

Lecture 14: Crowding and Co-Evolution

1. Review of fitness sharing
 - (a) Fitness sharing changes the raw fitness.
 - (b) (Explicit) fitness sharing relies on a similarity measure (e.g., distance).
 - (c) Implicit fitness sharing does not use a similarity measure.
2. Crowding, speciation and mating restriction
3. Co-evolution
4. Summary

What Is Crowding

- Crowding techniques insert new individuals into the population by replacing **similar** individuals.
- Crowding techniques strive to maintain the **pre-existing** diversity of a population.
- Crowding techniques do **not** modify fitness.

Deterministic Crowding

```
 $P(0) \leftarrow \text{initialise}();$   
FOR  $t \leftarrow 1$  TO  $g$  DO  
   $P(t) \leftarrow \text{shuffle}(P(t-1));$   
  FOR  $i \leftarrow 0$  TO  $\mu/2 - 1$  DO  
     $p_1 \leftarrow a_{2i+1}(t);$   
     $p_2 \leftarrow a_{2i+2}(t);$   
     $\{c_1, c_2\} \leftarrow \text{recombine}(p_1, p_2);$   
     $c'_1 \leftarrow \text{mutate}(c_1);$   
     $c'_2 \leftarrow \text{mutate}(c_2);$   
    IF  $[d(p_1, c'_1) + d(p_2, c'_2)] \leq [d(p_1, c'_2) + d(p_2, c'_1)]$  THEN  
      IF  $f(c'_1) > f(p_1)$  THEN  $a_{2i+1}(t) \leftarrow c'_1$  FI;  
      IF  $f(c'_2) > f(p_2)$  THEN  $a_{2i+2}(t) \leftarrow c'_2$  FI;  
    ELSE  
      IF  $f(c'_2) > f(p_1)$  THEN  $a_{2i+1}(t) \leftarrow c'_2$  FI;  
      IF  $f(c'_1) > f(p_2)$  THEN  $a_{2i+2}(t) \leftarrow c'_1$  FI;
```

Discussions

- Capable of niching, i.e., locating and maintaining peaks.
- Minimal replacement error (the error of replacing an individual of one class by another from a different class).
- Few parameters to tune.
- Fast because of no distance calculations.
- Population size must be large enough.
- Should use full crossover, i.e., crossover rate = 1.0.

Crowding Methods in General

- Unlike sharing methods, crowding methods do not allocate individuals proportional to peak fitness. Instead, the number of individuals congregating around a peak is largely determined by the size of that peak's basin of attraction *under crossover*.
- Similarity can be measured at either genotypic or phenotypic levels.

Speciation in a Narrow Sense

Speciation in a narrow sense focuses search within a peak.

- A speciation method restricts **mating to similar individuals** and discourages mating of individuals from different species.
- In order to apply such a speciation method, individuals representing each species must be found first. The speciation method **cannot** be used independently.
- Niching and speciation are complementary.
- Similarity can be measured at either genotypic or phenotypic levels.

Mating Restriction: Use Tags

Each individual consists of a tag and a functional string.

| | | | | |
|---------|-------|-------------|--------|------------|
| # 1 # 0 | 10010 | 1010 | | 101 |
|---------|-------|-------------|--------|------------|

template

tag

functional string

- Tags participate in crossover and mutation, but not fitness evaluation.
- Templates can also be used.
- This method has been shown to be effective for multi-modal function optimisation.
- Only individuals with the same tag are allowed to mate.

Mating Restriction: Use Distance

- Define a threshold parameter, σ_{mate} .
- Two individuals are allowed to mate only when their distance is smaller than σ_{mate} .
- EAs with niching and mating restriction were found to distribute the population across the peaks better than those with sharing alone.

Mating restriction is always applied during recombination.

