Evolutionary Design: Lessons from Biology

Kenneth De Jong
George Mason University

EA Applications

- Recurring theme:
  - Evolving parameters: ok
  - Evolving structures: harder
  - Evolving programs: hardest

Evolutionary Design

- Design optimization:
  - Difficult, but well-understood
  - Primarily numerical parameter optimization
  - Some discrete optimization
  - Important focus: multi-objective optimization

Evolutionary Design

- Conceptual design:
  - Difficult, not well-understood
  - Primarily non-numeric in nature
  - Premium on novelty

Evolutionary Design

- Conceptual design via EAs:
  - Represent, evolve complex structures
  - Satisfy highly non-linear constraints
  - Not effectively characterized as parameter optimization

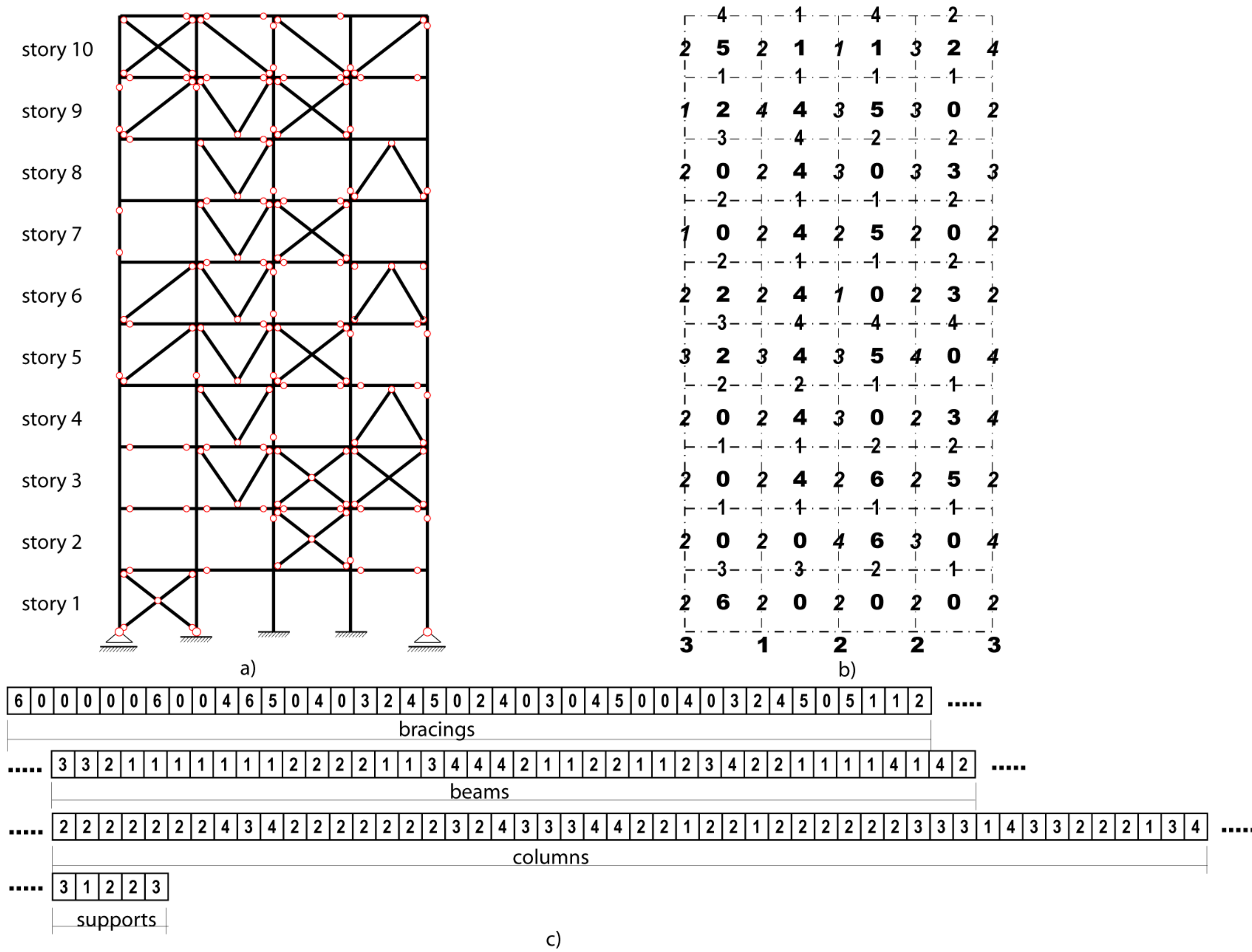
Evolutionary Design

- Thesis:
  - Standard EAs are good for design optimization.
  - Different EAs are needed for conceptual design.
    - Examples in this conference

