?

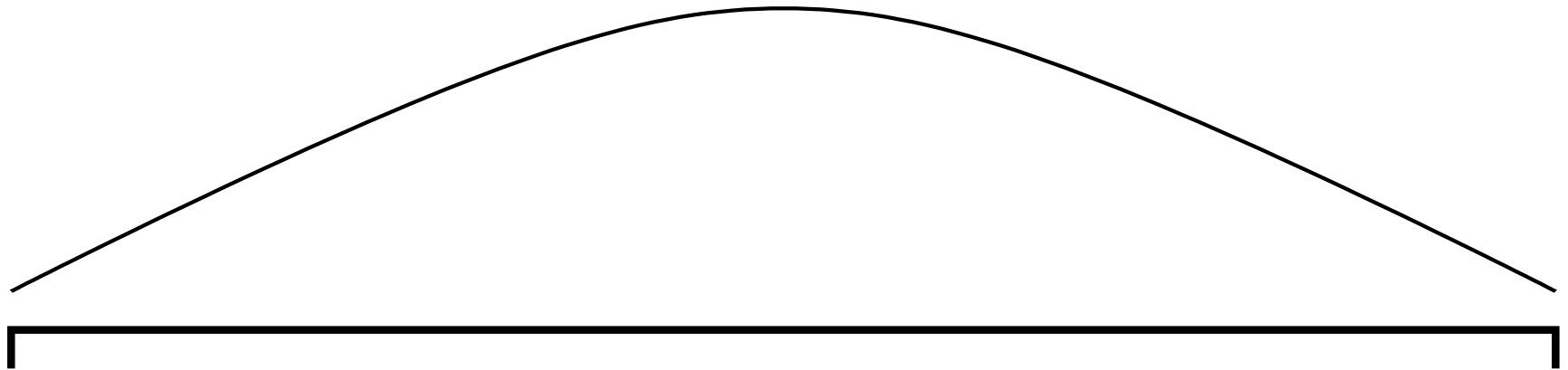
A GA in action



Note the convergence of the population, through the extinction of less fit lineages.

Types of landscapes for search

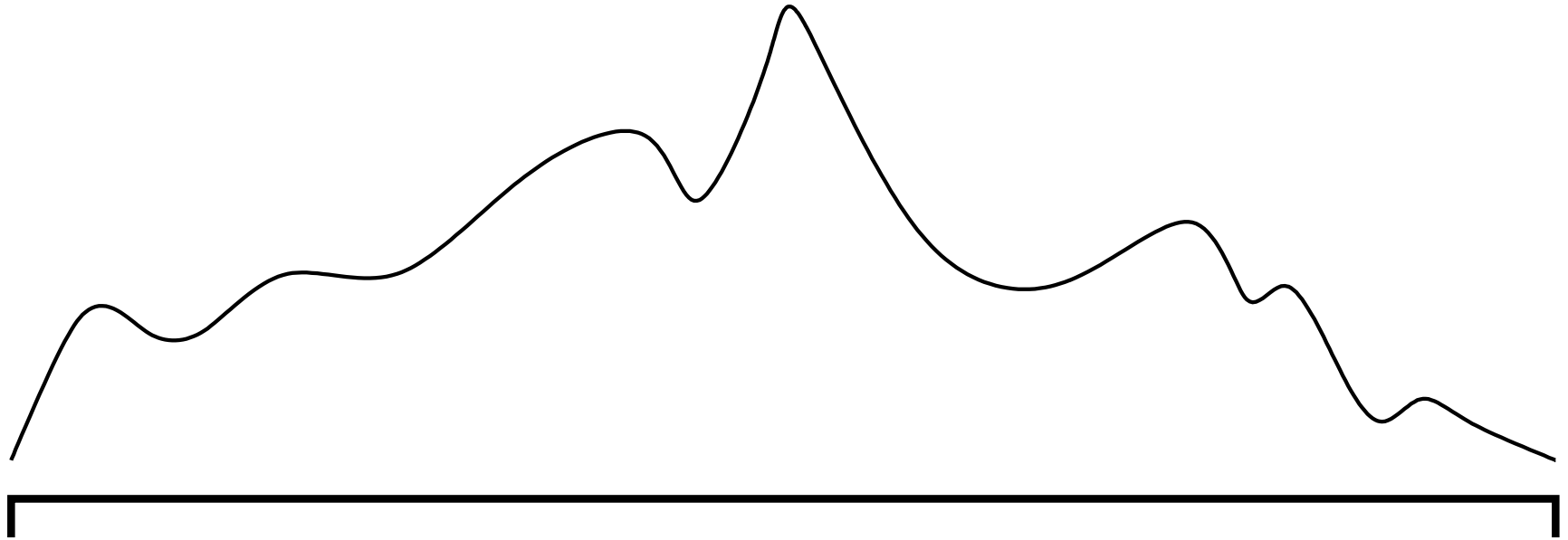
- What kinds of landscapes are there?
- Which ones are easy for a GA to search?



