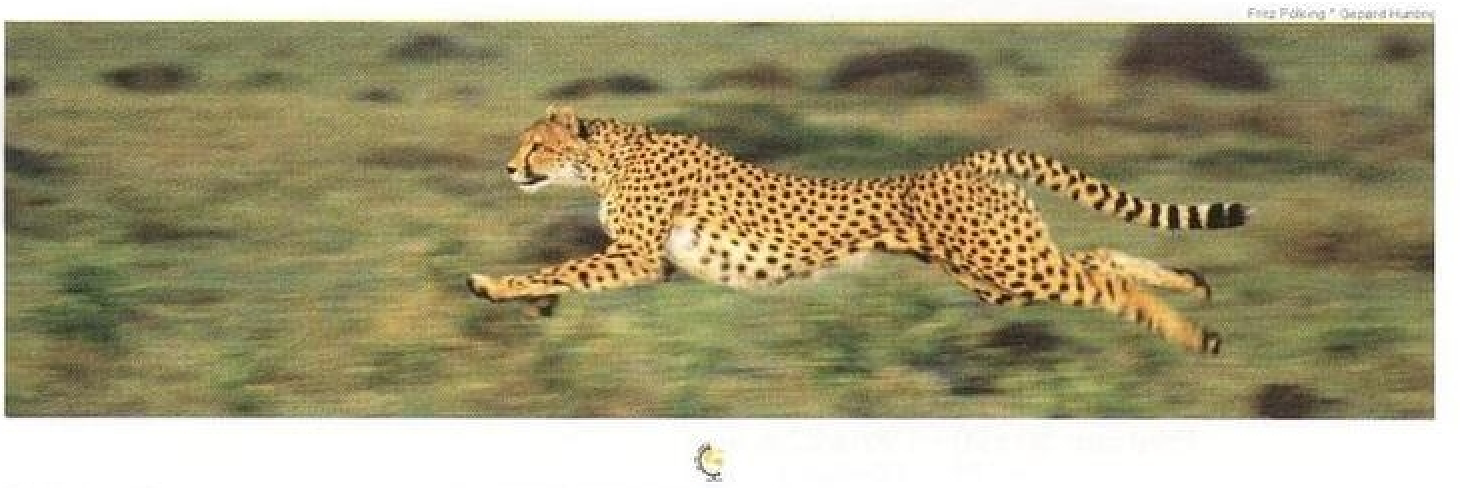
Co-Evolution

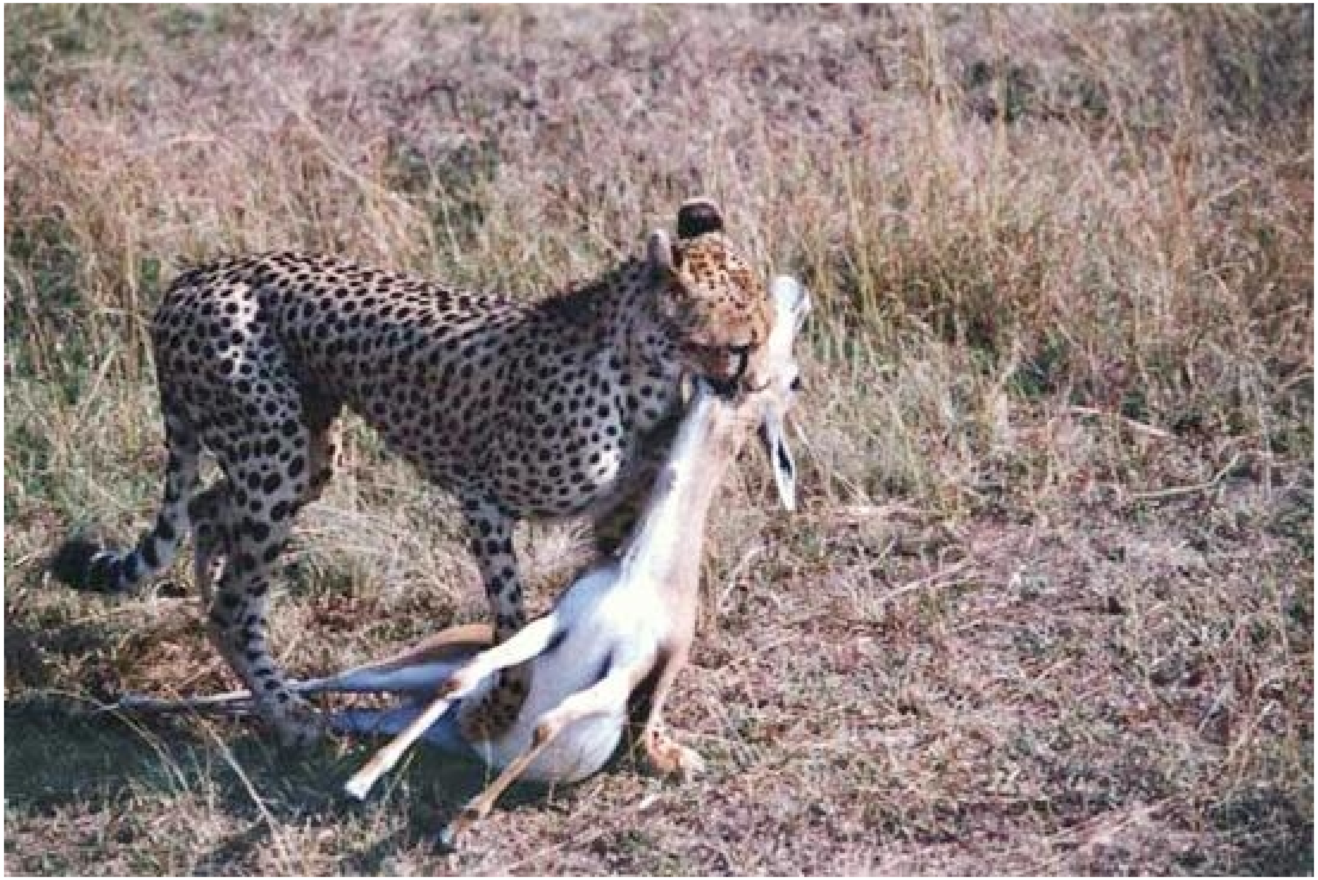
Ata Kaban

University of Birmingham

Cheetah: 60-70 mph, for up to 100 yards



Thompson's Gazelle: 50 mph...



The outcome (but about 50% of attempts are failures)

What is Co-Evolution?

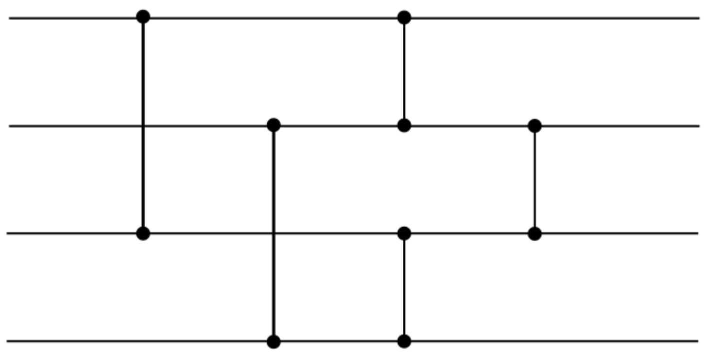
- Fitness of an individual depends on other individuals
  - Fitness landscape changes
  - Fitness of an individual may be different in different runs
- Change in one individual will change the fitness landscape in others
- Remember the Iterated Prisoner's Dilemma experiments? – the ones in which evolving strategies played against each other

