

CSC 550: Introduction to Artificial Intelligence

Fall 2008

Emergent approach to AI

- evolution, natural selection, and emergence
- genetic algorithms
 - NP-hard problems, data mining
- artificial life
 - cellular automata, Game of Life, artificial creatures

1

Emergent models

the connectionist approach to AI is patterned after the processes underlying brain activity

- artificial neurons are interconnected into networks
- info is sub-symbolic, stored in the strengths of the connections

the emergent approach is patterned after the processes underlying evolution

- genetic algorithms
 - potential solutions to problems form a population
 - better (more fit) solutions evolve through natural selection
- artificial life
 - ecosystems are defined via finite state machines
 - the conditions of biological evolution are simulated

2

Biological & social evolution

Darwin saw "... no limit to the power of slowly and beautifully adapting each form to the most complex relations of life ..."

- through the process of introducing variations into successive generations and selectively eliminating less fit individuals, adaptations of increasing capability and diversity emerge in a population
- evolution and emergence occur in populations of embodied individuals, whose actions affect others and that, in turn, are affected by others
- selective pressures come not only from the outside, but also from the interactions between members of the population

biological evolution: produces species by selecting among changes in the genome

social evolution: produces knowledge/culture by operating on socially transmitted and modified units of information (memes)

3

Evolution and problem-solving

evolution slowly but surely produces populations in which individuals are suited to their environment

- the characteristics/capabilities of individuals are defined by their *chromosomes*
- those individuals that are most fit (have the best characteristics/capabilities for their environment) are more likely to survive and reproduce
- since the chromosomes of the parents are combined in the offspring, combinations of fit characteristics/capabilities are passed on
- with a small probability, mutations can also occur resulting in offspring with new characteristics/capabilities

John Holland (1975) applied these principles to problem-solving →
Genetic Algorithms

- solve a problem by starting with a population of candidate solutions
- using reproduction, mutation, and survival-of-the-fittest, evolve even better solutions

4

Genetic algorithm

for a given problem, must define:

<i>chromosome:</i>	bit string that represents a potential solution
<i>fitness function:</i>	a measure of how good/fit a particular chromosome is
<i>reproduction scheme:</i>	combining two parent chromosomes to yield offspring
<i>mutation rate:</i>	likelihood of a random mutation in the chromosome
<i>replacement scheme:</i>	replacing old (unfit) members with new offspring
<i>termination condition:</i>	when is a solution good enough?

in general, the genetic algorithm:

```
start with an initial (usually random) population of chromosomes
while the termination condition is not met
  evaluate the fitness of each member of the population
  select members of the population that are most fit
  produce the offspring of these members via reproduction & mutation
  replace the least fit member of the population with these offspring
```

5

GA example

A thief has a bag in which to carry away the 'loot' from a robbery. The bag can hold up to 50 pounds. There are 8 items he could steal, each with a monetary value and a weight. What items should he steal to maximize his \$\$ haul?

tiara	\$5000	3 lbs
coin collection	\$2200	5 lbs
HDTV	\$2100	40 lbs
laptop	\$2000	8 lbs
silverware	\$1200	10 lbs
stereo	\$800	25 lbs
PDA	\$600	1 lb
clock	\$300	4 lbs

could try a greedy approach (take next highest value item that fits)

- based on value: tiara + coins + HDTV + PDA = 49 lbs, \$9,900

note that this collection is not optimal

- tiara + coins + laptop + silverware + PDA + clock = 31 lbs, \$11,300

6

GA example (cont.)

tiara	\$5000	3 lbs
coin collection	\$2200	5 lbs
HDTV	\$2100	40 lbs
laptop	\$2000	8 lbs
silverware	\$1200	10 lbs
stereo	\$800	25 lbs
PDA	\$600	1 lb
clock	\$300	4 lbs

chromosome: a string of 8 bits with each bit corresponding to an item

- 1 implies that the corresponding item is included; 0 implies not included
e.g., 11100000 represents (tiara + coins + HDTV)
01101000 represents (coins + HDTV + silverware)

fitness function: favor collections with higher values

- $\text{fit}(\text{chromosome}) = \text{sum of dollar amounts of items, or 0 if weight} > 50$
e.g., $\text{fit}(11100000) = 9300$
 $\text{fit}(01101000) = 0$