"Mt. Fuji" landscape

This landscape is smooth. It will be easy for a GA to improve in each generation.

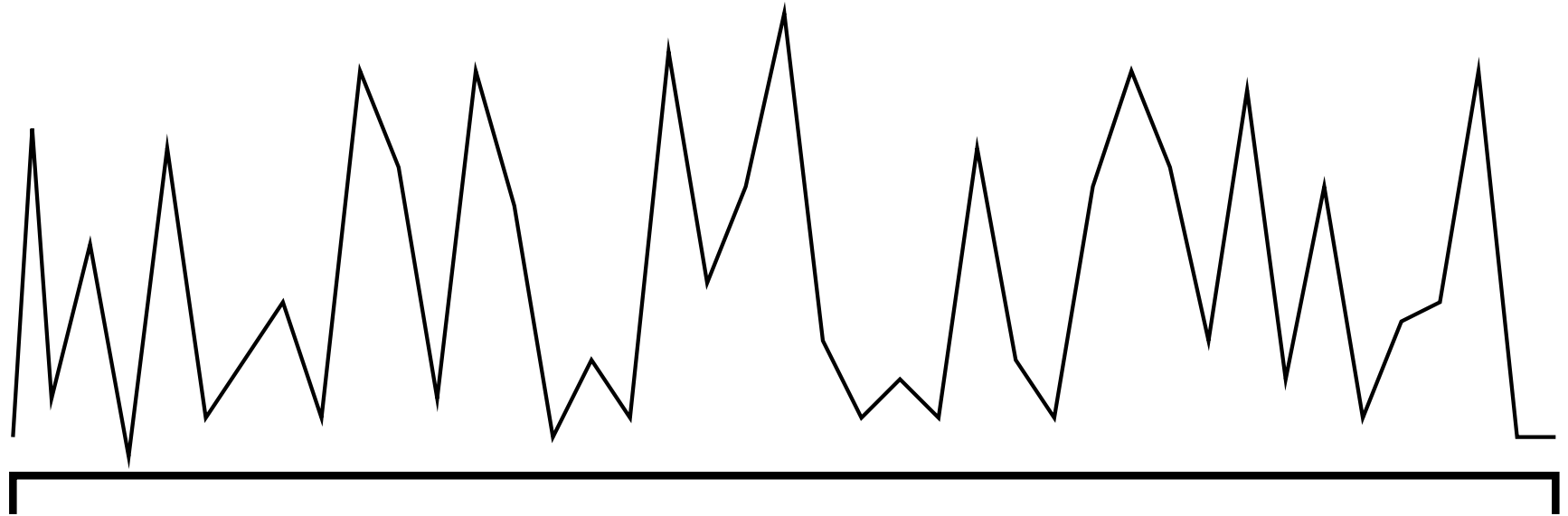
Types of landscapes for search



Moderately rugged landscape

This landscape is somewhat rugged, but mutation and crossover should ensure that the GA does not get stuck on local optima.

Types of landscapes for search



Highly rugged landscape

This landscape is extremely rugged, and the GA may be unable to find the global optimum.

Search spaces

- The term “landscape” suggests only two dimensions of variation.
- Let’s consider a *search space*, the multi-dimensional space of all possible solutions
- For a GA to work, the search space must be *locally correlated* to some degree, as real physical landscapes are.
- For example, if a basic pizza with ham and onions scores well, then ham alone, or ham, onions and olives, should also (more often than not) score well.

