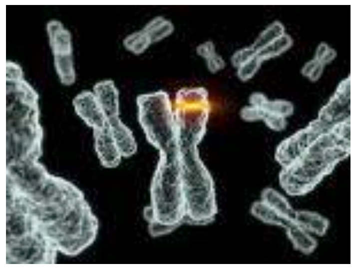
CS 289
Evolutionary Computation



Search, Optimization, Evolution

Problem Solving as Search

- Classic AI way of thinking (e.g McCarthy, Newell and Simon, 1950s)
- Wide domain can be cast this way (planning, theorem-proving, puzzles)
- Methods: optimal/complete search vs local search
- *But Some Problems are Provably Hard (NP-complete, 1973)*
- *And No Search Algorithm is the Best (No Free Lunch Theorem, 1995)*

- Good approach when i
  but *easy to check* if a so
  - What's are some exa

Brief Timeline
Metropolis algorithm (1953)
Darthmouth Ai conference (1950)
Genetic Algorithms (John Holland 1970s)
NP-completeness first understood (1973)
Linear Programming proven to be polynomial time (1979)
Simulated Annealing (1980s)
Genetic Programming and Ant Colony Optimization (~1990)
No Free lunch theorem 1995

Evolution: Darwin/Wallace 1859 Mendel 1900 DNA 1953

Casting Problems as Search

F(solution) = objective function to maximize/minimize
$F(x) = 2^{(-2(x-0.1)/0.9)^{\wedge}2} * ((\sin(5pix))^6$
F(x,y,z) = some complex but differentiable equation (classic optimization)
(e.g. GPS localization: position relative to a set of reference nodes)

F(v1, v2, v3…..vn) = no longer an equation!!!
How well a neural network with these weights classifies some images
How well do these feature detector parameters capture objects of interest
How well these 1991 stock allocations would have done in 2012
How well do these allocations maximize total agent utilities (class lottery!)

F(circuit/program) = no longer a simple vector/parameter representation!!
How well does this circuit solve the required task?
How well does a robot with this program navigate

F(fruitfly genome A) = how well does this individual survive compared to its buddies
Evolution as a search process over a very complex representation….

Local Search

Local Search

- Many Variants (Hill climbing, Simulated Annealing, K-beam)
- Exploitation vs Exploration
- Representation, Representation, Representation!!
  - Solution , Objective, Neighborhood function

Evolutionary Computation

Evolutionary Search is a *cooperative local search* method

Based on the belief that

= Evolution is a kind of optimization over a very complex landscape

= The Genotype-Phenotype separation works across all organisms

Key Features (GA/GP)

- Population (Always have many candidate solutions)
- Representation (Genes/Programs; Fitness function)
- Variation (Nbr function = cross-over between solutions!)
- Selection (Choose best solutions across all new candidates)

Simple model: Modern EvoBio looks at much more!

Some “Simple” Examples

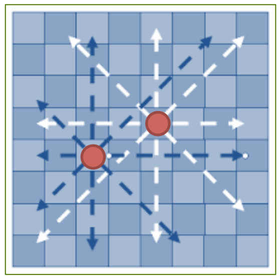
- N-queens
- EvoLISA (image compression)
- Evolving Cellular Automata (Crutchfield, Mitchell)

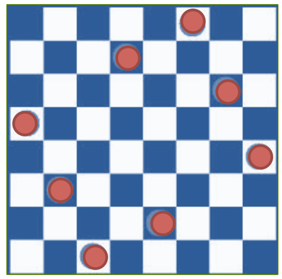
- Next lectures!
  - Evolving Lego bridges (Jordan Pollack’s Lab)
  - Evolving Robot Bodies and Brains (Pollack and Lipson)
  - Evolving Swarm behaviors.

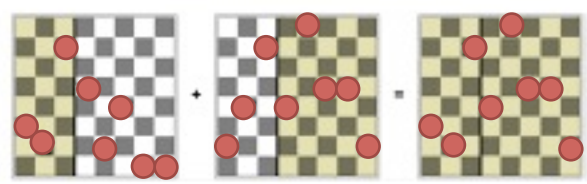
N-Queens Example

On a 8x8 board, place 8 queens, such that they can't kill each other

How do we cast this as an evolutionary computation?