7

GA example (cont.)

reproduction scheme: utilize crossover (a common technique in GA's)

- pick a random index, and swap left & right sides from parents
e.g., parents 11100000 and 01101000, pick index 4
1110|0000 and 0110|1000 yield offspring 11101000 and 01100000

mutation rate: generally kept very low, say 0.1%

- when offspring is born, possibly flip each bit with 0.1% likelihood

replacement scheme: pick fittest half, replace least fit half with offspring

- note: rather simplistic
- in practice, most GA's use a roulette wheel selection algorithm

termination condition: if no improvement over previous generation

- note: rather simplistic
- in practice, could stop at a threshold or use more complex statistics

8

GA example (cont.)

tiara	\$5000	3 lbs
coin collection	\$2200	5 lbs
HDTV	\$2100	40 lbs
laptop	\$2000	8 lbs
silverware	\$1200	10 lbs
stereo	\$800	25 lbs
PDA	\$600	1 lb
clock	\$300	4 lbs

Generation 0 (randomly selected):

11100000 (fit = 9300)	01101000 (fit = 0)
11001011 (fit = 9300)	11010000 (fit = 9200)
00010100 (fit = 2800)	01001011 (fit = 4300)
11110111 (fit = 0)	10011000 (fit = 8200)

choose fittest 4, perform crossover with possibility of mutation

111000 00 + 110010 11 →	11100011	11001001
110 10000 + 100 11000 →	11011000	10010000

Generation 1 (replacing least fit from Generation 0):

11100000 (fit = 9300)	11100011 (fit = 0)
11001011 (fit = 9300)	11010000 (fit = 9200)
11001001 (fit = 8700)	11011000 (fit = 10400)
10010000 (fit = 7000)	10011000 (fit = 8200)

9

GA example (cont.)

tiara	\$5000	3 lbs
coin collection	\$2200	5 lbs
HDTV	\$2100	40 lbs
laptop	\$2000	8 lbs
silverware	\$1200	10 lbs
stereo	\$800	25 lbs
PDA	\$600	1 lb
clock	\$300	4 lbs

Generation 1:

11100000 (fit = 9300)	11100011 (fit = 0)
11001011 (fit = 9300)	11010000 (fit = 9200)
11001001 (fit = 8700)	11011000 (fit = 10400)
10010000 (fit = 7000)	10011000 (fit = 8200)

choose fittest 4, perform crossover with possibility of mutation

1101 1000 + 1110 0000 →	11010000	11101000
1100101 1 + 1101000 0 →	11001010	11010001

Generation 2 (replacing least fit from Generation 1):

11100000 (fit = 9300)	11010000 (fit = 9200)
11001011 (fit = 9300)	11010000 (fit = 9200)
11101000 (fit = 0)	11011000 (fit = 10400)
11001010 (fit = 9000)	11010001 (fit = 9500)

10

Genetic algorithms & NP-hard problems

genetic algorithms are good for problems where an analytical solution is not known or is infeasible

e.g., theoretical CS has identified a class of problems known as *NP-hard*
no polynomial time algorithms are known for these problems
(require exhaustively searching all possible solutions → exponential time)

the implicit parallelism of GA's makes searching a huge space feasible

- can think of GA as massively parallel hill-climbing

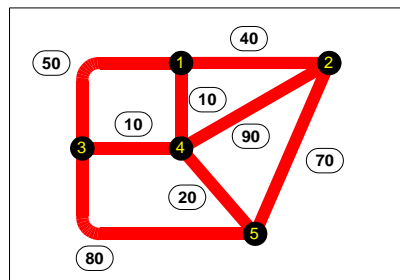
the "sack of loot" problem is an instance of an NP-hard problem known as the *0-1 knapsack problem*

- only known algorithm is to exhaustively test every combination
 $O(2^N)$ where N is the number of items

11

NP-hard: traveling salesman problem (TSP)

A salesman must make a complete tour of a given set of cities (no city visited twice except start/end city) such that the total distance traveled is minimized.