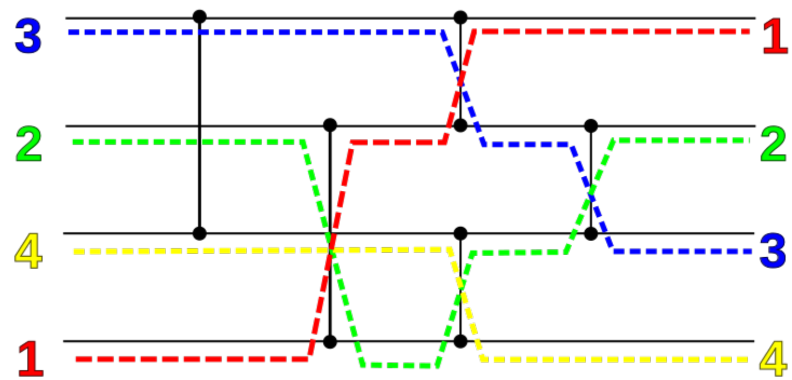
Types of co-evolution

- By evaluation
  - Competitive
  - Cooperative
- By population-organisation
  - Inter-population
  - Intra-population

Example1: Sorting algorithm

- Goal: place the elements in a data structure (e.g. list or tree) in some specified order (e.g. numerical or alphanumeric)
- One approach in Knuth's book is the so called sorting network (for fixed nos of elements)
  - Horizontal lines=elements in the list
  - Vertical arrows=comparisons to be made (in parallel)
  - If compared elements are in wrong order than swap
- Will look at the simple case of n=16 elements (e0—e15)




3
2
4
1
1
2
3
4

Designing sorting algorithms

- Goal: make them
  - correct
  - efficient (reduce no of comparisons)
- What is the minimum number of comparisons necessary for correct sorting?
  - hot Q around the ’60s (n=16 case)
    - ’62: Bose & Nelson developed a net that needs 65 comparisons
    - ’64: Batcher, Flyod, Knuth – 63 comparisons
    - ’69: Shapiro – 62 comparisons
    - ’69: Green – 60 comparisons
    - ’80s: W.D. Hillis – can GA find an answer to this problem?

- Encoding the sorting net
  - ordered list of pairs to be compared = phenotypes
  - considered nets of 60—120 comparisons
  - genotype: diploid chromosomes
    - 1 individual = 15 pairs of 32 bit chromosomes, each encoding 4 comparisons
    - See more details on M. Mitchell, pp. 21—27.
- Fitness measure: the percentage of cases sorted correctly
  - Problem: how to compute this?
    - Test all possible inputs – slow
    - Test fixed set of inputs? – which?

- First trials
  - To foster speciation, individuals placed on a 2D grid (i.e. spatial distance between them)
  - Fitness computed from random subsamples
  - Half of population with lower fitness deleted – replaced with a copy of a surviving neighbor
  - Pairing in the local neighborhoods
  - Special crossover for diploids, followed by mutation with p_m = 0.001.
  - Population size between 512—1million
  - 5000 generation