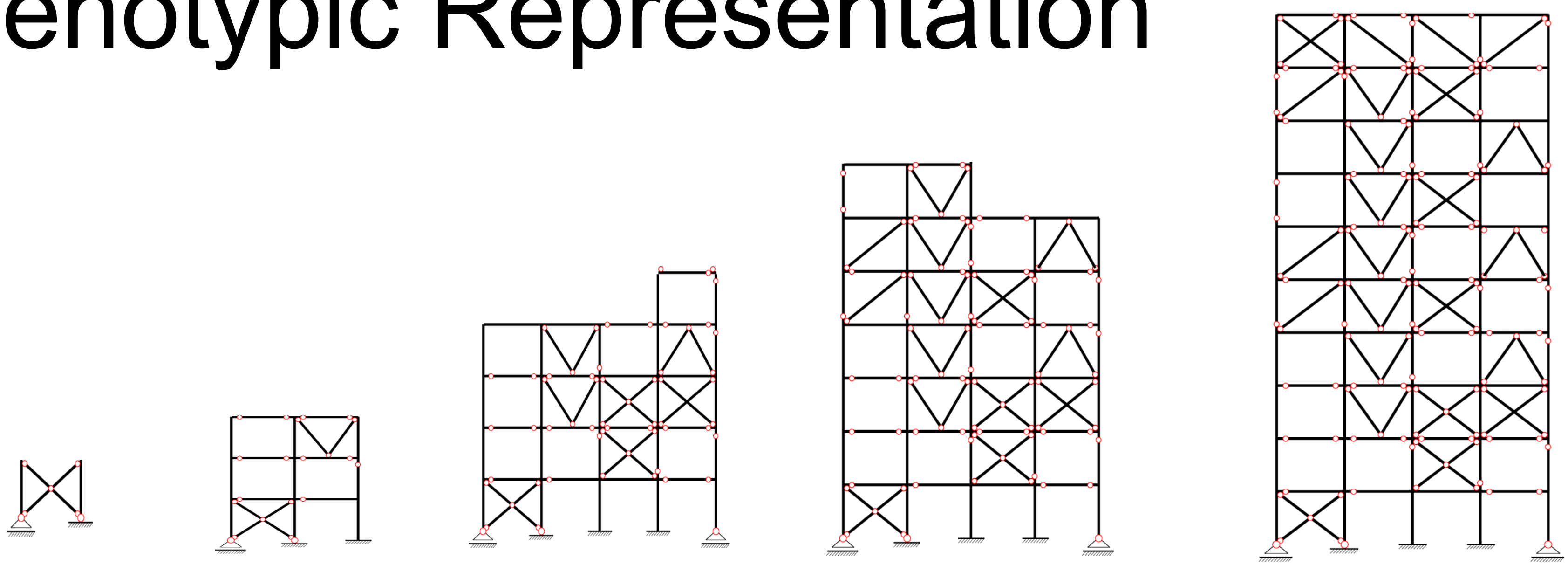
Evolutionary Design

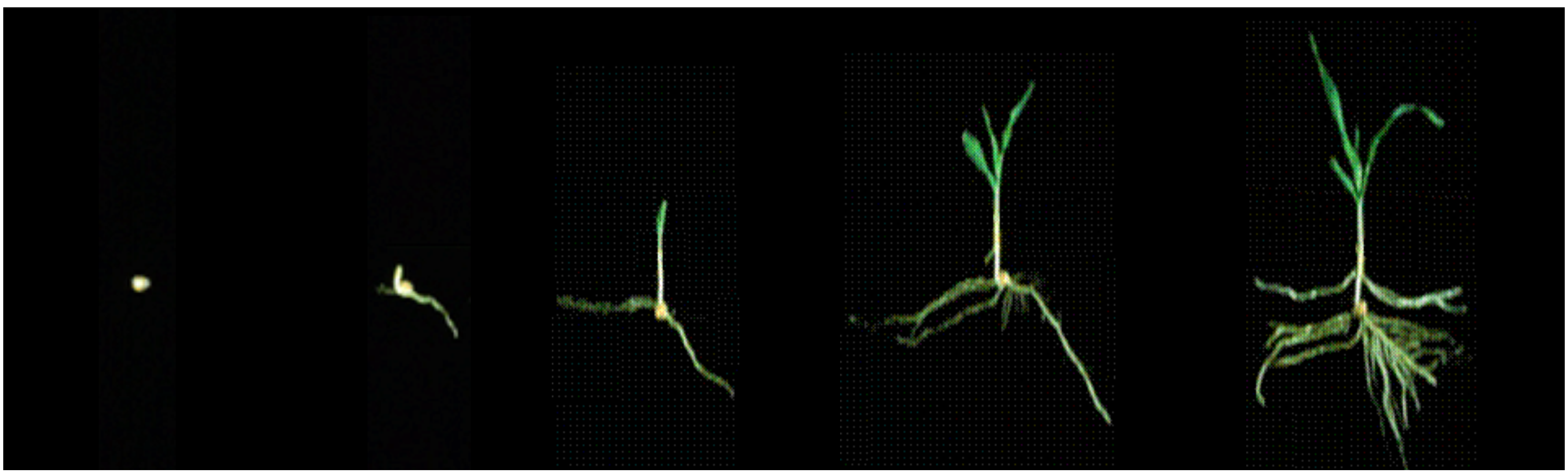
- Inspiration from biology:
  - Strong distinction between:
    - Genotype (plans)
    - Phenotype (objects)
  - Evolve plans, not objects
  - Morphogenesis: plans ==> objects

Phenotypic Representation



Genotypic Representation





Genotypic Representations

- "Dawkins strategy":
  - Hand-design a parameterized generator
  - Evolve parameters …
  - Issues:
    - Strong search bias, lack of novelty

Genotypic Representations

- More open-ended strategy:
  - Evolve the generator!
    - I.e., evolve programs …
  - Issue:
    - How to do so effectively?

Generative Representations

- “Inspiration from nature” strategy has yielded a variety of computation models:
