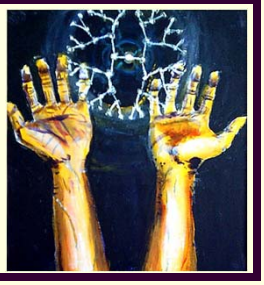
CS 351
Design of Large Programs
Evolving Virtual Creatures

Instructor: **Joel Castellanos**
e-mail: joel@unm.edu

Web: http://cs.unm.edu/~joel/
Office: Farris Engineering Center 319



10/13/2015

Image from Geneffects

Search Algorithms

- Name some. How do they work?
- Computer Scientists have been working on this for a long time. Is this a problem that is solved or an active field of research?
- If it is active, what type of results do people working on it publish?
- If something is an "active" field, then there are many funding sources. Who would fund this?

2

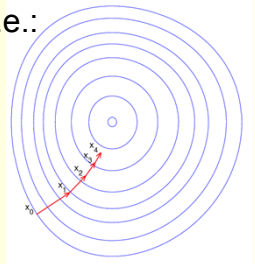
Search Algorithm

- In computer science, a search algorithm finds an item with specified properties among a collection.
- When "*Specified Properties*" includes "*best*" the search must include *all items in the collection*.
- Items may be:
  - Stored individually as records in a database.
  - Elements of a search space defined by a mathematical formula or procedure.
- Algorithms for these problems include:
  - Brute-force Search.
  - Heuristics that try to exploit partial knowledge about structure of the space.

3

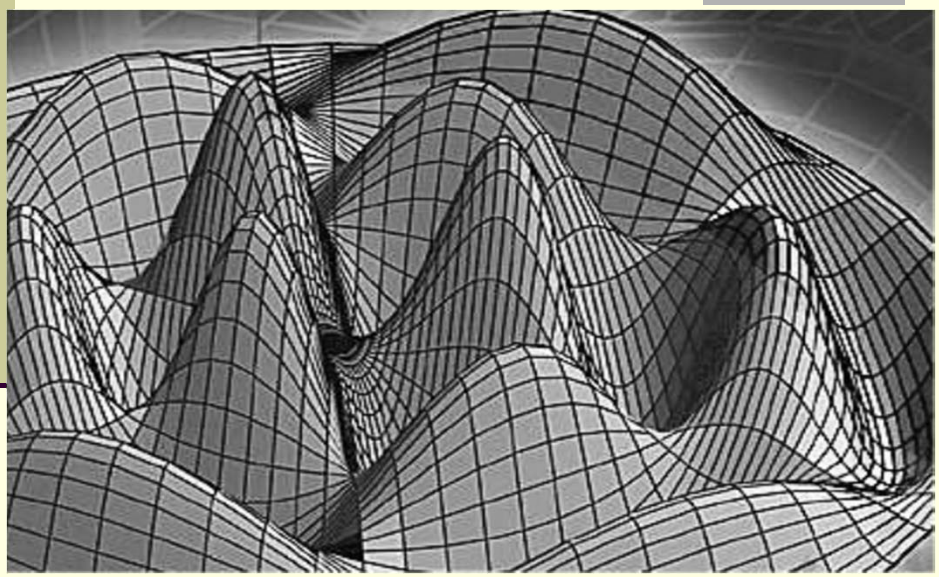
Local Search Methods

- ***Local search methods*** attempt to find a global solution by examining a particular item and some ***neighborhood*** around that item.
- The elements of the search space are viewed as the vertices of a graph, with edges defined by a set of heuristics applicable to the case.
- The space is scanned by moving from item to item along the edges, according to some heuristic. i.e.:
  - Steepest Descent.
  - Best-First Criterion.
  - Stochastic.



4

Search Space with Local Maxima



5

Hill Climbing

- In computer science, ***hill climbing*** is a mathematical optimization technique which belongs to the ***local search*** family.
- It is an iterative algorithm.
- It starts with an arbitrary solution to a problem.
- It then attempts to find a better solution by incrementally changing a single element of the solution.
- If the change produces a better solution, then that ***same*** change is made again. Otherwise a different change is made.
- This is repeated until no further improvements can be found.

6

Hill Climbing Mutation Types

- In pure hill climbing, a mutation means to ***incrementally change a single element of the solution***.
- Heuristics that try to exploit partial knowledge about structure of the space may define different ***types*** of mutation.