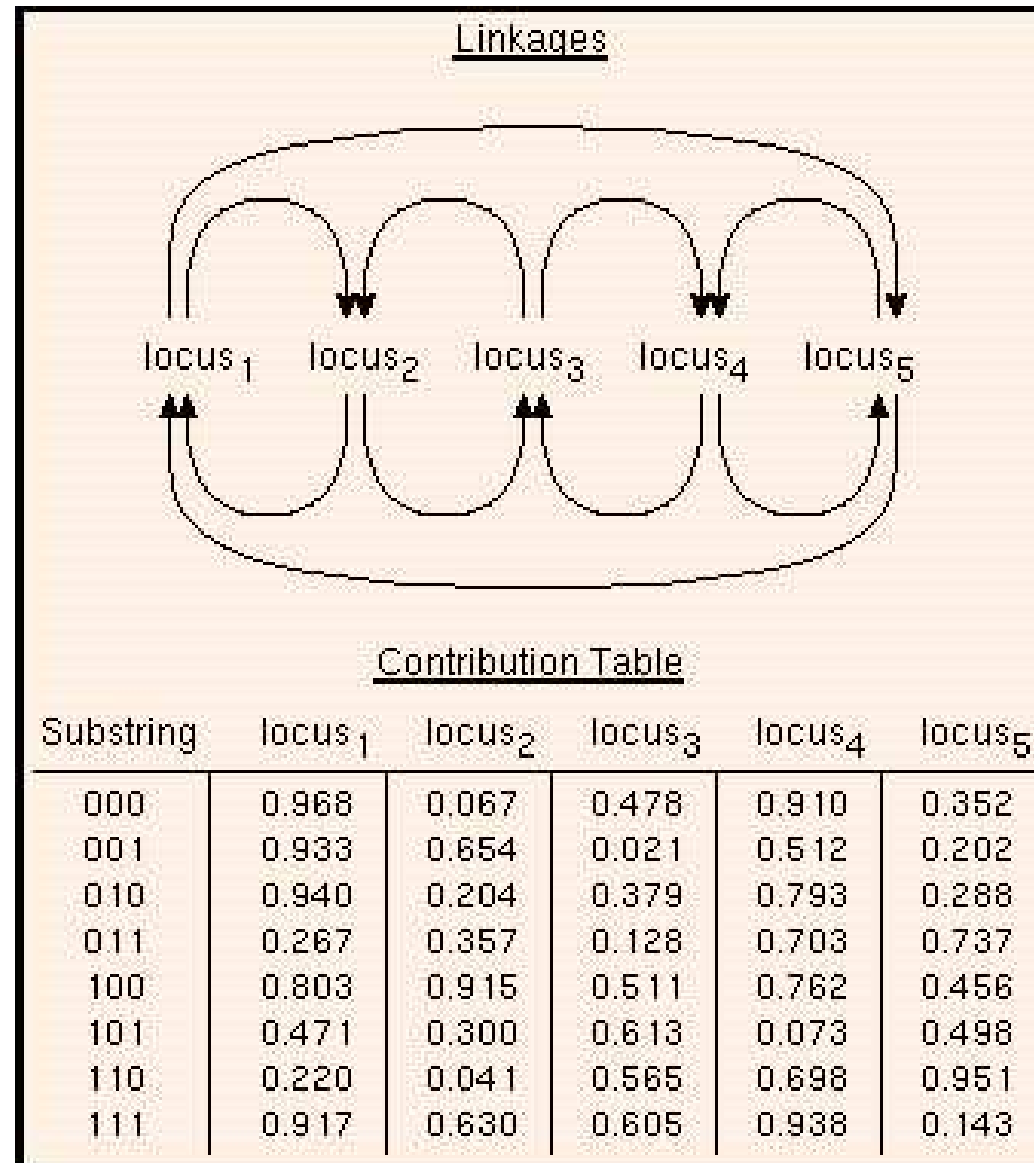
Ruggedness and epistasis

- As local correlation goes down, the ruggedness of the landscape goes up and it becomes more difficult to search.
- Rugged landscapes are high in *epistasis*, whereby the fitness contribution of a gene is modulated by other genes.
- For example, if anchovies = good, chillies = good, but anchovies + chillies = bad.
- Epistasis is also known as *interaction* (statistics) and *frustration of variables* (engineering), and it makes optimization problems hard.

Playing with epistasis

- Kauffman (1993) devised the *NK landscape* formalism for experimenting with epistasis: N genes are each influenced by K others.
- Concluded that GAs deal with epistasis quite well, up to a point.

Playing with epistasis



Other optimization methods?

- GAs may be appealing, but are they efficient? Is it really necessary to have a population, for example?
- Consider the *random restart hill-climber*:
 1. Choose a random parent solution
 2. Generate a mutated offspring
 3. If the mutant is better than the parent, it replaces the parent—go to step 2.
 4. If there are no improvements after N tries, go back to step 1 (but always remember the best solution so far).

GAs vs. hill climbers?

- The sad truth is that the random restart hill-climber wins on many test problems (e.g., Forrest & Mitchell's Royal Road functions).
- But GAs perform well on many real-world problems, so what's going on?
- Recent work with better test problems (Watson's Hierarchical If-And-Only-If) suggests that hierarchical problem structure is key to a GA's success.
- Example: lecture timetabling problems show hierarchical structure.

Two views of what GAs do

- GAs perform well on moderately rugged problem landscapes with the right kind of hierarchical structure. But how do they find good solutions?
- Returning to the notion of premature convergence:
 - early theory on GAs, associated with Holland's (1975) *schema theorem*, suggested this was a bad thing.
 - population could converge on a local optimum.
 - need a larger population size, or perhaps a higher mutation rate, to prevent this from happening.