  - L-systems
  - Cellular automata
  - Genetic regulatory networks
  - …

Generative Representations

- L-systems:
  - Basic definitions
  - Examples
  - Issues

L-systems

- Short for Lindenmayer systems
  - Developed ~ 1968 by A. Lindenmayer
  - Goal: model plant morphology
  - Approach: formal “rewrite” grammars:

L-systems

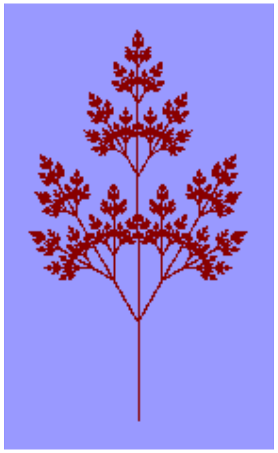
- Elements:
  - Set of interpretable symbols
  - Initial string of symbols (axiom/seed)
  - Rewrite rules (productions)

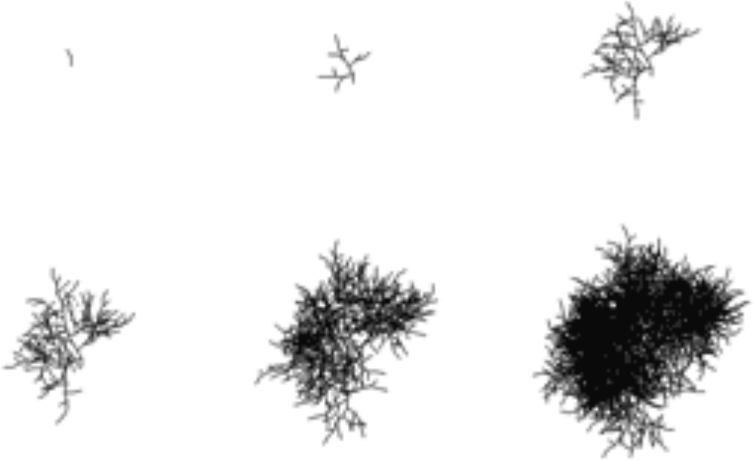
L-systems

- Example:
  - Symbols: A, B
  - Seed: A
  - Rules: A->B, B->AB
    - t=0: A
    - t=1: B
    - t=2: AB
    - t=3: BAB
    - t=4: ABBAB
    - t=5: BABABBAB
    - …