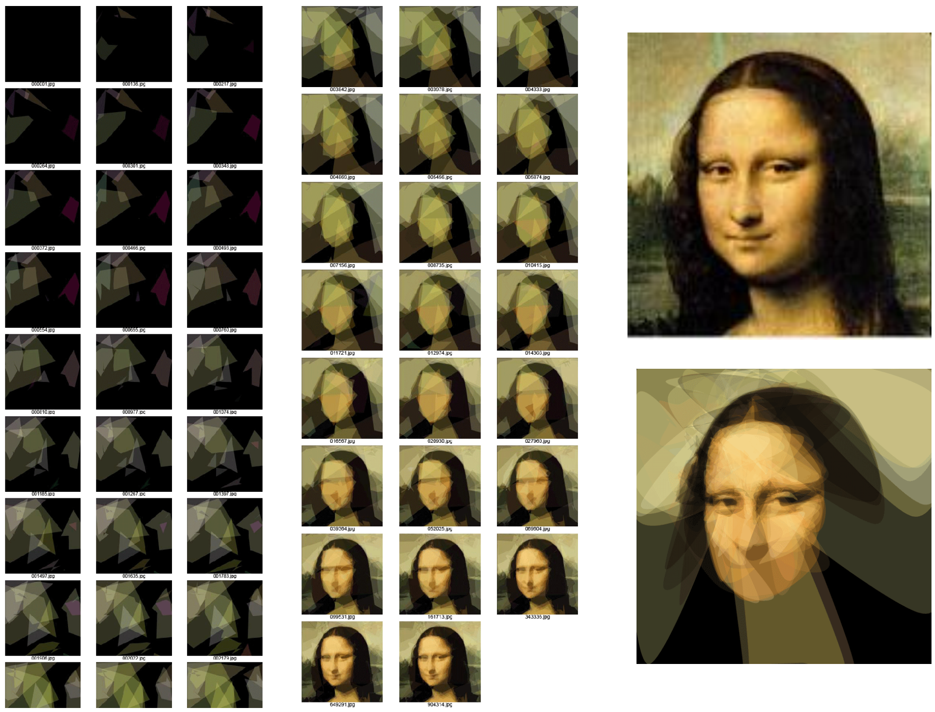




EvoLISA

(Roger Asling, 2008)

- Image Compression
  - Representation: 50 semi-transparent "polygons"
    - DNA = "vector of attributes"
  - Trying to find the best "DNA" to capture a given image
    - Not obvious how to find the right answer, but relatively easy to evaluate a given answer (how well does it match a given image)
    - Compression: Hard to compress, but easy to decompress!
  - Method: Genetic algorithm
    - Population | Variation| Selection
  - Question: Can we evolve a rep. of Mona Lisa?
    - Lets take a look



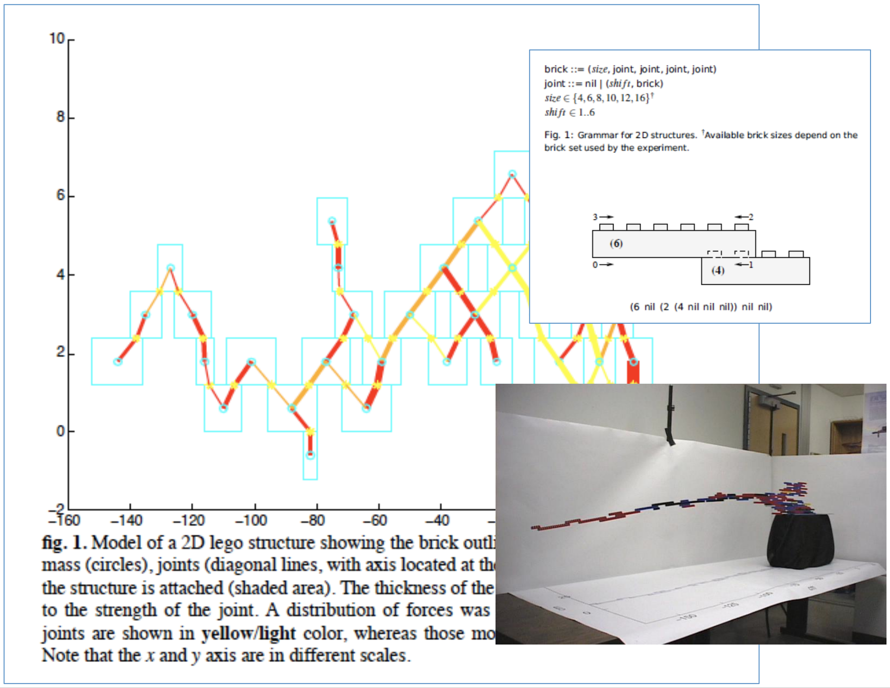
Evolving "Architectures" in Lego

(Pablo Funes and Jordan Pollack, ~1999)

- **Next Week's Topic**

- **EvoCAD**
  - Using evolution as a tool for exploring the design space
  - Fitness function: what is the goal of the structure (or could be human directed evolution)