A better metaphor?

- More recent theory on GAs (Harvey, 1992) accepts convergence as inevitable.
- After all, natural populations are highly genetically converged.
- Work on analyzing the search spaces for real-world problems suggests that extensive *neutral networks* usually exist.
- After converging, the population moves as a group or cloud, traversing a particular neutral network.
- Fitness improvements occur when a mutation reveals a way up to another network of higher fitness.

Evolutionary robotics

- Can we use GAs to evolve better robots?
- Some success with fairly simple behaviours:
 - obstacle avoidance
 - wall-following
 - sensory discrimination
 - evolved gait in six- and eight-legged robots
 - chasing and fleeing behaviours
 - simple navigation requiring memory

Potential problems

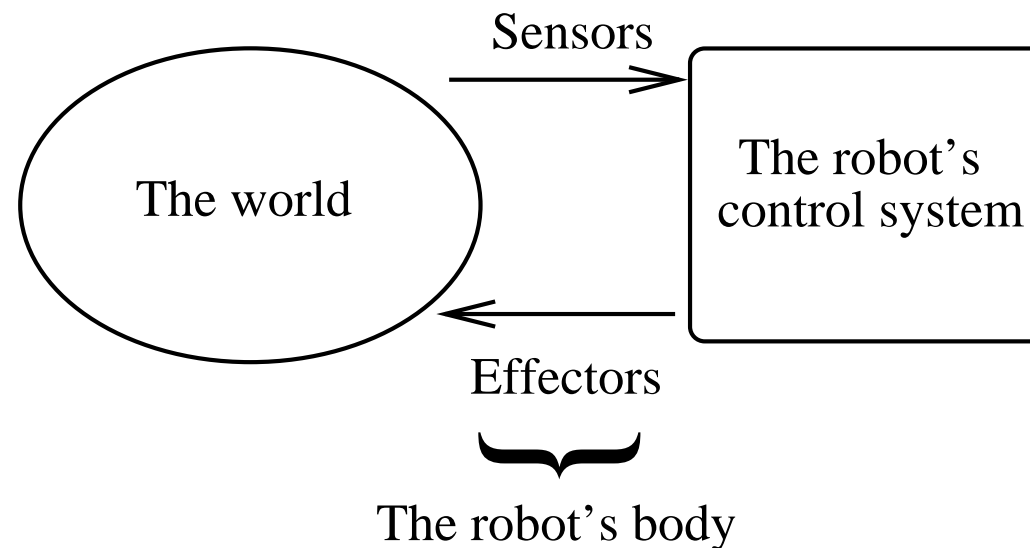
- How to represent robot designs as a bit-string (genotype-phenotype mapping)?
- How to assess the quality of a robot design (fitness function)?



Fitness functions covered last week—we will focus on the mapping problem.

Evolving brains or bodies?

