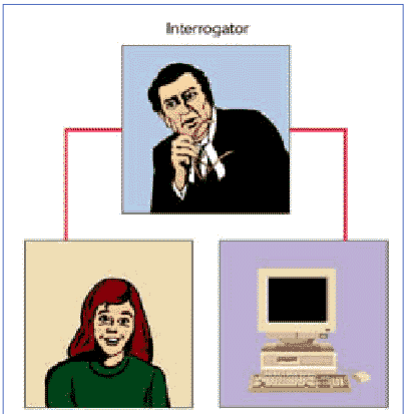
Introduction to Artificial Intelligence

What is Artificial Intelligence?

- One definition: AI is the study of how to make computers do things that people generally do better
- Many approaches and issues, e.g.:
  - Philosophy
  - Cognitive Science
  - Logic and Rules
    - Search, Game-Playing
  - Neural Networks
  - Evolution

Philosophical Discussions

- Test for intelligence?
  - Turing Test



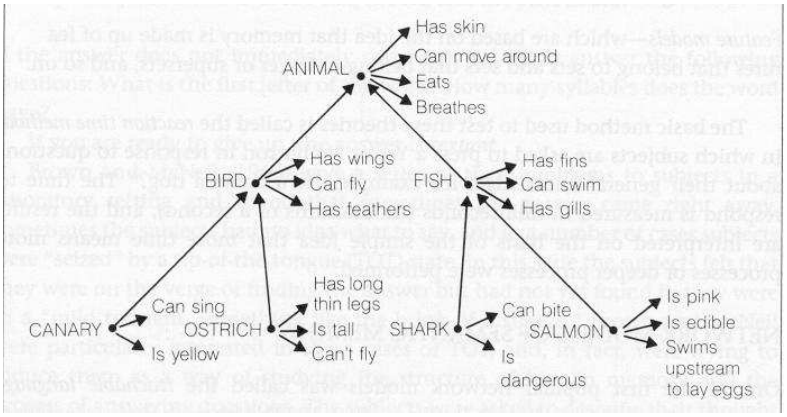
Philosophical Discussions

- Physical Symbol Hypothesis
  - Newell & Simon, 1976
  - The thinking mind consists of the manipulation of symbols. That is, a physical symbol system has the necessary and sufficient means for general intelligent action.
- If true, then a computer has the necessary means to implement general intelligent action
  - What rights should an intelligent computer have?
- Counter-arguments
  - Lack of consciousness
  - Lack of self-awareness

Cognitive Science

- Approach AI from the human perspective
  - Psychology and Cognitive Science
- Example: Sentence Verification Experiment

Semantic Network/Memory Model



Search and Problem Spaces

- Searching a "Problem Space" or "State Space" for a solution is a common theme in AI
  - relies largely on the computer's ability to search, by brute force, a huge number of possible states
- Example: Water Jug problem

Water Jug State Space Search

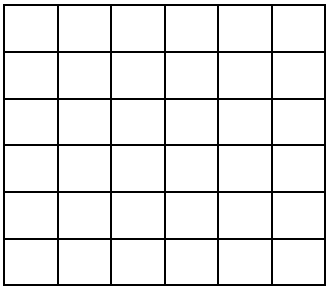
Classical Game Playing

- Consider a 2 player game like chess
  - Can't use the previous search technique, too many states
  - On average, about 35 moves can be made
  - If each player makes 50 moves, the number of states to search is 35^{100} which is untractable

Minimax

- Solution: Generate a search tree as far ahead as is feasible, compute a heuristic function for each state, and make the move leading to the best state
- Heuristic function: Computes a number that guesses how close the state is to winning

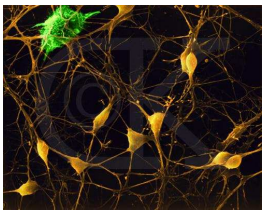
Minimax/Heuristic Example: Othello

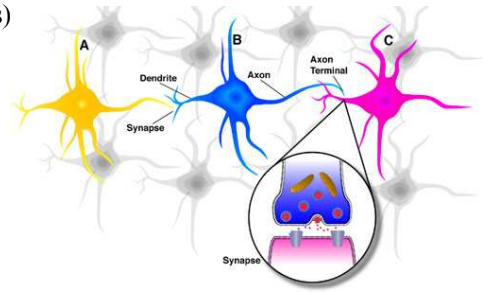


Heuristic: My pieces – His pieces

Neurons in the Brain

- Although heterogeneous, at a low level the brain is composed of neurons
  - A neuron receives input from other neurons (generally thousands) from its synapses
  - Inputs are approximately summed
  - When the input exceeds a threshold the neuron sends an electrical spike that travels that travels from the body, down the axon, to the next neuron(s)