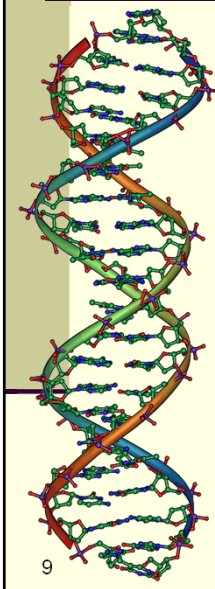
7

Adaptive Hill Climbing

- Start with an arbitrary mutation step size, S.
- Let each mutation be a random step, s_i, from 1 to S in a random direction, $\pm$ (or toward/away form center).
- Select a random *type* of mutation to a random *item*.
- When a mutation yields an improvement:
  1) Increase the probability of selecting that mutation (same type, same item, same direction).
  2) Adapt the step size:
     - If $s_i < S/2$, then decrease S.
     - If $s_i \geq S/2$, then increase S.
- Adaptive Rate: As slow as needed to observe trends.

8

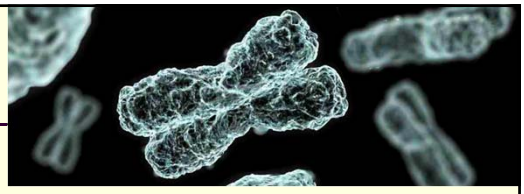
Genetic Algorithm



- A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems.
- Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology:
  - Inheritance
  - Mutation
  - Selection
  - Crossover

9

GA Chromosomes



- In genetic algorithms, a chromosome is a set of parameters which define a proposed solution to the problem that the genetic algorithm is trying to solve.
- The chromosome is often represented as an array of characters, although a wide variety of other data structures are also used.
- *Each parameter* that makes up the chromosome is called a *gene*.
- The *full set* of genes in a solution is called the solution's *chromosome*, *DNA* or the *genome*.

10

GA Genome Design Example 1

- Suppose the problem is to find the integer value of x between 0 and 255 that provides the maximal result for $f(x) = 536x - x^2$.
- Possible solutions are the integers from 0 to 255, which can all be represented as 8-digit binary strings.
- Thus, we might use an 8-digit binary string as our genome. If a given genome in the population represents the value 155, its genome would be 10011011.
- For this problem, the binary representation is a better genome than the base 10 string which is too short and had different constraints on different digits.

11

GA Genome Design Example 2

- A more realistic problem we might wish to solve is the travelling salesman problem.
- In this problem, we seek an ordered list of cities that results in the shortest trip for the salesman to travel.
- Suppose there are six cities, which we'll call A, B, C, D, E, and F.
- A good design for our genome might be the ordered list we want to try.
- An example genome we might encounter in the population might be DFABEC.

12

Fitness Function

- A *fitness function* is a particular type of objective function that quantifies the optimality of a solution (a genome) in a genetic algorithm so that that particular genome may be ranked against all the other genome.
- Generally, genomes which are more optimal, are more often allowed to reproduce and mix their datasets by any of several techniques.
- An ideal fitness function correlates closely with the algorithm's goal, and yet may be computed quickly. Speed of execution is very important, as a typical genetic algorithm must be iterated many, many times in order to produce a usable result for a non-trivial problem.

13

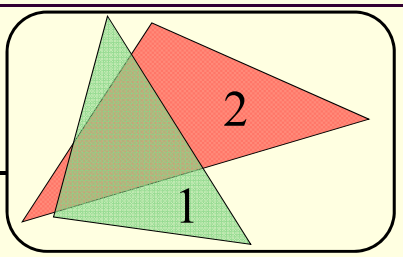
Where Did That Come From?

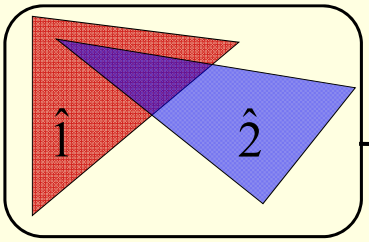
Natural selection does not grant organisms what they "need".