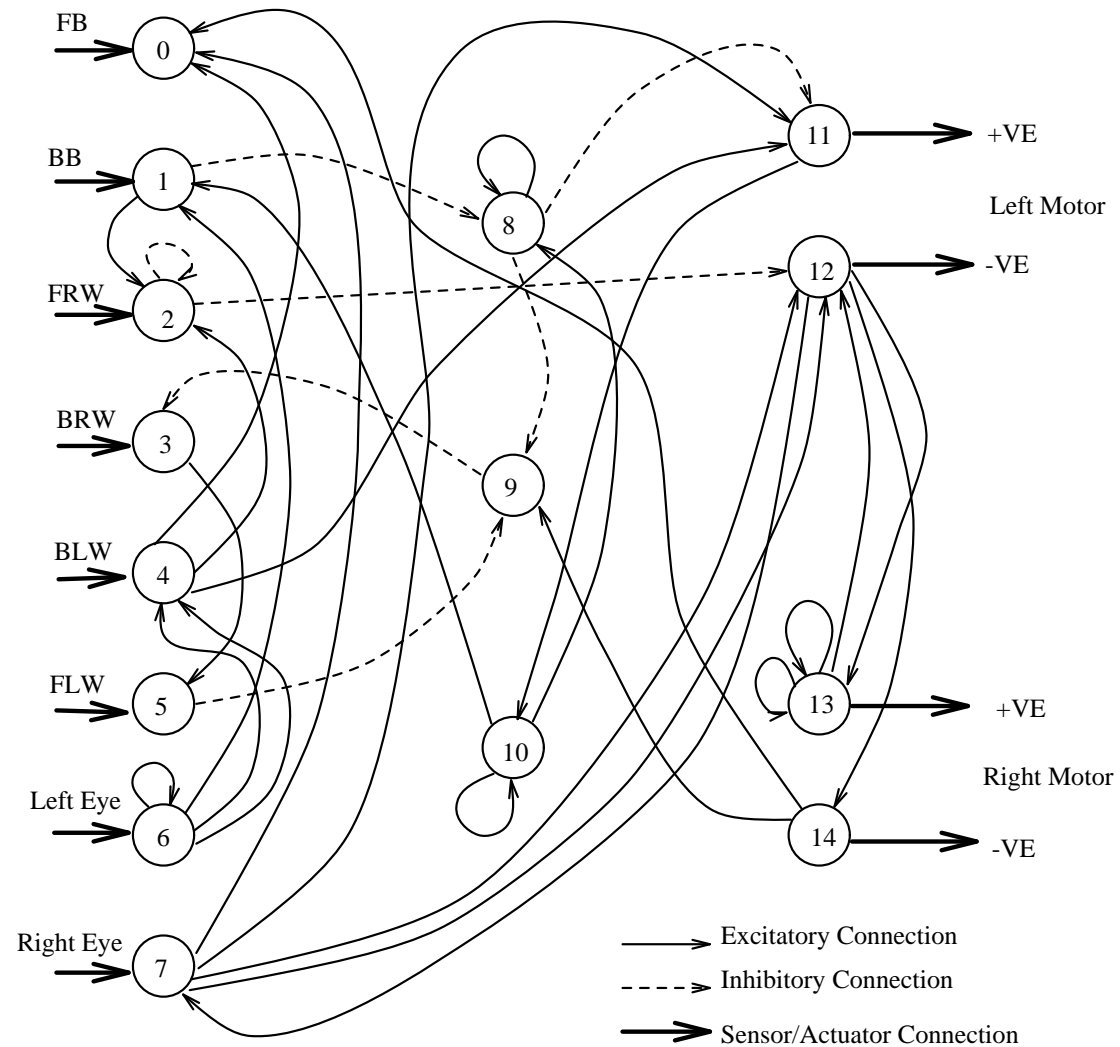
- Physical layout of most robots is relatively fixed.
- Easier to work on the evolution of robot control systems (e.g., neural nets, rule-based systems, finite-state machines), which tend to be far more flexible.