L-systems

- Examples:





L-systems

- Properties:
  - Compact “syntactic” model
  - Strong contact with formal grammars:
    - Context free: A -> B
    - Context sensitive: BAB->BCB
    - Non-deterministic rewriting
  - Strong contact with computation theory
    - Finite state machines, automata theory, Turing machines

L-systems

- Issues:
  - Requires a termination criterion
  - Interaction with the environment
  - Requires a domain-specific interpretation

  - 2D, 3D?
    - via interpretation
    - via 2D, 3D strings

  - Design:
    - Hand-construct grammar
    - Evolve it

L-systems

- Evolutionary design:
  - Given:
    - symbol set, axioms, string interpreter, fitness function
  - Find set of rewrite rules with high fitness
  - Examples:
    - Artificial vegetation (Ochoa, Jacob, Moch, …)
    - Tables (Popovici)
    - Robot morphology (Hornby, Pollack, Bentley, …)
    - Neural networks (Kitano, …)

L-systems

- Evolutionary design issues:
  - Complexity of the rules
    - Context free, context sensitive, …
  - Representation of rule sets
    - Fixed vs. variable length genome
    - Individuals as rules
  - Reproductive operators
    - Mutation?
    - Recombination?

Generative Representations

- Cellular automata:
  - Basic definitions
  - Examples
  - Issues

Cellular Automata

- Similar in spirit to L-systems:
  - Symbols, axioms, rewrite rules
  - Explicit bounded spatial topology
- Classical examples:
  - Conway’s Game of Life
  - Wolfram’s pattern generators

Cellular Automata

- Elements:
  - A topology of cells:
    - E.g., 1D array of N cells
  - Initial state of each cell
  - State transition rewrite rules:
    - E.g., $S_{i,t+1}$ <-- $S_{i-1,t}$ $S_{i,t}$ $S_{i+1,t}$

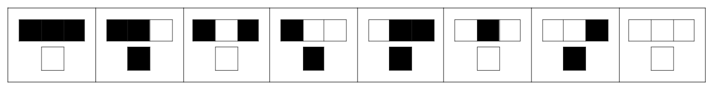
Cellular Automata

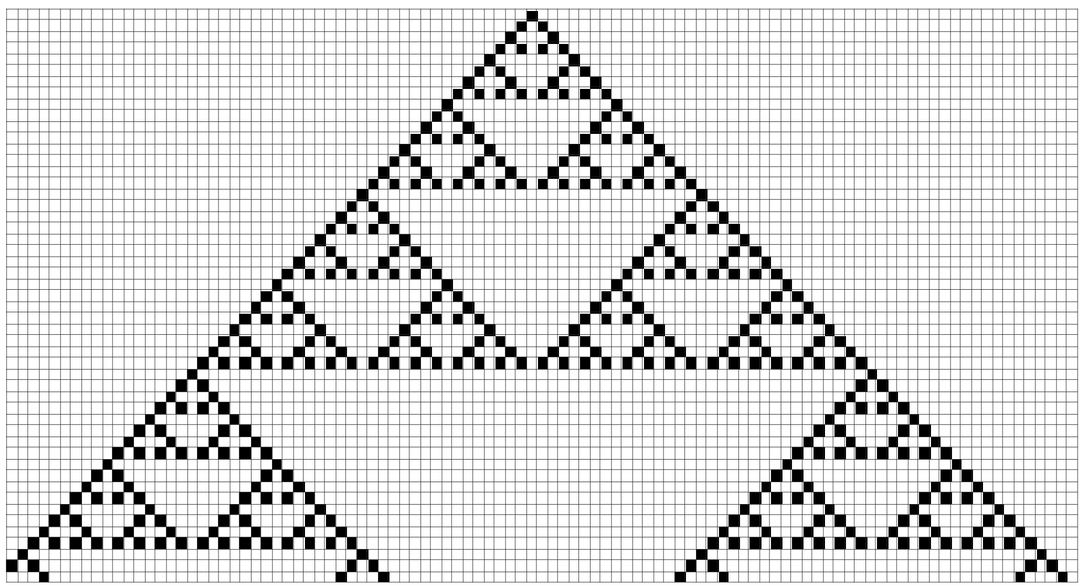
- Example: 1D binary CAs
  - 1D array of N cells (torus)
  - Initial state of each cell (0/1)
  - Rewrite rules:
    - Same for every cell (homogeneous)
    - Parallel synchronous updates