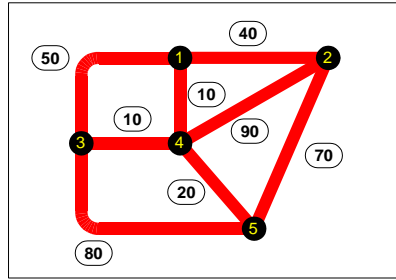
example: find the shortest tour given this map of 5 cities

in general, this problem is NP-hard

- only known algorithm is to exhaustively test every possible path
 $O(N!)$ where N is the number of cities

12

TSP (cont.)



chromosome: a string of 5 digits, corresponding to the order of cities visited

e.g., 12534 12453

fitness function: want to minimize total distance

▪ e.g., $\text{fit}(12534) = 210$ $\text{fit}(12453) = 280$ $\text{fit}(12345) = \infty$

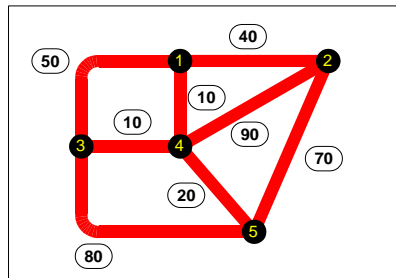
reproduction scheme: crossover?

e.g., parents 12534 and 12453, pick index 3

125 | 34 and 124 | 53 yield offspring 12553 and 12434

13

TSP (cont.)



clearly, straight crossover will not work

- need to ensure that each offspring is a valid permutation

partially matched crossover (PMX):

1. pick two indices 1|253|4 1|245|3
2. swap the portions between those indices in the parents – mark dupes with X 1|245|X 1|253|X
3. fill in X's with missing values 1|245|3 1|253|4

<http://www.ads.tuwien.ac.at/raidl/tspga/TSPGA.html>

14

GA Playground

- [Knapsack Problem with 50 items](#)
- [Traveling Salesman Problem on 48 state capitals](#)
- [Allocating resources to minimize cost](#)

15

Genetic algorithms applications

genetic algorithms for scheduling complex resources

e.g., Smart Airport Operations Center by Ascent Technology

- uses GA for logistics: assign gates, direct baggage, direct service crews, ...
- considers diverse factors such as plane maintenance schedules, crew qualifications, shift changes, locality, security sweeps, ...
- too many variables to juggle using a traditional algorithm (NP-hard)
- GA is able to evolve sub-optimal schedules, improve performance
- Ascent claims 30% increase in productivity (including SFO, Logan, Heathrow, ...)

genetic algorithms for data mining

using GA's, it is possible to build statistical predictors over large, complex sets of data

- e.g., stock market predictions, consumer trends, ...

GA's do not require a deep understanding of correlations, causality, ...

- start with a random population of predictors
- fitness is defined as the rate of correct predictions on validation data
- "evolution" favors those predictors that correctly predict the most examples

e.g., Prediction Company was founded in 1991 by astrophysicists (Farmer & Packard) developed software using GA's to predict the stock market – very successful

16

Artificial life

an interesting field of research under the emergent umbrella is *artificial life*

- alife is the study of lifelike organisms and systems built by humans
- goal is to better understand life as it exists on earth and might exist elsewhere
- success modeling
 - ✓ the growth of plants, shells
 - ✓ the development of cooperation in herds/schools (Prisoner's Dilemma)
 - ✓ the foraging behavior of ants
 - ✓ the flocking behavior of birds

key ideas of alife

- complex natural phenomena can be reproduced in machines
- complex behavior emerges from simple systems interacting in a complex world
- artificial creatures can evolve to suit their environment, similar to natural ones

