Evolutionary and Co-evolutionary Learning
Appeal of biology

- Distribution and parallelism
- Robustness to errors, noise
- Adaptation and learning; self-constructing
- Self-organizing resource allocation
- Self-protection and self-repair
- Complex, emergent behavior from simple rules
Problem with biologically inspired systems:

– Once we’ve built them (or they’ve built themselves), can we understand how they work?
Evolution by Natural Selection

- Organisms inherit traits from parents
- Traits are inherited with some variation, via mutation and sexual recombination
- Due to competition for limited resources, the organisms best adapted to the environment tend to produce the most offspring.
- This way traits producing adapted individuals spread in the population in computers

Charles Darwin

John Holland
Real-world applications of genetic algorithms (a few examples)

• Used by GE to automate parts of aircraft design

• Used by pharmaceutical companies to discover new drugs

• Used by John Deere to optimize production schedules

• Used by the London Stock Exchange to automatically detect fraudulent trades

• Used to generate realistic computer animation in the movies *Lord of the Rings: The Return of the King* and *Troy*

• Used to train deep neural networks to play Atari games
Case Study:

Evolution of Collective Computation in Cellular Automata

(Crutchfield et al.)
Cellular automata

- Idealized complex systems
  - large network of simple components
  - limited communication among components
  - no central control
  - complex pattern dynamics from simple rules
Brief history of CAs

• Invented in the 1940s by Stanislaw Ulam and John von Neumann to prove that self-reproduction is possible in machines (and to further link biology and computation).

• Used in early years of computer science as model architecture for massively parallel computation.

• Used up to present for modeling complex systems in physics, geology, chemistry, biology, economics, etc. and for image processing and other parallel applications.
• Current renewal of interest in computer science due to:

  – Molecular and quantum-scale computation (e.g., quantum dot cellular automata; molecular self-assembly of nanoscale electronics)

  – Renewed interest in how biological systems compute (and how that can inspire new computer architectures)

  – A “new kind of science”? 
One-dimensional cellular automata

Rule:
Space-time diagram
ECA 110
A computational task for cellular automata

- Design a cellular automata to decide whether or not the initial pattern has a majority of “on” cells.
  - If a majority of cells are initially on, then after some number of iterations, all cells should turn on.
  - Otherwise, after some number of iterations, all cells should turn off.
How to design a CA to do this?
Packard used cellular automata with 6 neighbors for each cell:

Rule:
A candidate solution that does not work:
local majority voting
Evolving Cellular Automata with Genetic Algorithms
The “chromosome” of a cellular automaton is an encoding of its rule table:
The “chromosome” of a cellular automaton is an encoding of its rule table:

Rule table:

```
   0  0  1  1
   0  0  .  .
   0  0  .  .
   1  1  .  .
   .  .  .  .
   .  .  .  .
   1  1  .  .
```

“Chromosome”: 0 0 0 1 0 0 1 1 . . . . . . . 1
Create a random population of candidate cellular automata rules:

rule 1: 0010001100010010111100010100110111000...
rule 2: 0001100110101011111100011101001010...
rule 3: 1111100010010101000000011100010010101...
   ...
rule 100: 0010111010000001111100000101001011111...
Calculating the Fitness of a Rule

• For each rule, create the corresponding cellular automaton. Run that cellular automaton on many initial configurations.

• Fitness of rule = fraction of correct classifications
For each cellular automaton rule in the population:

**rule 1:** 0010001100010010111100010100110111000...1

Create rule table
Run corresponding cellular automaton on many random initial lattice configurations.

Fitness of rule = fraction of correct classifications

rule 1 rule table:

incorrect

\[
\begin{array}{cccccc}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\end{array}
\]
GA Population:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Binary Code</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>rule 1</td>
<td>0010001100010010111100010100110111100...</td>
<td>0.5</td>
</tr>
<tr>
<td>rule 2</td>
<td>000110011010101111111111000011101001010...</td>
<td>0.2</td>
</tr>
<tr>
<td>rule 3</td>
<td>11111000100101010000000011100010010101...</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Select fittest rules to reproduce themselves

<table>
<thead>
<tr>
<th>Rule</th>
<th>Binary Code</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>rule 1</td>
<td>0010001100010010111100010100110111100...</td>
<td>0.5</td>
</tr>
<tr>
<td>rule 3</td>
<td>11111000100101010000000011100010010101...</td>
<td>0.4</td>
</tr>
</tbody>
</table>

...
Create new generation via crossover and mutation:

Parents:
rule 1: 0010001 \times 100010010111100010100110111000...
rule 3: 1111100 \times 010010101000000011100010010101...
Create new generation via crossover and mutation:

Parents:
rule 1: 0010001 × 100010010111100010100110111000...
rule 3: 1111100 × 010010101000000011100010010101...