Cellular Automata

- E.g.:
  - symmetric neighborhood of diameter 1 => 2^3 = 8 rules
    - 000 -> 0, 001 -> 1, 010 -> 0, 011 -> 1
    - 100 -> 1, 101 -> 0, 110 -> 1, 111 -> 0
  - T=0: 0001000
  - T=1: 0010100
  - T=2: 0100010
  - T=3: 1010101
  - T=4: 1000001
  - T=5: 1100011
  - …

Cellular Automata





Cellular Automata

- Properties:
  - Compact “syntactic” representation
  - Strong contact with formal grammars, computation theory
  - Explicit spatial representation
    - Simplifies interpretation
    - Simplifies termination
    - Constrains space of generated objects
  - Design:
    - By hand
    - Evolve them

Cellular Automata

- Evolutionary design:
  - Given:
    - symbol set, axioms, spatial topology, fitness function
  - Find set of rewrite rules with high fitness
  - Examples:
    - Artificial societies (Axtell, Ilichinski, …)
    - Programs (Mitchell, Crutchfield, Sipper, …)
    - Chemical structures (Gerhardt, Schuster, …)
    - Building designs (Kicinger, Arciszewski, …)

Cellular Automata

- Evolutionary design issues:
  - Representation of rule sets
    - Simpler than L-systems: just fixed-length RHS
    - Explodes combinatorially with:
      - Neighborhood length, number of symbols, CA dimension
  - Reproductive operators
    - Mutation?
    - Recombination?

Generative Representations

- Genetic regulatory networks:
  - Basic definitions
  - Examples
  - Issues

GRNs

- Similar in spirit to L-systems, CAs:
  – Symbols, axioms, rewrite rules
  – Explicit gene/cell model
    - Genes: independent rewrite rules
    - Cells: collections of genes + shared global memory

GRNs

- Also, similar in spirit to:
  - AI blackboard models
  - Holland's classifier systems