Fitness Sharing by Speciation

- Use tags to identify species (peaks).
- For a given problem, let k be the number of different tags. Let $\{S_0, S_1, \dots, S_{k-1}\}$ be k species of individuals and $\|\cdot\|$ be the cardinality of a set. Then,

$$f_i^{share} = \frac{f_i^{raw}}{\|S_j\|}, \quad i \in S_j, \quad j = 0, 1, \dots, k-1$$

- Recombination occurs only among individuals with the same tag.
- A tag can be mutated.
- No distance is used here.
- This is actually sharing plus mating restriction.

Summary of Niching and Speciation

Fitness Sharing modifies fitness.

- (explicit) fitness sharing
- implicit fitness sharing
- fitness sharing with mating restriction

Crowding is about replacement strategies.

- deterministic crowding

Speciation in a narrow sense occurs during recombination. It is all about mating restriction.

- by tags
- by distances

What Is Co-evolution

The fitness of one individual depends on other individuals.

- The fitness landscape is not fixed, but coupled.
- The same individual may have different fitness in different populations.

Co-evolution can be regarded as a kind of landscape coupling where adaptive moves by one individual will deform landscapes of others, e.g., in prey-predator problems.

Co-evolution here is much closer to biological evolution.

Different Types of Co-evolution

Based on the number of populations involved:

Inter-population co-evolution has two or more populations.

Intra-population co-evolution has only one population.

Based on the relationship among individuals:

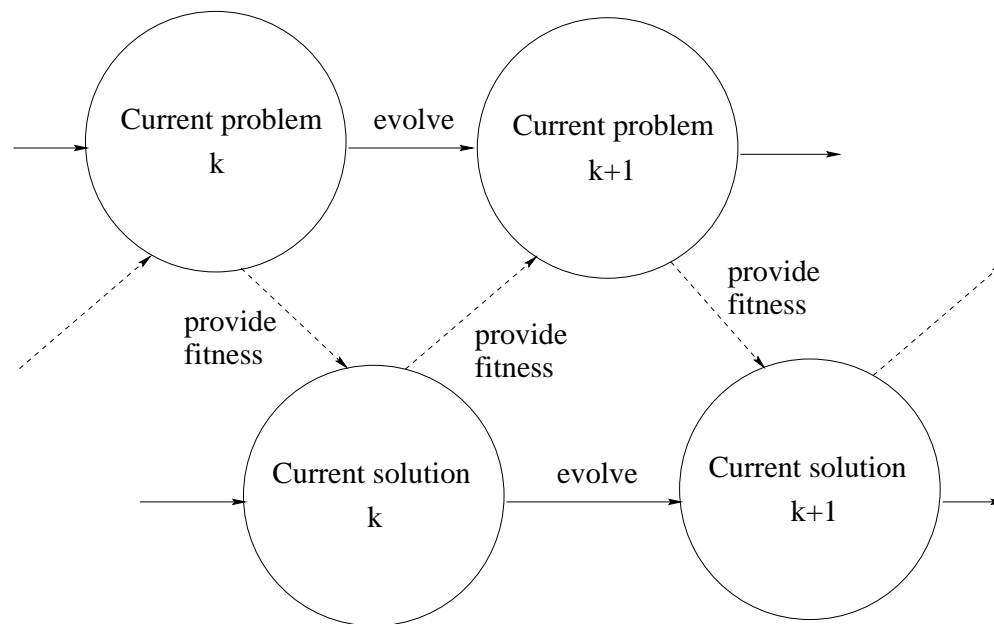
Competitive co-evolution: Individuals are competing against each other.

Co-operative co-evolution: Individuals co-operate with each other in performing a task.

Co-evolution in Design

Inter-population co-evolution.

Many design problems do not have a fixed specification because people change minds and design environments also change.



Co-evolving Programs and Test Cases

1. Given a software specification, one can co-evolve programs along with test cases/data.
2. Furthermore, one can use a similar idea to fix software bugs automatically using co-evolution.
3. The key issue is fitness evaluation.
4. See Reference [4] for more details.

Intra-population Co-evolution: An Example

- There is only one population involved.
- A famous example: TD-Gammon.
 - A grand master level player.
 - Learns by self-playing (i.e., co-evolution)

What make the player so strong: good machine learning algorithm or self-playing?