→ result: GA found sorting net of 65 comparisons ☹

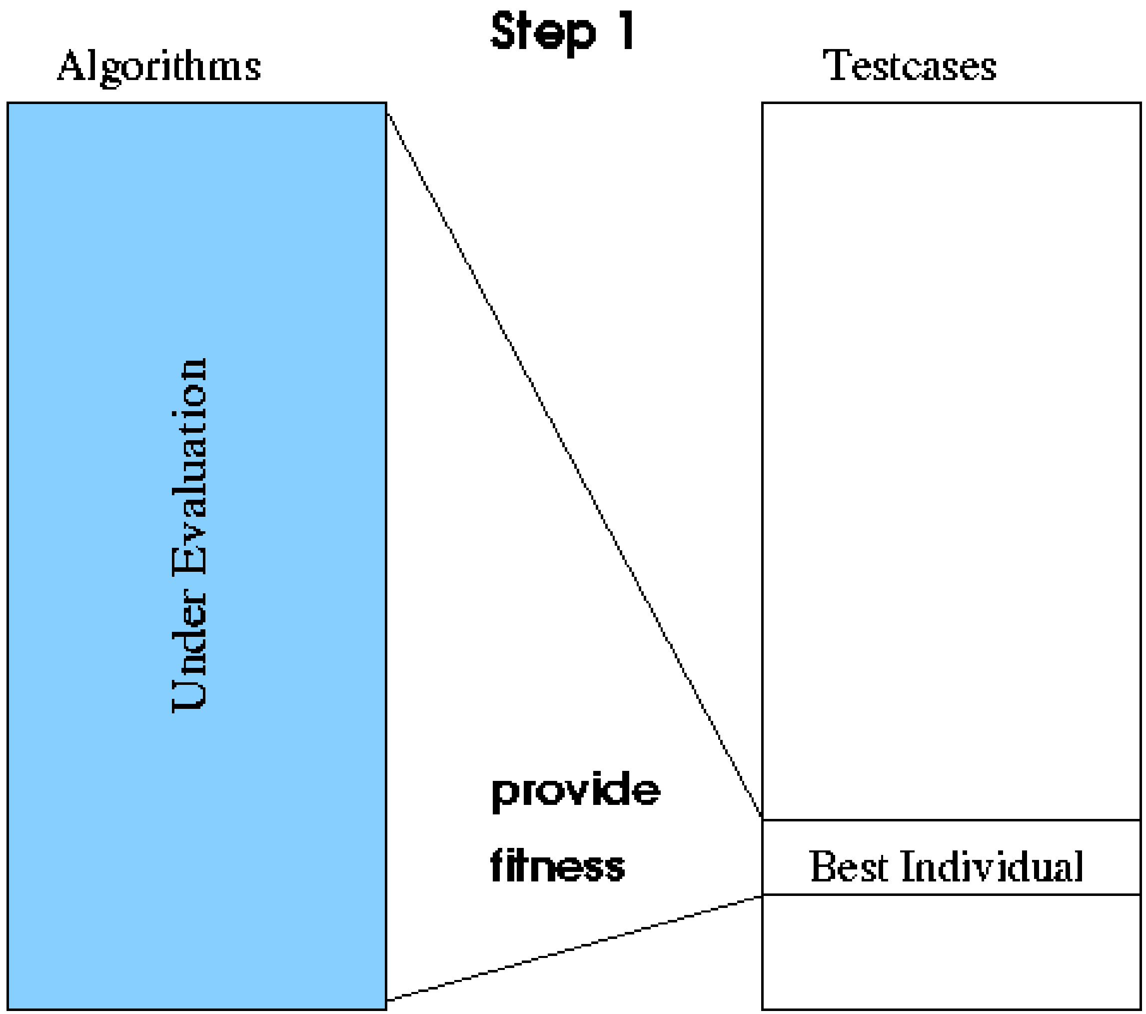
- Why didn’t the GA do better?
  - After early generations, with randomly generated test cases used to compute fitness, the difficulty of test cases stayed roughly the same!
- Solution: Co-Evolution