Child:
0010001010010101000000001110001001010101...
Create new generation via crossover and mutation:

Parents:
rule 1: \[0010001 \times 100010010111100010100110111000\ldots\]
rule 3: \[1111100 \times 0100101010000000011100010010101\ldots\]

Child:
\[0010001 \boxed{0} 100101010000000011100010010101\ldots\]
Create new generation via crossover and mutation:

Parents:
rule 1: 0010001 × 100010010111100010100110111000...
rule 3: 1111100 × 010010101000000011100010010101...

mutate

Child:
00100000100101010000000011100010010101...
Create new generation via crossover and mutation:

Parents:

rule 1: \[0010001 \times 100010010111100010100110111000\ldots\]
rule 3: \[1111100 \times 010010101000000011100010010101\ldots\]

Child:

\[00100000100101010000000011100010010101\ldots\]

Continue this process until new generation is complete.
Then start over with the new generation.

Keep iterating for many generations.
A cellular automaton evolved by the genetic algorithm
How do we describe information processing in complex systems?
David Marr’s levels of explanation for information-processing systems

• **Computation Theoretic:** What are the goals of the computation?

• **Algorithmic:** How are the input and output represented, and how is the input transformed into the output?

• **Implementational:** How can the algorithm be physically realized?
"simple patterns": black, white, checkerboard
Simple patterns filtered out
### Regular Domains

<table>
<thead>
<tr>
<th>$\Lambda^0 = 0^*$</th>
<th>$\Lambda^1 = 1^*$</th>
<th>$\Lambda^2 = (01)^*$</th>
</tr>
</thead>
</table>

### Particles (Velocities)

| $\alpha \sim \Lambda^0 \Lambda^1 (0)$ | $\beta \sim \Lambda^1 01 \Lambda^0 (0)$ |
| $\gamma \sim \Lambda^0 \Lambda^2 (-1)$ | $\delta \sim \Lambda^2 \Lambda^0 (-3)$ |
| $\eta \sim \Lambda^1 \Lambda^2 (3)$   | $\mu \sim \Lambda^2 \Lambda^1 (1)$   |

### Interactions

| decay       | $\alpha \rightarrow \gamma + \mu$ |
| react       | $\beta + \gamma \rightarrow \eta, \mu + \beta \rightarrow \delta, \eta + \delta \rightarrow \beta$ |
| annihilate  | $\eta + \mu \rightarrow \emptyset_1, \gamma + \delta \rightarrow \emptyset_0$ |

---

laws of

"particle physics"

---

"particles"
Level of particles can explain:
- Why one CA is fitter than another
- What mistakes are made
- How the GA produced the observed series of innovations

Particles give an “information processing” description of the collective behavior

→ “Algorithmic” level
How the genetic algorithm evolved cellular automata

generation 8

generation 13
How the genetic algorithm evolved cellular automata

generation 17

generation 18
How the genetic algorithm evolved cellular automata

generation 33

generation 64
Another Task: Synchronization
laws of
“particle physics”

Hordijk, Crutchfield, and Mitchell: Can model CAs in terms of particle “kinematics”

(note “condensation phase”)

particle model of CA
results of particle model

particle model of CA
Co-evolutionary Learning
Problem for learning algorithms:

How to select training examples appropriate to different stages of learning?