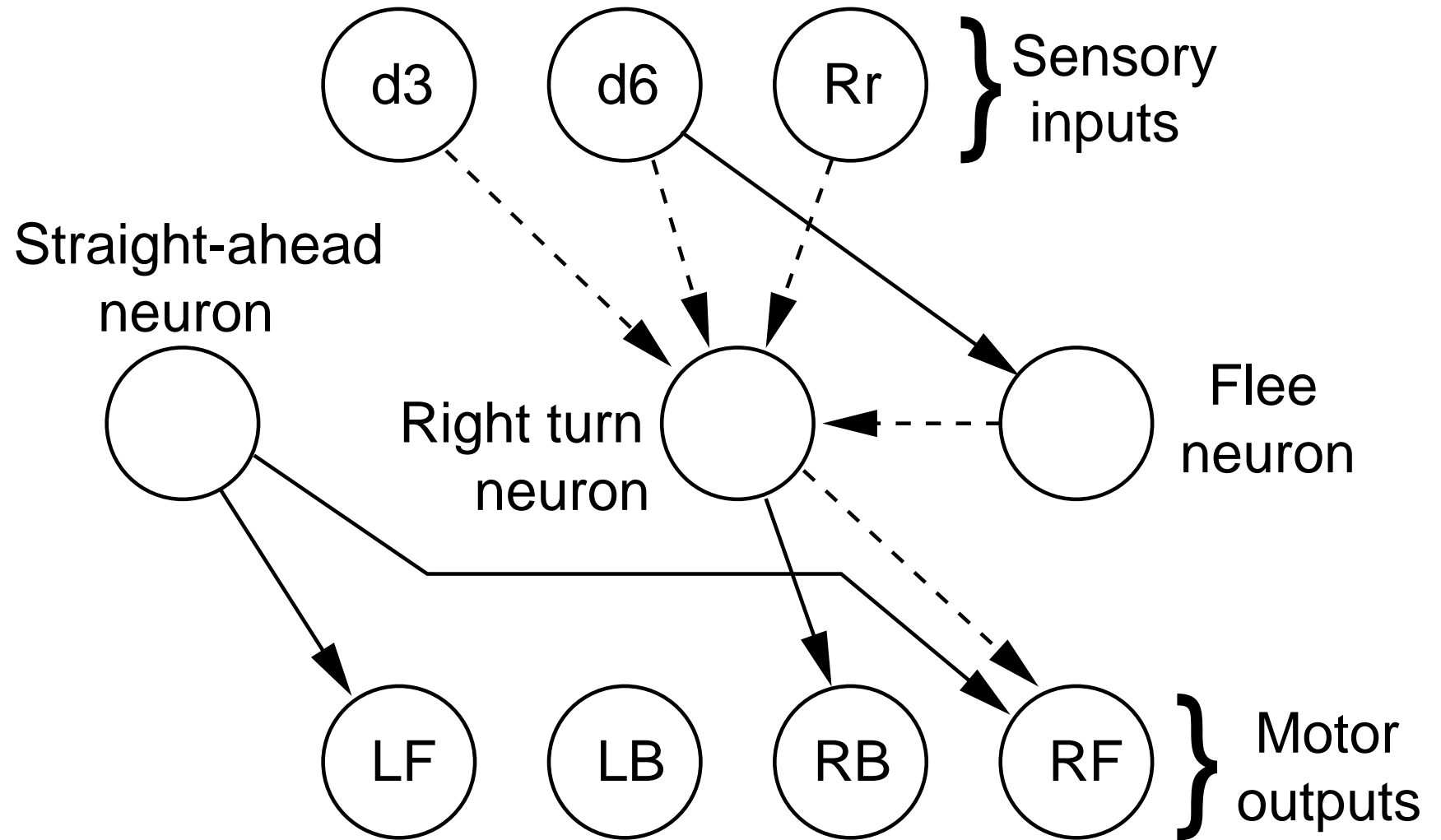
ANNs for robot control

- Neural networks an attractive choice for robot control:
 - capable of “generalization”
 - tolerant to noise
 - degrade gracefully under component failure
- This means that the fitness landscape for their evolution is likely to be relatively smooth.
- Continuous-time recurrent neural networks are particularly attractive, as they can approximate any smooth dynamical system.

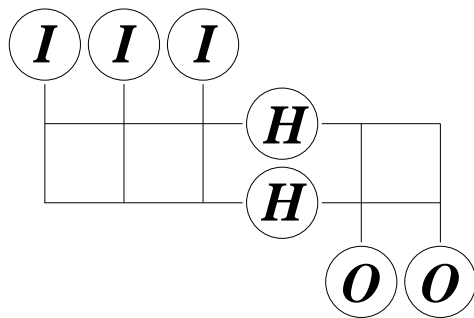
Sample evolved networks



Sample evolved networks



Encoding ANNs



- Simplest genetic encoding scheme for a neural net is to directly specify the connection weights within a fixed architecture.
- Specifying the connections for the simple feed-forward net shown here would take only 10 real values.
- However, the network architecture would be fixed, and if it was inadequate for solving the problem, the GA could not succeed.

Wish list for encoding schemes

Complete: any network can be encoded.

Compact: encodings should be of minimal size.

Closed: any genotype should produce a meaningful network.

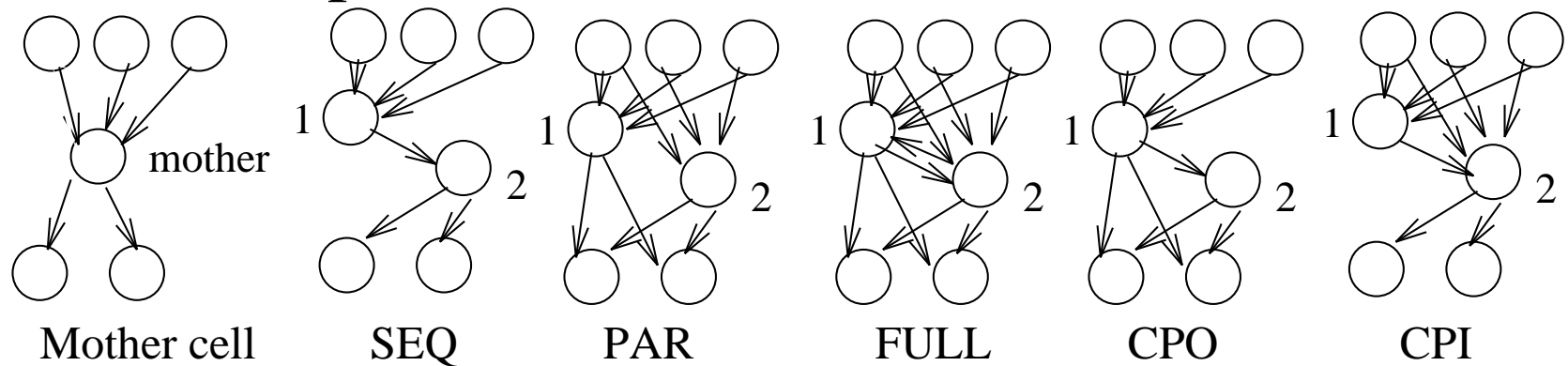
Modular: possibility of sub-networks, module re-use, recursion (as in real neural circuits, e.g., motion detection in flies).