14

Uniform Crossover

Each genome in this example has just 2 triangles.





$$[(x_1^1 y_1^1 x_2^1 y_2^1 x_3^1 y_3^1 r^1 g^1 b^1 t^1),(x_1^2 y_1^2 x_2^2 y_2^2 x_3^2 y_3^2 r^2 g^2 b^2 t^2)]$$

$$[(\hat{x}_1^1 \hat{y}_1^1 \hat{x}_2^1 \hat{y}_2^1 \hat{x}_3^1 \hat{y}_3^1 \hat{r}^1 \hat{g}^1 \hat{b}^1 \hat{t}^1),(\hat{x}_1^2 \hat{y}_1^2 \hat{x}_2^2 \hat{y}_2^2 \hat{x}_3^2 \hat{y}_3^2 \hat{r}^2 \hat{g}^2 \hat{b}^2 \hat{t}^2)]$$

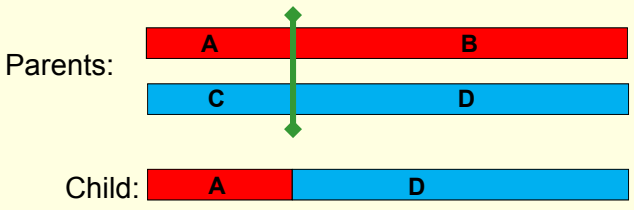
$$[(\hat{x}_1^1 \hat{y}_1^1 \hat{x}_2^1 \hat{y}_2^1 x_3^1 \hat{y}_3^1 \hat{r}^1 g^1 b^1 t^1),(x_1^2 y_1^2 x_2^2 \hat{y}_2^2 x_3^2 \hat{y}_3^2 \hat{r}^2 g^2 \hat{b}^2 t^2)]$$

Triangle 1 Triangle 2

15

Single Point Crossover

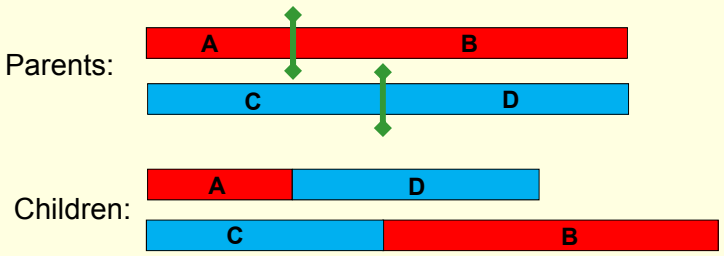
- Starting from one end of one parent genome, copy gene by gene until some random crossover point is reached.
- At that crossover point, start copying from the second parent and continue until the end of the genome.



16

Cut and Splice

- Pick a random location in the DNA of each of the parents.
- Split the parent DNA at these locations.
- Produce two children by splicing the first part of one with the second part of the other.



17

GA Selection

Selection is the stage of a genetic algorithm in which individual genomes (full solution sets) are chosen from a population for breeding (recombination or crossover).

There are many different methods for selection a pair of parents from the population such as:

- Tournament Selection and
- Fitness Proportionate Selection

All selection methods must have the following properties:

- Every genome in the population must have some chance of being chosen.
- Genomes with higher fitness must be chosen more often than genome with less fitness.

18

Unit of Selection: The Gene

- A unit of selection is a biological entity within the hierarchy of biological organisation (e.g. genes, cells, individuals, groups, species) that is subject to natural selection.
- For several decades there has been intense debate among evolutionary biologists about the extent to which evolution has been shaped by selective pressures acting at these different levels.
- For example, is it group or individual selection that has driven the evolution of altruism?
- Most genetic algorithms use the gene as the only unit of natural selection.

19

Fitness Proportionate Selection: Gaussian

Commonly, the population in a GA is in the 100s or 1000s.

Consider a GA with a population of 100 that has been sorted from most fit in element 0 to least fit in 99.

A Very Simple Gaussian Approximation:

Goal: a half bell curve: 0 is the most probable number, 1 a bit less...99 is possible, but not very common.

```
java.util.Random rand = new java.util.Random()
r1 = rand.nextInt(100);
r2 = rand.nextInt(100);
```

20

Fitness Proportionate Selection: Triangle Number

- For a simple example, let the population consist of 6 genomes.
- Let these genomes be sorted from most fit (genome number 0 through least fit (genome number 5).
- Then one way to make a fitness proportionate selection would be:
  1. Choose a random number, r, between 1 and the *triangular number* (T_N) of the population, N: $T_N=N\,(N+1)/2$. Here, $N=6$. Thus, $r = [1, 21]$.
  2. The genome number, g, is found by solving $r = g(g+1)/2$ for g.

