

## Search Spaces

- Normally, when doing a search, we are looking for a feasible solution to a problem
- What if?
  - We don't know where to start searching.
  - We are interested not only in the answer, but in the search itself.
  - We know what a good answer looks like, but have no known heuristic for finding it.

## Genetic Computing

- We can use evolution as an inspiration in creating our search programs.
- Lecture based off of material found at:  
<http://cs.felk.cvut.cz/~xobitko/ga/>



Picture obtained from:  
[www.genetic-programming.org](http://www.genetic-programming.org)

## Some History

- 1960s: The idea was introduced by I. Rechenberg in his work *Evolution Strategies*
- 1975: John Holland invented Genetic Algorithms (GAs) and published the book *Adaptation in Natural and Artificial Systems*
- 1992: John Koza used genetic algorithms to evolve programs to perform certain tasks and called the method **genetic programming** (GP). He used Lisp for this.

## The Positive About GAs

- Some uses of GAs:
  - NP-hard problems
  - machine learning
  - evolving simple programs
  - evolving art and music
- Easy to implement: just change the chromosome and fitness function.
- Easy to parallelize

## Disadvantages of GAs

- Computational time: GAs may be slower than other methods
- It may be difficult to formulate a problem in terms of a chromosome and fitness function

## Biology Background

- Chromosomes: strings of DNA that serve as a model for the whole organism
- Genes: blocks of DNA that encode a particular protein. These proteins represent traits such as eye color.
- During reproduction:
  - Recombination/crossover: Genes from parents form in some way to create a whole new chromosome
  - Mutation: the elements of the DNA are a bit changed or mutated.
- Fitness: the success of an organism in its life

## What kinds of search?

- See: <http://cs.felk.cvut.cz/~xobitko/ga/>
- Search space example: We want to find the global minimum in the search space
  - Where do we start?
  - Where do find the solution?
  - Hill climbing won't work with this problem

## NP Problems

- It's very hard to find a solution, but once we have a candidate, it's easy to check the solution we have in polynomial time.
- Many AI problems are NP problems:
  - Traveling Salesman (NP-hard)
  - SAT: Boolean Satisfiability (NP-complete)

## How Does this Apply to Search?

1. [Start] Generate random population of n chromosomes (suitable solutions for the problem)
2. [Fitness] Evaluate the fitness  $f(x)$  of each chromosome x in the population
3. [New population] Create a new population by repeating following steps until the new population is complete
4. [Replace] Use new generated population for a further run of algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 2

## How a New Population is Determined

1. [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
2. [Crossover] With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
3. [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).
4. [Accepting] Place new offspring in a new population

## Encoding a Chromosome

- Some chromosomes are binary strings:

Chromosome 1	1101100100110110
Chromosome 2	1101111000011110

## Encodings

- Binary: string of 1's and 0's
- Permutation: every chromosome is a string of numbers, which represents a number in a sequence
- Direct value encoding: done for complicated problems like real value problems
- Tree encoding: useful for encoding programs

## CNF SAT Problem

$$(\neg a \vee c) \wedge (\neg a \vee c \vee \neg e) \wedge (\neg b \vee c \vee d \vee \neg e) \wedge (a \vee \neg b \vee c) \wedge (\neg e \vee f)$$

- Is there some choice of truth for the given literals that will make the whole clause true?
- How would we encode this?

## Encoding the Traveling Salesman Problem

- Say we've got 9 cities we want to tour
- How would we encode this?

## Crossover

- Selects genes from the parent chromosomes and creates a new offspring.

Chromosome 1	11011   00100110110
Chromosome 2	11011   11000011110
Offspring 1	11011   11000011110
Offspring 2	11011   00100110110

## CNF SAT Problem

- How do we do crossover with the SAT problem?

## The Traveling Salesman Problem?

- Crossover?

## Mutation

- To prevent all population members from falling into a local minimum, we may mutate a part of the offspring's chromosome

Offspring 1	1101111000011110
Offspring 2	1101100100110110
Mutated Offspring 1	1100111000011110
Mutated Offspring 2	1101101100110100

## See on-line example

---

- GA Example  
(1D function): <http://cs.felk.cvut.cz/~xobitko/ga/>

## Mutation: CNF SAT

---

- How do we do mutation with the SAT problem?

## Mutation: Traveling Salesman

---

- How do we do mutation on the traveling salesman problem?

## Variations on mutation and crossover

---

- See Mutation and Crossover Section: <http://cs.felk.cvut.cz/~xobitko/ga/>

## Probabilities

---

- Crossover and mutation both have some probability of happening
  - You can set such probabilities
- Population size has trade-offs
- See the Parameters section at: <http://cs.felk.cvut.cz/~xobitko/ga/>

## Selection

---

- Roulette wheel: more fit chromosomes have more chances to win
- Ranking: Chromosomes are ranked with the highest ranking having the most chances of winning
- Steady state: Most chromosomes survive into the next generation
- Elitism: We copy the best chromosomes over before reproduction

## Games?

- GA's are used in several different ways:
  - Evolving behavior that is more realistic than simple rules can give
    - Example: Space ship landing [from the book: AI Techniques for Game Programming by Buckland and LaMothe]
  - Optimizing a bunch of creature parameters
    - Example: a set of monsters with similar life spans [AI Wisdom 1 book – "Evolving the Perfect Troll" by Francois Dominic Laramee]

## More uses of GA's

- On-line evolution of creatures in a simulation
  - Example: Creatures [a commercial game]
- Creating maps
  - Example: DungeonMaker [http://alifegames.sourceforge.net/dungeonmaker/index.html]
- Off-line evolution of monsters to play better against human opponents
  - Example: bSerene [http://alifegames.sourceforge.net/bSerene/index.html]

## Suggestions

- See Recommendations Section:  
<http://cs.felk.cvut.cz/~xobitko/ga/>