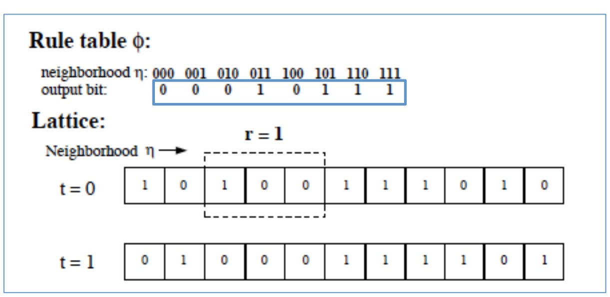
- **Bridges, Cranes, and other structures**
  - Specific goal (fitness of the structure)
  - Representation: how do I describe the current structure and generate new structures (variation)



Evolving Cellular Automata

(Melanie Mitchell and John Crutchfield 1994)

- Evolving a CA to do "computation"
  - E.g. Density, Synchronization, Pattern formation
- Representation: simple and cool idea
  - "rule table" as a "binary string"



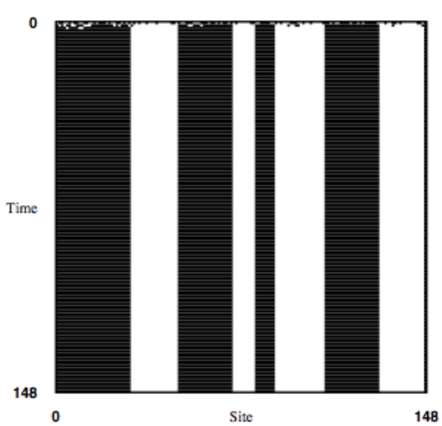
Evolving Programs!

Implementation Example

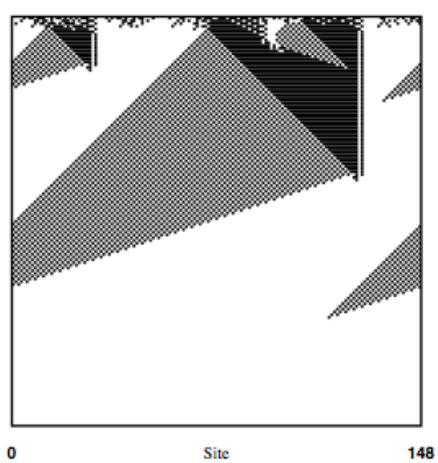
- Problem: Classify Density (if black > ½ , then CA turns all black)
  - Binary Cellular Automata Ring, N=149, CA Rule radius r=3
  - Claim: no fixed radius rule exists that works for different lattice sizes, but even for a fixed lattice size it is hard to find a rule of low radius (not proven impossible)
- Representation: 128 bit string, population of 100 Candidates
- Variation: Cross-over and Mutation
- Selection: Fitness F100 (performance on 100 randomly generated CAs)

- How it works
  - Start with 100 candidate solutions
  - Test fitness F100 for all of them
  - Pick top 20 (elite set)
  - Generate another 80 by cross-over of randomly selected elite parents, do two mutations in each offspring (did not use the roulette wheel)
  - Repeat

- Interesting Results
  - Naïve methods: Majority, Expand-block
  - Sophisticated Method: “particle” method
  - Also looked at other “computation” problems (like synchronization)



Majority solution



Evolved “Particle-based” solution

Cooperative Transport

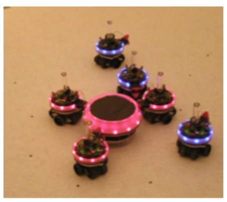
- Ant-inspired Robots = Swarms Bots

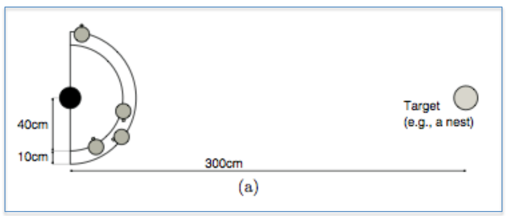
  How should they coordinate to move something whose weight and size they don't know apriori?

  - Individual Robot Capabilities
    - Can see object and goal direction from a distance
    - Don't know weight or size of object
    - Can see other robots ring color, and flash a color themselves
    - Can grip an object (or each other) and move in any direction.

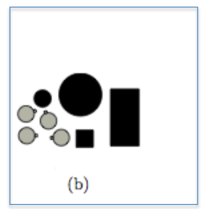
Activity: Find a solution for robots using GAs

Representation | Variation | Selection





(a)


(b)

Cooperative Transport

(Roderich Gross, Marco Dorigo, 2004-2006)

- Representation:
  - Neural Network with hidden nodes
- Variation: Perturb Weights
- Selection:
  - Simulation over a random objects
  - **Need a Complex Fitness Function:**
    - Assembly performance (just attaching gets points) and Distance gain