21					
20	15				
19	14	10			
18	13	9	6		
17	12	8	5	3	
16	11	7	4	2	1
0	1	2	3	4	5

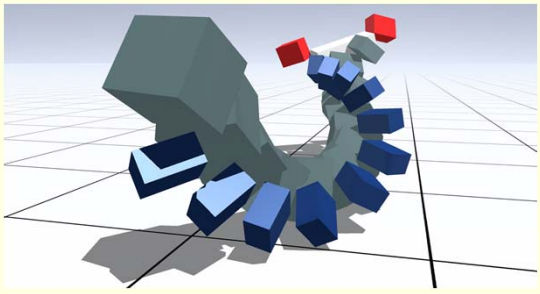
Genome Number
Sorted by Fitness.

$$g = N - \text{int}\left(\frac{1+\sqrt{1+8(r-1)}}{2}\right)$$

21

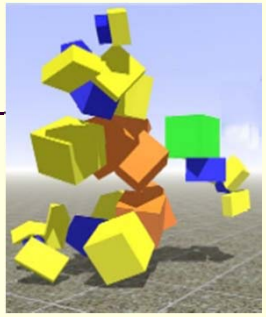
Project 2: Evolving Virtual Creatures

- Based on *Evolving Virtual Creatures* by Karl Sims SigGraph 94 (paper on class website).
- Find (and render) the set of rectangular solids, hinges and instructions that encode the highest jumping creature.
- Fitness: during a creature's simulation, the maximum height of the lowest point on the creature.
- How large is this Search Space?



22

Body Structure



- Creatures are composed of rectangular solids (blocks) and Joints (not rendered).
- Joints may only be placed at contact points of two body parts (rectangular solids).
- No block may have a dimension < 1 meter.
- The maximum dimension of any block may be no more than 10x that same block's minimum dimension.
- Each creature must begin life in contact with the ground.
- Each creature must begin life with no body parts intersecting each other or the ground.
- At no time during the creature's life may any of its parts become disconnected.

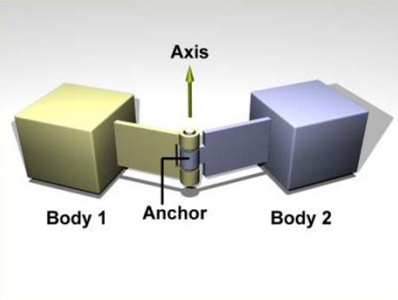
23

Physics, Units and Constants

- All units are MKS (Meter, Kilogram, Second) based with angles measured in radians.
- Given in `vcreature.phenotype.PhysicsConstants`
- Bullet Physics running at 80 frames per second.
- Creature density: uniform, 4 kg/m^3
- Acceleration due to gravity: -9.81 m/s^2.
- Surface Sliding Friction (blocks and ground): 1.0 bullet units.
- Bounciness: floor=0.3, blocks=0.5 bullet units.
- Damping: linear=0.2, angular=0.1 bullet units.

24

Joints



- All Joints must be Bullet Physics one degree-of-freedom `HingeJoint`.
- Range: $\pm\frac{\pi}{2}$ radians.
- Each joint connects a parent block to a child block.
- Maximum Impulse (Newton•sec) is proportional to the surface area of the parent block.
- Since all blocks have the same density, the impulse needed to move the child block is effectively proportional to the volume of the child (and all its descendant blocks.)
- No joints may apply an Impulse until 1 second after starting.

25

Sensors & Other Data

- The neuron rule table can access sensor data from each box. Each box has:
  - A touch sensor (boolean)
  - A height (center of gravity) sensor (float)
  - An angle sensor for each degree of freedom of each joint(float).
- Additionally, each creature specifies the up and forward vector of the root box and the physics engine sets these vectors for all other boxes. This is intended for use by the 3D rendering module.