Scalable: fixed-length encoding allows for a range of network complexities (direct encodings mean huge search spaces).

Expressive power: scheme expresses all network parameters (Mataric & Cliff, 1996).

Example 1

- Gruau and Quatramaran (1996) used a “cellular encoding”.
- Start with a single neuron connected to inputs and the outputs.

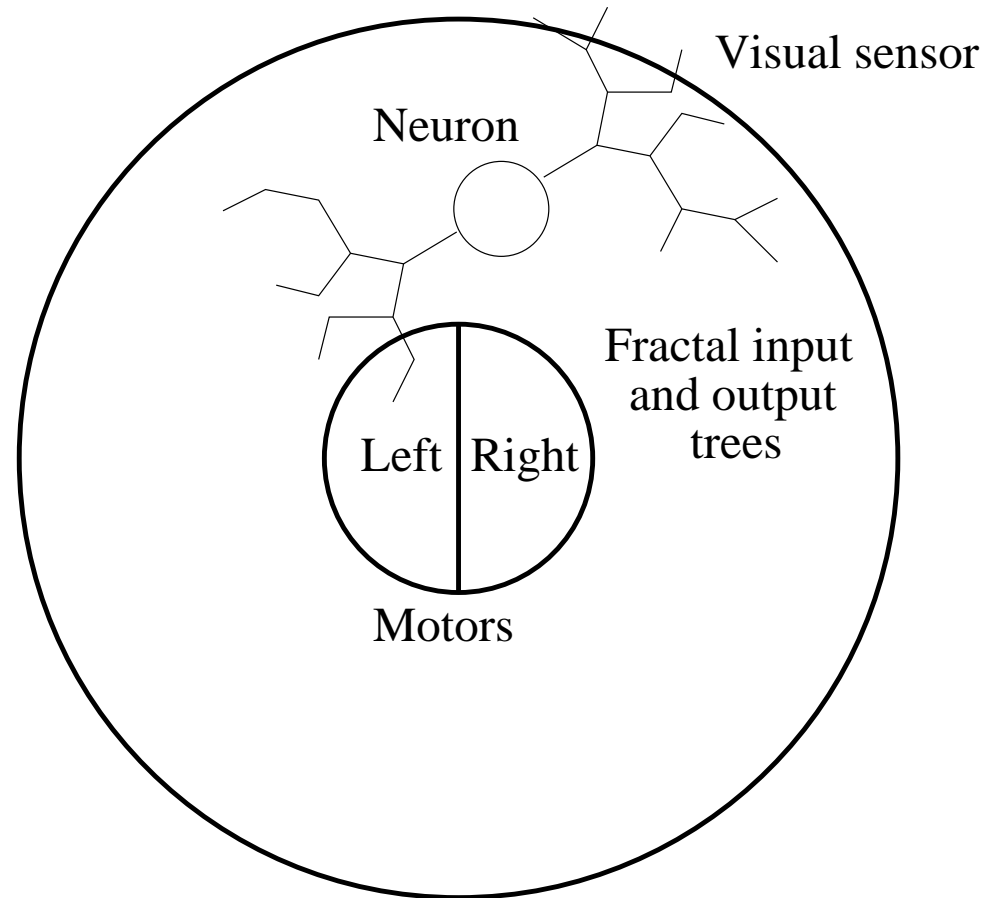


- The genotype spells out a series of parallel transformations like the ones above.
- Method used to evolve a walking gait in an 8-legged robot.

Example 2

- Cliff and Miller (1996) used a fractal tree-growing method to model the growth of axons, dendrites and synapses.
- Move toward biological plausibility, but the search spaces were very big and very sparse.
- Duplicating parts of the genetic code found to be useful.
- One attractive feature of the model was that sensor morphology was based on the same encoding—starting to cross the body/brain divide.

Example 2



Method used to get pursuit and evasion behaviour in simulated robots (more on this work next lecture).

Evolving in simulation

- How to test the fitness of a robot design?
 1. program a physical simulation of the robot's body and its environment.
 2. hook up each evolved neural net to the simulated robot, and measure how well it works.
- If the simulation has been done right, the best neural net can be downloaded onto a real robot and should work.
- The attractions of the idea are speed and flexibility.

Problems?

The problems with evolving in simulation are:

- Modelling the real world accurately is hard, especially the problem of getting the right level of noise into the simulated sensor readings. (If the computer model is very detailed, it may actually run *slower* than real-time.)
- A great deal of effort is required to construct accurate simulations of particular robot bodies in particular environments. This work does not generalize well.