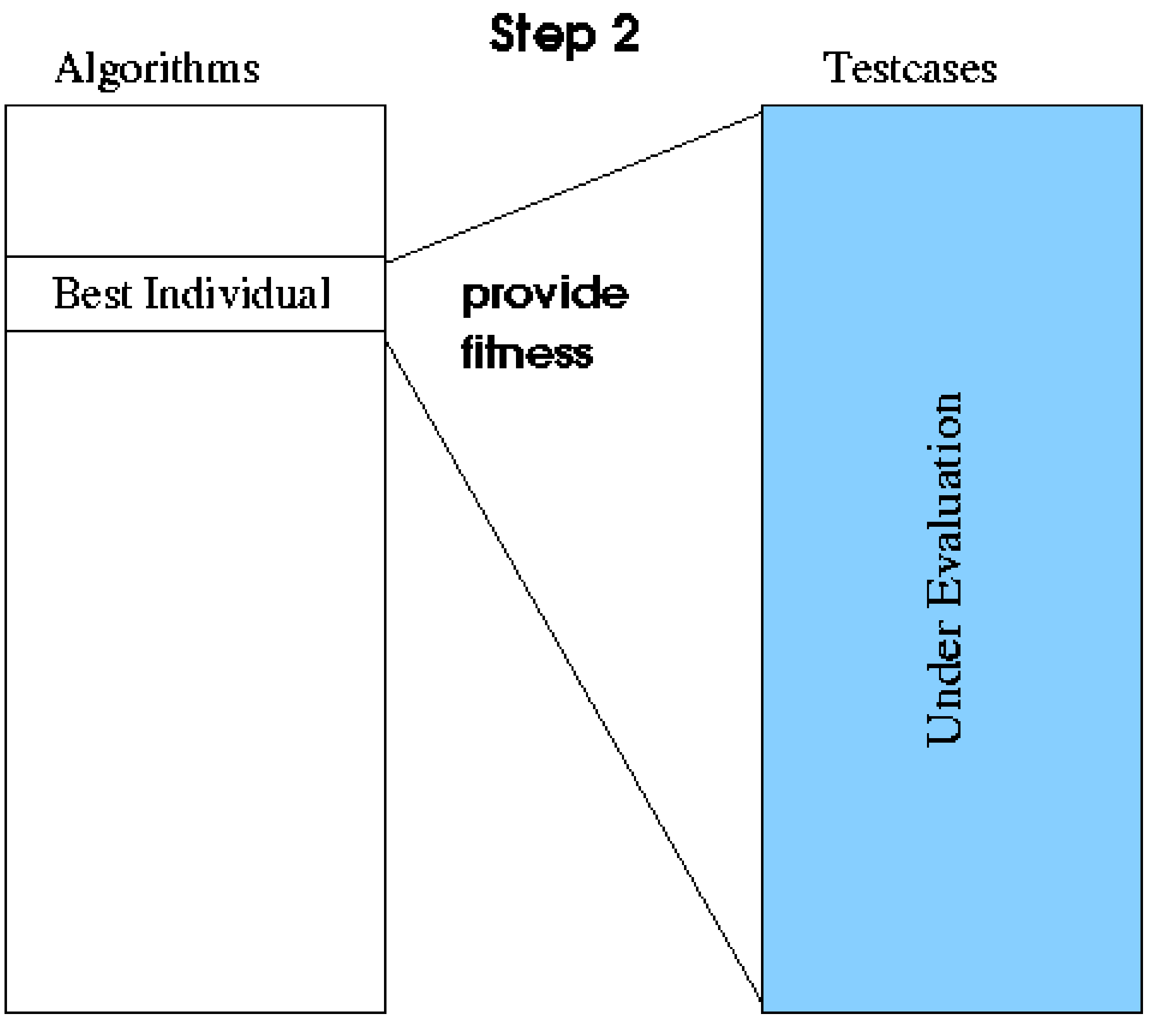
  = evolve both algorithms and test cases!

    - Algorithms try to sort
    - Test cases try to ‘trip up’ algorithms

  → predator/prey (or host/parasite) relationship inspired from nature

  → as the algorithms got better, the test cases got harder (specifically targeting weaknesses in the networks)
- Result: 61 comparisons


Step 1
Algorithms
Testcases
Under Evaluation
provide
fitness
Best Individual


Step 2
Algorithms
Testcases
Best Individual
provide
fitness
Under Evaluation

- Importance of the work:
  - introduces new technique inspired by co-evolution. The results are convincing in that this new technique is potentially powerful.

- Inter-population competitive co-evolution

Example2: Game Playing

- Intra-population competitive co-evolution
- Task: evolve a backgammon player
- Problem: evaluation
  - Against human player
  - Against ‘conventional’ program
  - Against internet players
- Solution: co-evolution
  - Play against other evolving programs
- Intra-population
  - All genotypes are of the same type
  - Only one population

Co-Evolving Backgammon Players

- TD-Gammon
  - Grand Master level player
  - Learns by self-playing
- What makes it successful ?
  - Temporal-Difference Learning ?
  - Or simply self-playing ?
- Simple use of NN in game play:
  - Generate all legal moves
  - Feed them to the NN
  - Best output is new move
  - NN used to evaluate positions

Simple Backgammon Learner

Evolve a NN that plays backgammon

1. Initial NN is NN(k) ; k=0;
2. Generate a mutant challenger of NNk:

   w‘(i,j) = w(i,j) + Gaussian (0, s);

3. If NN‘(k) is beaten by NN(k)

   NN(k+1) = NN(k)

   else

   NN(k+1) = 0.95 * NN(k) + 0.05 * NN‘(k)