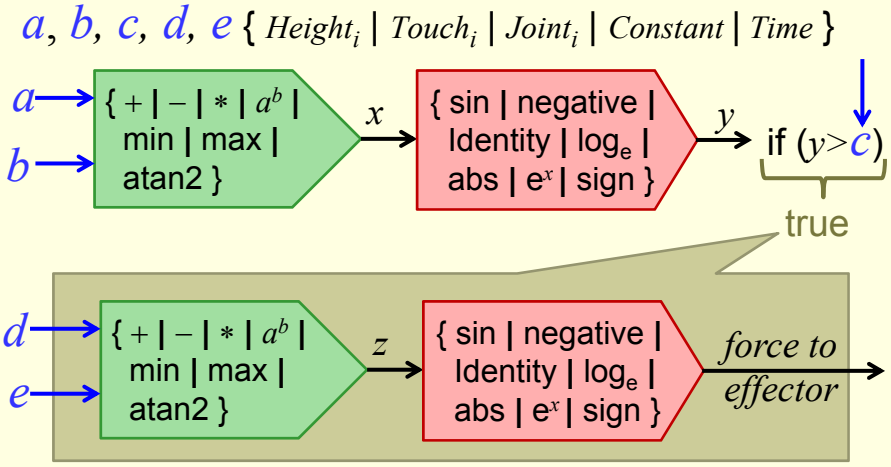
26

The Creature's Brain: Effector Rule Table

1. Each effector (degree of freedom of each joint) has a sequence of 0 to n rules called the effector's rule table.
2. The first rule in a rule table that is true activates its neuron and no more rules from that table are evaluated.
3. In a given time step, if none of an effector's rules are true, then the effector is sent a default signal of 0 dynes.

27

Neuron Anatomy of a Joint's Rule Table



Functions values that are undefined, (i.e. log(-2)), return 0

28

Grading Rubric: total 800 Pts (1 of 3)

Note: Having not yet done this project myself, I will not list numeric fitness requirements. What is meant by "good" and "efficient" will be determined over the next few weeks as I write my own implementation.

50	Arrival at a fit, unique (should be very different from what any other group finds), reloadable creature that will run the same in your implementation of the simulator as in the instructor's. Must NOT use serialization to save/restore genome.
75	Hill climbing: GUI setting for hill climbing only uses adaptive hill climbing to efficiently approach the top of an n-dimensional hill in fitness land. The adaptive heuristic used is your choice. This choice must be clearly documented. Include in the documentation not only what you did, but a **quantitative** analysis of how effective it was as well as what you would likely change if starting over.
75	Multi Core Hill climbing: Each thread of GUI selectable 1 through 32 worker threads should each independently climb a different hill. GUI must allow viewing of user selectable thread's current creature while other threads continue to work.

29

Grading Rubric (2 of 3)

35	GUI display showing for each thread, current fitness, fitness change per minute and total fitness change from start.
50	GA: GUI setting for Genetic Algorithm to auto start for a particular thread when that thread's hill climbing fitness change per minute slows to below a user specified threshold. The GA must select from a population that includes all threads. Only genomes currently under modification may be excluded form the pool.
75	GA effectiveness: Hill climbing + GA must reach significantly higher fitness after 2 hours than hill climbing alone.
50	Analysis 1: Must produce table and graph showing Effectiveness of Hill climbing alone verses with GA over multiple 12 run hour periods.
50	Analysis 2: Must produce table and graph showing comparative effectiveness of using 1, 2, 4, 8, 16 and 32 threads on a 4 or 8 core (your choice) machine. Note: Since most time should be spent in the fitness calculation, if bullet physics is able to parallelize many of your calculations, it might be that on a 4 core machine your performance is actually better when you have only 1 or 2 worker threads. If so, that is part of your analysis.
50	Analysis 3: Must produce table and graph showing population diversity (in total number of gene differences) verses time for a 12 hour period.

30

Grading Rubric (3 of 3)

50	Analysis 4: Quantitatively compare the effectiveness of different cross-over heuristics and different selection heuristics. Which heuristics you uses is your choice, but must have more than one of each.
50	GUI must remain responsive at all times.
50	Program must be able to run from command line on a PuTTY connection without a GUI and without a 3D display, but with fitness progress reports sent to console every minute and best creature auto saved every hour.
70	Organized class presentation of working best creature, most "special" creature and analysis.
70	Project design, version control, code documentation and code style.
+120	Extra Credit: First place team (most fit Creature in class).
+60	Extra Credit: Second place team.
+40	Extra Credit: Third place team.

31