See Reference [1] for more details.

A Simplified Algorithm

Evolve a neural network that plays backgammon.

1. Let the initial NN be NN_0 , $k = 0$;
2. Generate a mutant challenger of NN_k :

$$w'_{ij} = w_{ij} + \text{Gaussian}(0, \sigma).$$

3. If NN'_k is beaten by NN_k ,
 - then $NN_{k+1} = NN_k$,
 - else $NN_{k+1} = 0.95 \times NN_k + 0.05 \times NN'_k$.
4. $k = k + 1$, go to Step 2 unless the halting criteria are satisfied.

Experimental Results

- Use 197-20-1 fully-connected feed-forward NN. Initial weights were 0's.
- No training for NNs!
- The EA had population size 1!
- There was only one simple mutation — adding Gaussian noise to weights.
- No recombination at all.

Performance: 40% winning against PUBEVAL after 100,000 generations. PUBEVAL is a strong program trained by experts.

Iterated Prisoner's Dilemma Games

Non-zero sum, non-cooperative games.

Player B

| | | C | D |
|----------|---|------|------|
| Player A | C | 3, 3 | 5, 0 |
| | D | 0, 5 | 1, 1 |

How can an EA learn to play the game optimally starting from a population of random strategies?

Evolutionary Approach to IPD Games

Use co-evolution: The fitness of an individual is determined by the payoff it obtains by playing against all other individuals in the population.

Representation of strategies: Each strategy can be encoded as a binary string easily. One could also use NNs or payoff matrices.

Evolutionary Algorithm: Initialised by random strategies.

Results: Cooperation (the optimal strategy in this case) can be evolved easily.