One solution:

Co-evolve training examples, using inspiration from host-parasite coevolution in nature.
Host-parasite coevolution in nature

- Hosts evolve defenses against parasites
- Parasites find ways to overcome defenses
- Hosts evolve new defenses
- Continual “biological arms race”
Heliconius-egg mimicry in Passiflora

http://www.ucl.ac.uk/~ucbhdjm/courses/b242/Coevol/Coevol.html
• Darwin recognized the importance of coevolution in driving evolution

• Coevolution was later hypothesized to be major factor in evolution of sexual reproduction
Coevolutionary Learning

Candidate solutions and training examples coevolve.

– **Fitness of candidate solution (host):** how well it performs on training examples.

– **Fitness of training example (parasite):** how well it defeats candidate solutions.
Sample Applications of Coevolutionary Learning

– Game playing strategies (e.g., Rosin & Belew; Fogel; Juillé & Pollack)

• Hosts: Candidate strategies for Nim, 3D Tic Tac Toe, backgammon, etc.

• Parasites: Another population of candidate strategies
- HIV drug design (e.g., Rosin)

  • Hosts: Candidate protease inhibitors to match HIV protease enzymes

  • Parasites: Evolving protease enzymes
– Robot behavior (e.g., Sims; Nolfi & Floreano)

• Hosts: Robot control programs

• Parasites: Competing robot control programs
• Cooperative:

  – Cooperative coevolution of neural network weights and topologies (e.g., Potter & De Jong; Stanley, Moriarty, Miikkulainen)

Here I’ll focus on “host-parasite” competitive coevolutionary learning.
Why should we expect competitive coevolutionary learning to work?

Hypotheses:

1. Allows arms races to emerge, with the evolution of training examples targeted to weaknesses in learners, and subsequent adaptation of learners to those training examples, and so on.

2. Helps maintain diversity in the population

Effect: Increases success of learning while reducing the amount of training data needed
These hypotheses are plausible but have been largely untested in work on coevolution.
Practical problems observed in coevolutionary learning

- Cycling:

low "true" fitness
- Loss of gradient for hosts
• Over-“virulence” of parasites
Hypothesis

Distributing host and parasite populations in space will overcome these impediments by:

– Preserving diversity in the populations

– Fostering arms races between hosts and parasites
Experiments
(Mitchell, Thomure, & Williams, 2006)

Spatial  Non-spatial

Coevolution

Evolution
Evolving cellular automata

- Problem is to design 1D CA that classifies initial configurations (ICs) as "majority 1s" or "majority 0s".
• Hosts are candidate CA rule tables

• Parasites are initial configurations to be classified
Spatial Coevolution
Spatial Coevolution

- 2D toroidal lattice with one host \((h)\) and one parasite \((p)\) per site
Spatial Coevolution

- 2D toroidal lattice with one host ($h$) and one parasite ($p$) per site

Each $h$ is replaced by mutated copy of winner of tournament among itself and 8 neighboring hosts.

Each $p$ is replaced by mutated copy of winner tournament among itself and 8 neighboring parasites.

fitness($p$) = \[
\begin{cases}
0 & \text{if } h(p) \text{ is correct} \\
> 0 & \text{if } h(p) \text{ is not correct}
\end{cases}
\]

fitness($h$) = fraction of 9 neighboring $p$ answered correctly
Non-Spatial Coevolution
Non-Spatial Coevolution

- No spatial distribution of host and parasite populations

  Each $h$ is replaced by mutated copy of winner of tournament among itself and 8 randomly chosen hosts.

  Each $p$ is replaced by mutated copy of winner tournament among itself and 8 randomly chosen chosen parasites.

  \[ \text{fitness}(h) = \text{fraction of 9 } p \]

  (randomly chosen from parasite population)

  answered correctly

  \[ \text{fitness}(p) = \begin{cases} 
    0 & \text{if } h(p) \text{ is correct} \\
    > 0 & \text{if } h(p) \text{ is not correct}
  \end{cases} \]

  for host $h$ randomly chosen from host population
• **Spatial Evolution:**

  – Same as spatial coevolution, except parasites don’t evolve.

  – A new population of random parasites is generated at each generation.
• **Non-Spatial Evolution:**

  – Same as non-spatial coevolution, except parasites don’t evolve.

  – A new sample of 100 random parasites is generated at each generation.