GRNs

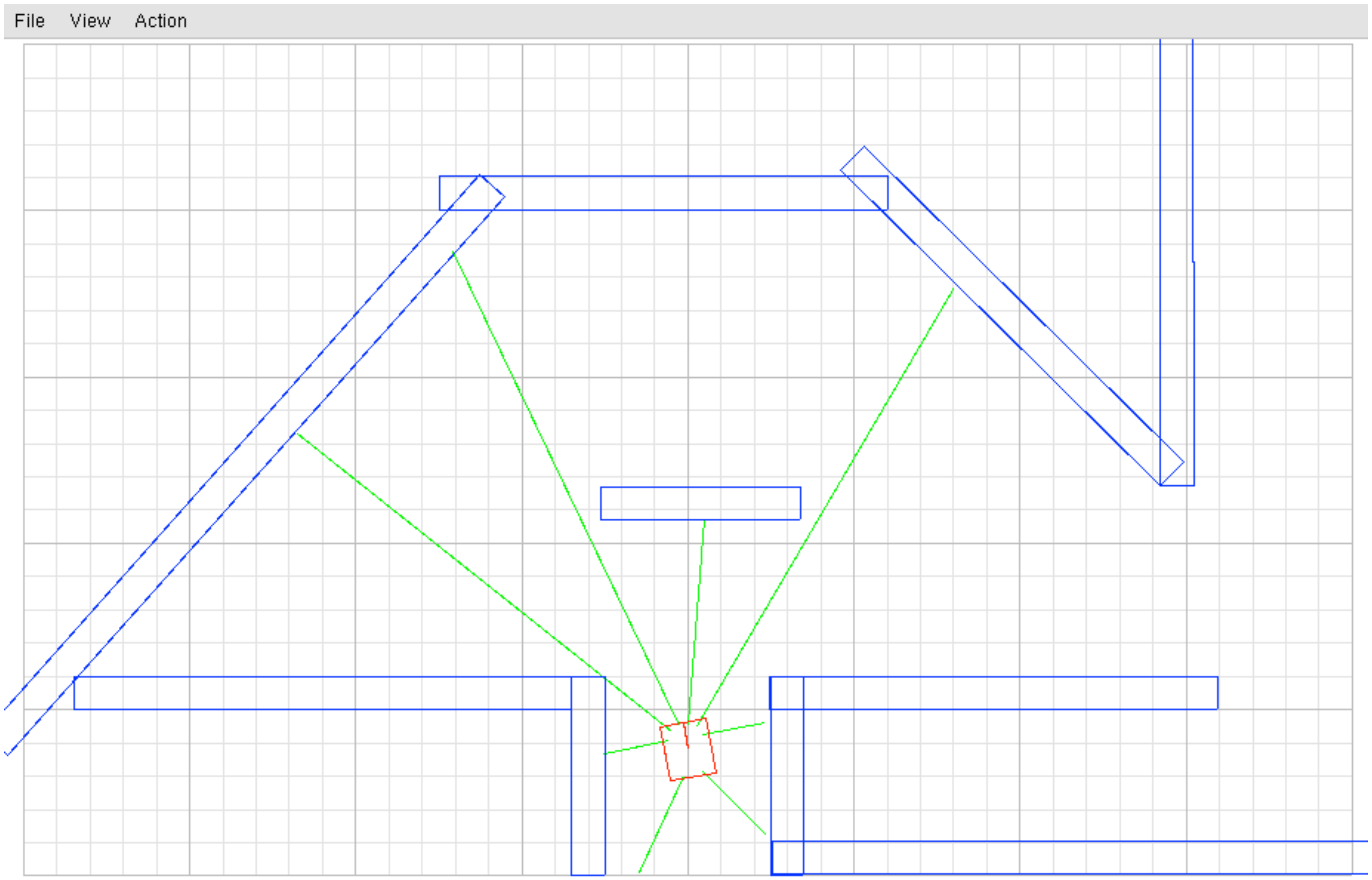
- Elements:
  - A cell with N genes and shared memory
  - Initial state of memory
  - Genes (rewrite rules):
    - LHS: match memory elements
    - RHS: Insert/delete/modify memory elements

GRNs

- Example: control system applications
  - 1-cell GRN used for control:
    - Sensors: input into working memory
    - Actuators: respond to contents of working memory

GRNs

- Example: obstacle avoidance



GRNs

- Example: artificial embryogeny

GRNs

- Example: artificial embryogeny
  - Seed: single cell
  - Morphogenesis (via embryogeny):
    - Model cell division, growth, maturation
  - Grow complex, multidimensional objects

GRNs

- Properties:
  - Compact “syntactic” representation
  - Strong contact with formal grammars, computation theory
  - Persistent dynamical properties
    - React
    - Repair
    - Regenerate
  - Design:
    - By hand
    - Evolve them

GRNs

- Evolutionary design:
  - Given:
    - symbol set, axioms, application, fitness function
  - Find set of rewrite rules with high fitness
  - Examples:
    - Robot control (Kumar, Grajdeanu, …)
    - Multi-cellular objects (Miller, Federici, Gordon, …)

GRNs

- Evolutionary design issues:
  - Complexity of the rules
    - LHS: complex patterns
  - Representation of rule sets
    - Fixed vs. variable length
    - Rules as individuals
  - Reproductive operators
    - Mutation?
    - Recombination?

Generative Representations

- Interesting open EA question:
  - Which generative representations are more evolution-friendly?
    - L-systems
    - Cellular automata
    - Genetic regulatory networks
    - …

Evolution Friendliness

- Difficult to measure in general
- Best abstract measure (Price):
  - Fitness correlation
    - Between parents and offspring
  - Is diagnostic rather than constructive

Conclusions

- Evolutionary design issues:
  - Scalability of direct representations
  - Applicability to conceptual design
- Hint from nature: use generative representations
  - Powerful, compact
  - Open EA issues

Additional Information

- www.cs.gmu.edu/~kdejong
- www.cs.gmu.edu/~eclab