University of BRISTOL

Coevolution

Seth Bullock

bristol.ac.uk

University of BRISTOL

Fitness Function Design Is Hard

- As we have seen, a GAs success or failure is often down to our design decisions:
  - Our choice of genetic representation of the phenotype space
  - Our choice of genetic operators and selection scheme
  - Our design of a fitness function
- The third of these is particularly trying:
  - Many fitness functions are hard for GAs to make progress on due to local optima, ruggedness, neutrality, deception, etc.

University of BRISTOL

Coevolution

- But what if we could skip that tricky bit?
- Real evolving populations *do not consult a fitness function* to find out how many offspring each individual is allowed.
  - They just get on with it
  - Organisms co-operate & compete with the other organisms in their environment, and have offspring as part of this activity
- Coevolution: *when multiple populations of evolving organisms influence each other's evolution*

University of BRISTOL

Coevolution

- Coevolution = An automatic fitness function:
  - There is no fitness function designer in nature
- Coevolution = An implicit fitness function:
  - Instead of being set an *explicit* objective target by the fitness function, real organisms are assessed *relative* to their current environment
- Coevolution = A dynamic fitness function:
  - Instead of being assessed against a *fixed* target, real organisms cope with environments that change/evolve over time

University of BRISTOL

Example

- We would like to evolve a robot goalkeeper for RoboSoccer
  - Designing a fitness function has proven to be quite hard
- Instead, we decide to *coevolve* a population of goalkeeper robots against a population of robot penalty takers
  - We start off with random penalty takers and random goalkeepers
  - As soon as penalty takers start to be able to beat random goalkeepers..
  - ..there is pressure on the goalkeepers to get better..
  - ..putting pressure on the penalty takers to get better.. ..and so on.
    - If things work out, coevolution generates a good solution without us having to define a 'good' strategy via an explicit fitness function

University of BRISTOL

Coevolutionary GA

- Co-operative Coevolution:
  - Each population deals with a separate part of the overall problem
  - A solution combines individuals from each population
  - Appropriate when the overall problem is highly structured
- Competitive Coevolution:
  - One population of “solutions” to the problem that we are interested in coevolves against a population of “challengers” that must be defeated
  - Solutions are fit if they defeat many challengers.
  - Challengers are fit if they defeat many solutions.
    - Reminiscent of natural host-parasite or predator-prey coevolution.