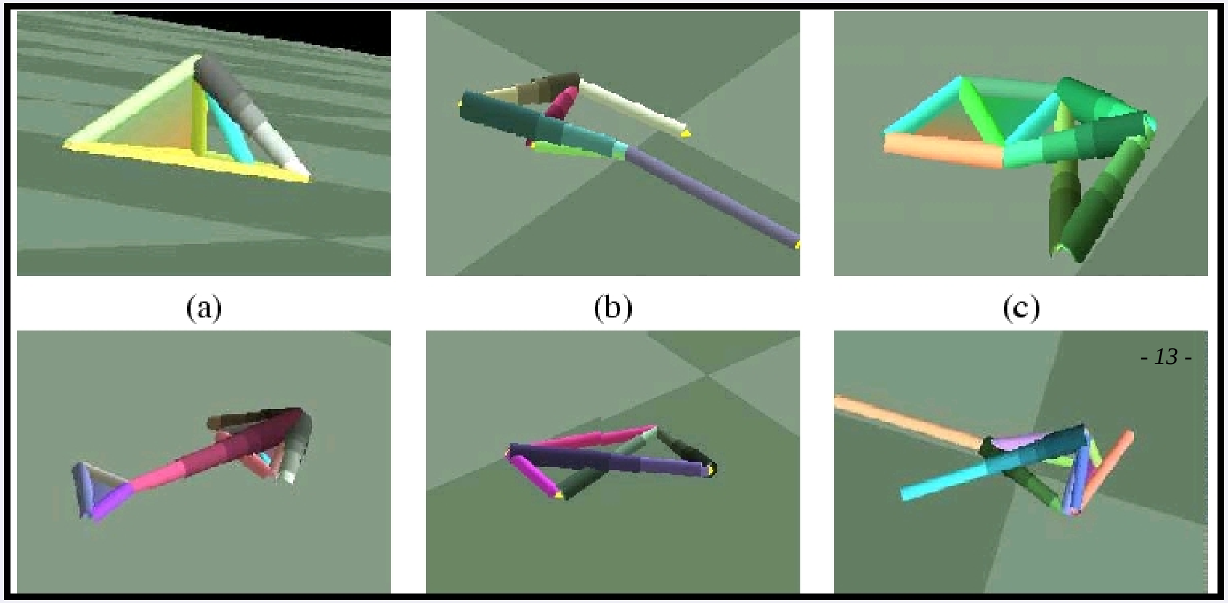
4. k = k+1; go to step 2 unless finished

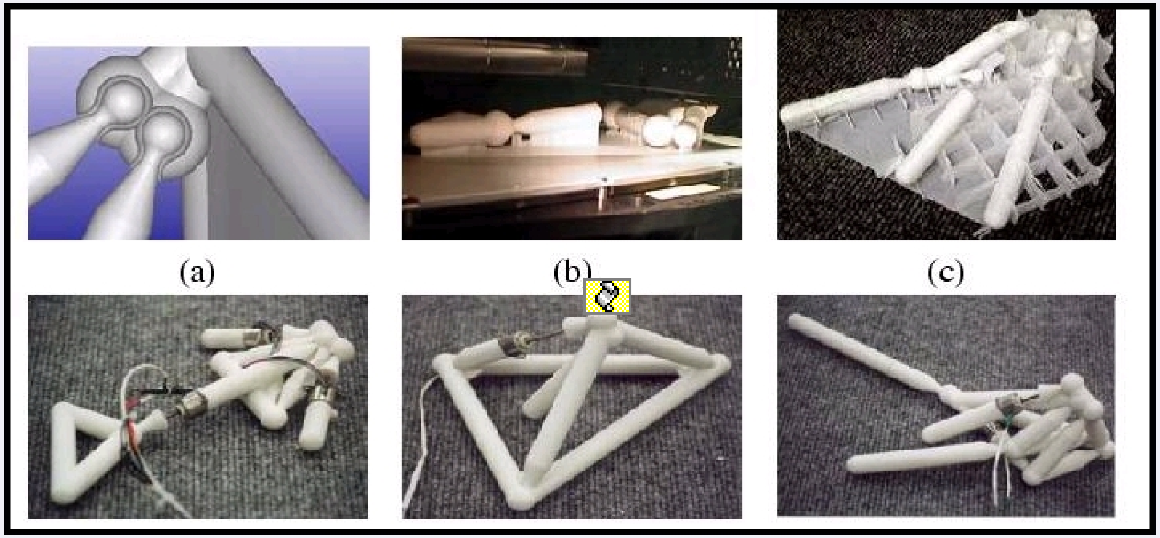
- Setting
  - Uses 197-20-1 fully connected feed-forward NN; initial weights 0s
  - No training for NNs !
  - EA with population size 1
  - Only simple mutation: add Gaussian noise to weights
  - No recombination
  - Performance
- 40% winning against a strong expert-trained program (PUBEVAL) after 100.000 generations