  – Fitness of a host is classification accuracy on these 100 randomly generated parasites
## Results

<table>
<thead>
<tr>
<th></th>
<th>Function Induction</th>
<th>Cellular Automata</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spatial Coev.</strong></td>
<td>78% (39/50)</td>
<td>67% (20/30)</td>
</tr>
<tr>
<td><strong>Non-Spatial Coev.</strong></td>
<td>0% (0/50)</td>
<td>0% (0/20)</td>
</tr>
<tr>
<td><strong>Spatial Evol.</strong></td>
<td>14% (7/50)</td>
<td>0% (0/30)</td>
</tr>
<tr>
<td><strong>Non-Spatial Evol.</strong></td>
<td>6% (3/50)</td>
<td>0% (0/20)</td>
</tr>
</tbody>
</table>

Percentage of successful runs
In short: Spatial coevolution significantly out-performs other methods on both problems
Analysis

Why was spatial coevolution successful?

Hypotheses:

1. Maintains diversity over long period of time

2. Creates extended “arms race” between hosts and parasite populations

Here we examine these hypotheses for the CA task.
The GA evolves four types of strategies:

- Random
- Default
- Block Expanding
- Particle
“Default” strategy
(performance ≈ 0.5)
“Block-expanding” strategy
\((0.55 \leq \text{performance} < 0.72)\)
“Particle” strategy
(performance ≥ 0.72)
**Measuring diversity in host population**

- Plot distribution of host strategies in typical CA runs

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>Very low performance</td>
</tr>
<tr>
<td>Default</td>
<td>Low performance</td>
</tr>
<tr>
<td>Block expanding</td>
<td>Medium performance</td>
</tr>
<tr>
<td>Particle</td>
<td>High performance</td>
</tr>
</tbody>
</table>
Spatial Coevolution
(typical run)
Non-Spatial Coevolution
(typical run)
Spatial Evolution
(typical run)
Non-Spatial Evolution
(typical run)
Summary of diversity results

- Spatial coevolution seems to preserve more diversity in the host population than the other methods.

- In the other methods:
  - **Spatial methods:** One strategy quickly comes to dominate the population
  - **Nonspatial methods:** Population oscillates between two different strategies
Arms races between the host and parasite populations
Figure 8: Performance of best CA in the population on the 400 parasites at the same generation (dashed) and on 400 randomly generated initial configurations, for the spatial coevolutionary run. The data is plotted every 10 generations.
Arms races between the host and parasite populations

• Hypothesis: Initial configurations are evolving “deceptive blocks”:
Arms races between the host and parasite populations

• Hypothesis: Initial configurations are evolving “deceptive blocks”:

  – Occurrence of a block of seven or more adjacent 0s or 1s in the 149-bit IC

  – Probability of such an occurrence in a randomly generated string of the same density is less than 0.01.
• For a typical run of spatial coevolution, we plotted the fraction of parasites with deceptive blocks
Specialization of ICs

deceptive blocks

generations

default
block-expander
particle
Arms Races During Particle Epoch

Close-up of performance on parasites and on random ICs

- Performance on parasites at this generation
- Performance on random ICs

Generation:
- Gen. 2285
- Gen. 2248

Performance:
- 1.2
- 1
- 0.8
- 0.6
- 0.4
- 0.2
- 0

Generation range: 1600 to 2300
Figure 18: Left: Example of a misclassification made by the highest-performing CA at generation 2248 on a parasite at the same generation. Right: Correct classification of same parasite by the highest-performing CA at generation 2285. Greek letters indicate particles, described in text.
Summary:
Why does spatial coevolution improve performance?

• Maintains diversity in population

• Produces “arms race” with ICs targeting weaknesses in CAs (e.g., block expanding) and CAs adapting to overcome weaknesses (e.g., particle strategies)

• Note that spatial coevolution requires significantly fewer training examples for successful learning
Future work

• Theory
  – Generality of results
  – Better characterizations of diversity and arms races
  – Better understanding of role of spatial distribution of populations

• Theory of dynamics on networks

• Effects of different interconnect topologies
Possible applications to real-world problems
Possible applications to real-world problems

– Drug design to foil evolving viruses/bacteria

– Coevolving software/hardware with test cases

– Evolving game-playing programs

– Coevolving computer security systems with possible threats
References