Genotypical Representation of Strategies

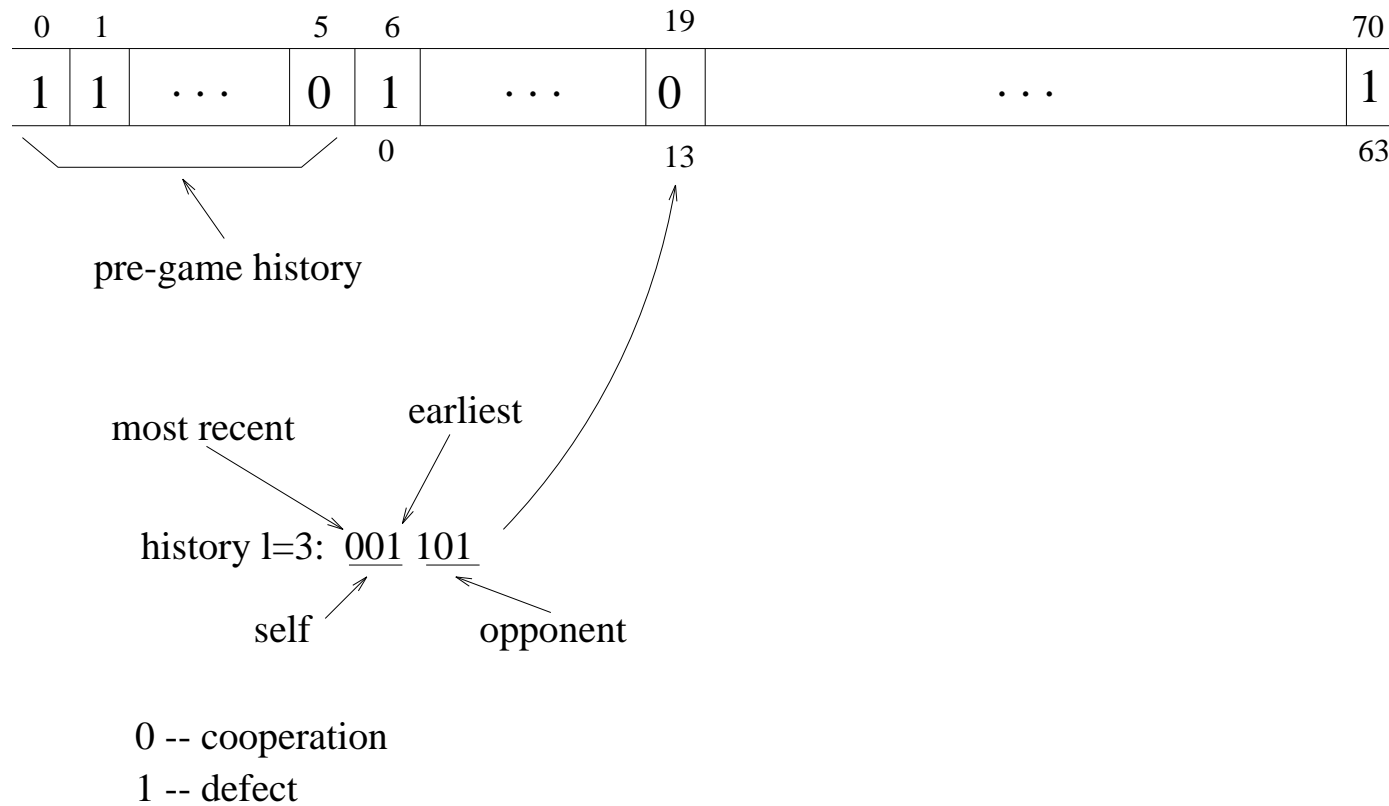


Figure 1: Encoding of strategies, assuming history (memory) length 3.

Speciation by Implicit Fitness Sharing

The implicit fitness sharing implemented in our experiments can be described by Figure 2.

For each strategy i in the EA population, do the following C times:

1. From the EA population, select a sample of σ strategies.
2. Find the strategy in that sample which achieves the highest score (or the largest winning margin, if you prefer) against the single test strategy i .
3. The best in the sample receives payoff. In the case of a tie, payoff is shared equally among the tie-breakers.

Figure 2: Payoff function for implicit fitness sharing.

When and Why Co-evolution

1. We don't know the fitness function.
2. There are too many cases to test in order to obtain a fitness value. Co-evolution can be used to focus search on the difficult part.
3. The fitness is inherently changing in time.
4. Increase and maintain diversity.
5. Improved robustness and fault-tolerance.

More Examples of Co-Evolution

- Classification (see reference [3] for more details) and recognition
- Interactive games and computer-aided learning, e.g., checker, chess, etc.
- Robot morphology and controller
- Many others.

Summary

1. Co-evolution is everywhere. It can occur in a single or multiple populations. It can be used in **many** areas.
2. An individual's fitness is **not** fixed in co-evolution.
3. Co-evolution is **not well understood yet** in evolutionary computation.
4. Artificial Co-evolution and biological evolution.

References

1. Pollack, J, Blair, A. and Land, M. (1996). "Coevolution of A Backgammon Player." *Proceedings Artificial Life V*, C. Langton, (Ed), MIT Press.
2. Juillé, H. and Pollack, J. B. (1996). "Dynamics of Co-evolutionary Learning." *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, Cape Cod, MA, September 9 - 13, 1996, MIT Press, pp. 526-534.
3. Juillé H. and Pollack, J. B. (1996). "Co-evolving Intertwined Spirals." *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, San Diego, CA, February 29 - March 2, 1996, MIT Press, pp. 461-468.
4. A. Arcuri and X. Yao, "Coevolving programs and unit tests from their specification," *Proc. of the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE'2007)*, Atlanta, Georgia, USA, pp.397-400, ACM Press, New York, NY, USA, November 2007.
5. P. Darwen and X. Yao, "Speciation as automatic categorical modularization," *IEEE Transactions on Evolutionary Computation*, **1**(2):101-108, 1997.
6. T. Bäck, D. B. Fogel, and Z. Michalewicz (eds.), *Handbook of Evolutionary Computation*, IOP Publ. Co. & Oxford University Press, 1997. Section C6.1 and Section C6.2. (In the school library)