Example3: Evolution of a complete robot

- Inter-population Cooperative Co-Evolution
- **Task: Evolve Morphology and Behaviour of a robot**
  - **Structure**
  - **Control**
- **Evaluation: no problem**
  - **Simulation**
  - **But: slow**
- **Problem: Evolution**
  - **All-in-one is possible, but:**
  - **Very large search space**
- **Solution: Co-Evolution**
  - **Divide Problems**
  - **Evolve Structure and Control separately**

GOLEM project, Pollack et. al





Coupling cooperative Co-Evolution

- **Tight coupling (1)**
  - **Body-Individual and Controller-Individuals evolve into 'matched pair'**
  - **Body 'a' only works well with controller 'b'**
- **Option: Two-Part Genotype**
  - **Keep both genotypes in one Individual**
  - **Evolve both together**
- **Tight coupling (2)**
  - **Change in one might depend on change in the other**
  - **Can slow down evolution**
- **Option: delayed evaluation**
  - **Mutate morphology individuals**
  - **Evolve controller individuals (possibly more than one generation)**
  - **Assign fitness to morphology individuals**

Example4: Pattern recognition

- **Intra-population Cooperative Co-Evolution**
- **Taks: Evolve a NN to recognize a set of letters**
  - **Grey values fed into NN**
  - **Output classifies letters**
  - **Classification problem**
- **Co-Evolution: Divide and Conquer**
  - **Different Individuals specialize on different letters**
  - **Group output is combined into one by external mechanism (e.g. voting)**
- **Speciation**

When and Why Co-Evolution?

- **No fitness-function known**
  - **Bootstrapping by co-evolution**
- **Too many fitness cases**
  - **Co-Evolve fitness and cases**
- **Modularizable Problem**
  - **Divide and Conquer**

Other examples

- **Creative Design Systems**
  - **Evolve designs and design specifications**
- **Other Games**
  - **Tic-Tac-Toe**
  - **Prisoner's Dilema**
  - **Checkers**
- **Artificial Life**
  - **Complex simulated Ecosystems**

Summary

- Co-Evolution is Everywhere
- Can be cooperative and competitive
- Can be in one population, or more than one
- An individual's fitness is not fixed in co-evolution
- Co-evolution is not well explored in Evolutionary Computation

References

- Mitchell, M, Introduction to Genetic Algorithms (book), MIT Press, 1996, pp.21—27.
- Pollack, J, Blair, A. and Land, M. (1996). Coevolution of A Backgammon Player. *Proceedings Artificial Life V*, C. Langton, (Ed), MIT Press. Available at: http://www.demo.cs.brandeis.edu/papers/alife5.pdf
- Pollack, Jordan B., Lipson, Hod, Hornby, Gregory S., and Funes, Pablo. Three Generations of Automatically Designed Robots. *Artificial Life*, 7:3, pg 215-223. 2001. Available at: http://www.demo.cs.brandeis.edu/papers/long.html#pollack_alife01
- Darwen, P. J. and Yao, X. (1996). Automatic Modularization by Speciation. *Third IEEE International Conference on Evolutionary Computation*, available at: http://www.demo.cs.brandeis.edu/papers/icec96darwen.ps.gz