17

Early alife

John von Neumann is considered the father of alife

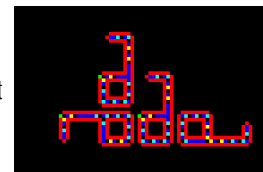
- he felt that all living creatures were simply incredibly complex automata
 - life can be modeled as a finite state machine, information is KEY
- he viewed reproduction as the transfer and reimplementation of information
 - note: his conjectures were before the time of Watson & Crick*

von Neumann was intrigued/obsessed with the idea of self-replication

- he envisioned a robot made of girders, a sensor, a welder, and an arm
 - each robot would be capable of constructing an exact duplicate from raw materials
- eventually, he settled on a more abstract model – cellular automata
 - the world is made up of a grid of cells, each cell a finite automata
- von Neumann described an incredibly complex self-replicating cellular automata
 - 29 different states, as many as 150,000 cells required for one organism

in 1979, Chris Langton devised a simpler model

- required only 8 different states
- a self-reproducing organism could be constructed out of 100 cells
- interesting parallels with the way coral reproduces



18

Conway's Game of Life

Cellular automata provide a simple model for constructing and experimenting with ecosystems

mathematician John Conway began experimenting with rules for 2-D CA's in the early 60s

evolved into the Game of Life – popular with researcher & hobbyists

the Life ecosystem is a rectangular grid of cells

- a cell can be alive (i.e., contain a living individual) or dead
- simple rules model evolution of the ecosystem
 1. a dead cell becomes alive in the next generation if it has exactly 3 neighbors
3 neighbors provide protection, but not too much competition
 2. a living cell survives in the next generation if it has 2 or 3 neighbors
< 2 neighbors leave individual vulnerable, > 3 represent overpopulation

[Game of Life](#)

19

Cellular automata and computation

Game of Life demonstrates that even in a simple world with simple rules, complex behavior can emerge

- Game of Life has been shown to be Turing-complete (can model AND, OR, NOT)

the video game "The Sims" is a natural extension of cellular automata

- each Sim character is a finite state machine
 - 5 personality traits, 6 basic skills, 8 moods (hunger, energy, comfort, fun, ...)
- the algorithm defining the behavior of each Sim is simple:
 1. Pick a mood or skill that the Sim needs to improve.
 2. Go to an object (or another Sim) where that need can be improved.
 3. Animate the Sim performing the action with the object that improves that need.
 4. Increase the mood or skill improved by performing that action.
 5. Return to Step 1.



20

Artificial creatures

in the 1980s, graphical artist Craig Reynolds conducted research into "flocking behaviors"

- he created a simulation of birds flying, based on the following general rules:
 1. If heading towards an obstacle, a bird will steer to avoid it.
 2. If too close to another bird, a bird will steer to avoid contact.
 3. For protection and better feeding/reproductive opportunities, a bird will tend to match the speed and direction of neighboring birds.
 4. For protection, a bird will prefer to fly in the center of the flock.
 5. All other things aside, a bird will fly in the direction of its destination.

flocking model originally developed by Reynolds has been used in a variety of applications, including films (Lion King, Batman Returns, ...)

[Boids simulation](#)

[Fish simulation](#)

[Disney Meets Darwin](#)

21

Tierra

in 1990, biologist Tom Ray created Tierra, an artificial world populated by artificial creatures

- creatures are blocks of assembly language code (32 different instructions)
- each creature contained an "electronic template" could search its environment and find compatible templates for reproduction
- mutations could also occur, substituting instructions in a creature

Ray started with a population of creatures, 80 instructions long (shown in red)

CPU time was divided evenly among all creatures, 12 million instructions executed per second

eventually mutations appeared that were able to reproduce (colors) – smaller mutations began to dominate since could reproduce faster

eventually, 45-instr. parasites developed (shown in yellow) that borrowed reproductive code from their hosts

parasites dominated until immune hosts (shown in blue) evolved, and the process continued