GAs in the real world

- Possible in principle to evolve robots in the real world.
- Guaranteed validity, but all sorts of problems:
 - limited battery life
 - tangled tether cords,
 - returning the robot(s) to the starting position after each trial.

Simulation *and* reality

Possible to combine the speed of simulation with the grittiness of the real world.

- Thompson's "Mr. Chips" robot: fitness assessed in virtual reality. Robot mounted on jacks, the spinning of its wheels was linked to its progress around a virtual environment. Used to evolve room-centring behaviour.
- The gantry robot: rather than simulate real vision (prohibitive computational costs), just mount a camera with a rotating mirror on a gantry. The camera is moved around the environment and "pretends" to be a two-wheeled Khepera robot. Used to evolve robots that can distinguish shapes.

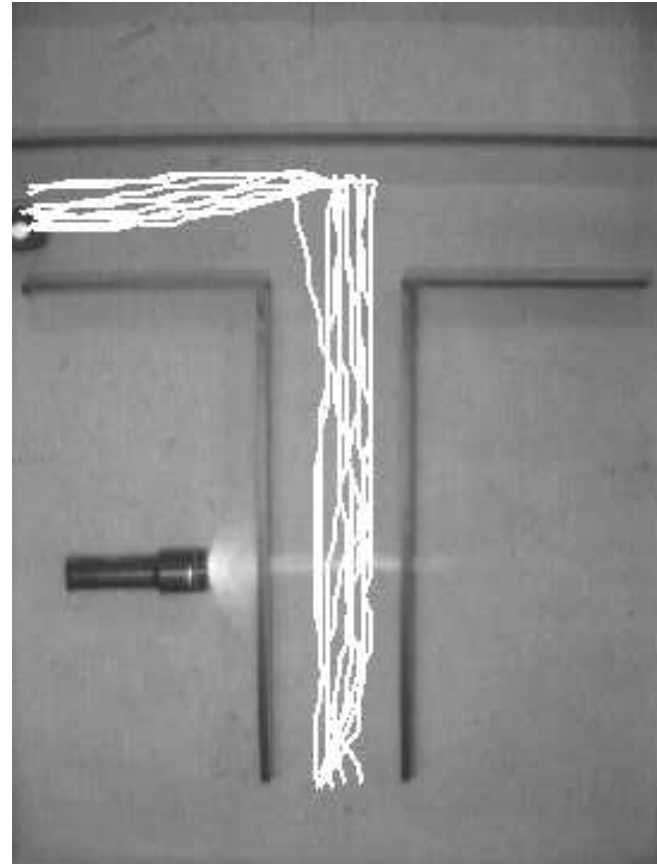
General problems

- All of these methods of automating robot design can be subject to serious problems of time and scaling-up raised by Mataric and Cliff (1996).
- If a range of parameter values (e.g., lighting conditions, starting position, colour of walls) are investigated, it will take forever.
- Particular problem of extended behaviours, in which a robot is supposed to “do X for as long as possible”, etc.

Quick and dirty simulations?

- Possible solution to scaling-up problem offered by Jakobi (1997).
- Use cheap, fast simulations that simulate just the right parts of reality.
- Add wildly unpredictable noise to all other aspects of the simulation so that the robot cannot come to rely on anything but the “base set”.

Quick and dirty simulations?



Used to evolve robots that can navigate a T-maze and turn left or right depending on which side the light was.

Sometimes it does work



The Golem Project, DEMO Lab, Brandeis University.

Next time

- Co-evolution and “arms races” in nature
- A co-evolutionary GA
- Co-evolved sorting networks
- Further applications in biology and engineering

References

- Cliff, D., & Miller, G. F. (1996). Coevolution of pursuit and evasion II: Simulation methods and results. In Maes, P., Matarić, M., Meyer, J.-A., Pollack, J., & Wilson, S. W. (Eds.), *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pp. 506–515. MIT Press / Bradford Books, Cambridge, MA.
- Gruau, F., & Quatramaran, K. (1996). Cellular encoding for interactive robotics. Cognitive science research paper 425, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK.
- Harvey, I. (1992). Species Adaptation Genetic Algorithms: A basis for a continuing SAGA. In Varela, F. J., & Bourgine, P. (Eds.), *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pp. 346–354. MIT Press / Bradford Books, Cambridge, MA.
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.
- Jakobi, N. (1997). Evolutionary robotics and the radical envelope of noise hypothesis. Cognitive science research paper 457, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK.
- Kauffman, S. A. (1993). *The Origins of Order*. Oxford University Press, New York.
- Matarić, M. J., & Cliff, D. (1996). Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems*, 19(1), 67–83.