University of BRISTOL

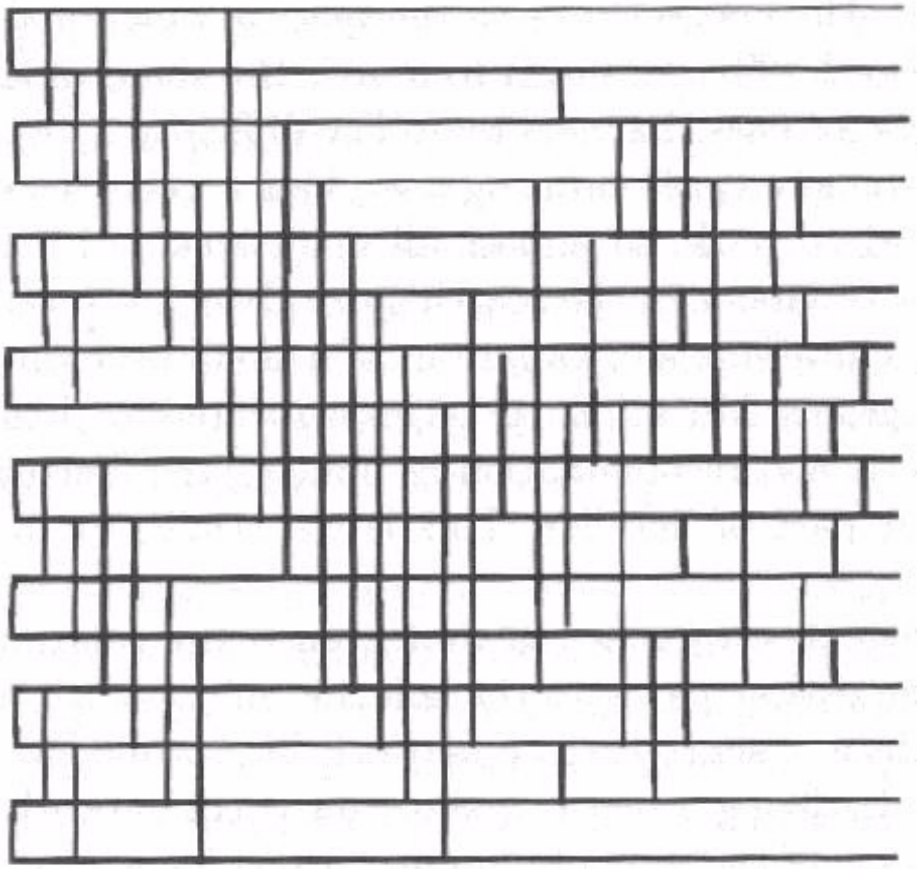
Examples

- The first coevolutionary GA was due to Hillis (1990) and was applied to solving a list-sorting problem.
  - What's the most efficient way to sort a list of length *n*?
  - Hillis pitched a population of 'hosts' (sorting networks) against a population of 'parasites' (each a set of lists to be sorted)
  - He initialised his host population with a reasonable solution and initialised his parasite population with random lists
  - Hosts rewarded for sorting parasite lists; parasites for being unsorted
  - In a few hours, he was able to coevolve a sorter that used 61 'gates'
    - Engineers had taken years to achieve a sorter with 60 gates

University of BRISTOL

List Sorting Example

- History of *n*=16 sorting networks
  - Batcher & Knuth (1964) **63** gates
  - Shapiro (1969) **62** gates
  - Green (1969) **60** gates (best known)
  - Hillis (1990): Evolved **65** gates…
    Hillis (1990): Coevolved **61** gates…
- Used in telecoms switches
- Now also relevant to GPUs
  - Saving even one gate can make a huge performance difference



University of BRISTOL

Why Did Regular Evolution Fail?

Evolution (~65K hosts for 5K generations) had problems:

1. Many local optima: reasonable sorters are surrounded by rubbish
2. Exhaustive assessment is prohibitively slow
   - There are 16! different lists to consider…
   - (Even reducing to 2^{16} binary lists is still a lot to check for each sorter)
3. But assessing against a random sample of lists is also inefficient:
   - Most randomly generated lists are sorted by reasonable sorters
   - Really challenging lists are rare, so there's a lack of pressure to improve

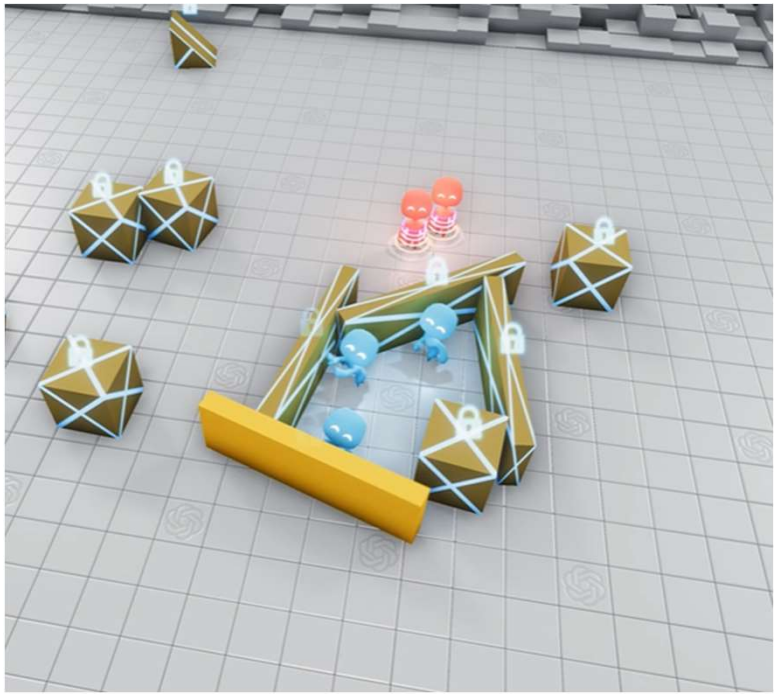
University of BRISTOL

Why Was Coevolution So Much Better?

- Coevolution solved these problems automatically:
  - Instead of random lists, sorters needed to sort sets of lists that were selected to become harder and harder.
  - If sorters got stuck on a local optimum, parasites were under pressure to punish that strategy by finding the lists that it couldn’t sort.
  - When improvements in the sorter population meant selection pressure got weaker, parasites were selected to get still harder.

Recall

From Karl Sims' (1994) *Blockies* to OpenAI's (2019) *Hide & Seek:*



University of BRISTOL

Thank you!