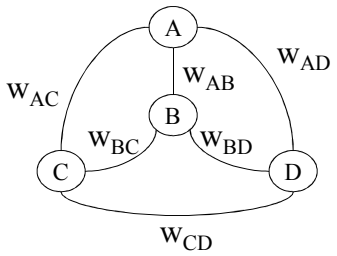


Example: Hopfield Networks

- Many different types of computer-based neural networks
- Machine learning
  - Given examples, learn the category
- One is a Hopfield network which is a type of content-addressable memory
  - Network stores attractor points that represent concepts
  - Given a fuzzy input the system converges to the nearest attractor

Standard Binary Hopfield Network

- Recurrent; Every unit is connected to every other unit
- Weights connecting units are symmetrical
  - $w_{ij} = w_{ji}$
- If the weighted sum of the inputs exceeds a threshold, its output is 1 otherwise its output is -1
- Units update themselves asynchronously as their inputs change



Hopfield Memories

- Setting the weights:
  - A pattern is a setting of on or off for each unit
  - Weights are adjusted so they strengthen connections to other units that are turned on at the same time, weakened if they are turned off at the same time
- Demo
  - http://www.cbu.edu/~pong/ai/hopfield/hopfieldapplet.html

Evolution in Computers

- Genetic Algorithms – most widely known work by John Holland
- Form of machine learning
- Based on Darwinian Evolution
  - In a competitive environment, strongest, "most fit" of a species survive, weak die
  - Survivors pass their good genes on to offspring
  - Occasional mutation

Evolution in Computers

- Same idea in computers
  - Population of computer program / solution treated like the critters above, typically encoded as a bit string
  - Survival Instinct – have computer programs compete with one another in some environment, evolve with mutation and sexual recombination

GA's for Computer Problems

Population of critters → Population of computer solutions
Surviving in environment → Solving computer problem
Fitness measure in nature → Fitness measure solving computer problem
Fit individuals life, poor die → Play God and kill computer solutions that do poorly, keep those that do well. i.e. "breed" the best solutions typically Fitness Proportionate Reduction
Pass genes along via mating → Pass genes along through computer mating
Repeat process, getting more and more fit individuals in each generation.
Usually represent computer solutions as bit strings.

The Simple Genetic Algorithm

1. Generate an initial random population of M individuals (i.e. programs or solutions)
2. Repeat for N generations
    1. Calculate a numeric fitness for each individual
    2. Repeat until there are M individuals in the new population
        1. Choose two parents from the current population probabilistically based on fitness (i.e. those with a higher fitness are more likely to be selected)
        2. Cross them over at random points, i.e. generate children based on parents (note external copy routine)
        3. Mutate with some small probability
        4. Put offspring into the new population

Crossover

Typically use bit strings, but could use other structures

Bit Strings: Genotype representing some phenotype

Individual 1: 00101**0001** Individual 2: **10011**0110

New child : **10011****0001** has characteristics of both parents, hopefully better than before

Bit string can represent whatever we want for our particular problem; solution to a complex equation, logic problem, classification of some data, aesthetic art, music, etc.

Example : Traveling Salesman Problem

- 29 Node Traveling Salesperson Problem
- 29! = 8.8 trillion billion billion possible asymmetric routes.
- ASCI White, an IBM supercomputer being used by Lawrence Livermore National Labs to model nuclear explosions, is capable of 12 trillion operations per second (TeraFLOPS) peak throughput
- Assuming symmetric routes, ASCI White would take 11.7 billion years to exhaustively search the solution space

- A genetic algorithm approach
  - Randomly generate a population of solutions
    - Each solution represents an entire solution, i.e. a random ordering of nodes representing a loop
      - Given nodes 1-6, we might generate 423651 to represent the loop of visiting 4 first, then 2, then 3, then 6, then 5, then 1, then back to 4
  - Assign each solution a fitness value
    - Fitness is just the sum of the distance for edges in the loop; lower is more fit
  - Evolve a new, hopefully better, generation of the same number of agents
    - Select two parents randomly, but higher probability of selection if better fitness
    - New generation formed by crossover and mutation

- Crossover
  - Must combine parents in a way that preserves valid loops
  - Typical cross method, but invalid for this problem

    Parent 1 = 423651 Parent 2 = 156234

    Child 1 = 423234 Child 2 = 156651
  - Use a form of order-preserving crossover:

    Parent 1 = 423651 Parent 2 = 156234

    Child 1 = 123654
    - Copy positions over directly from one parent, fill in from left to right from other parent if not already in the child
- Mutation
  - Randomly swap nodes (may or may not be neighbors)

- Traveling Salesman Applet:

  Generates solutions using a genetic algorithm

  http://www.generation5.org/jdk/demos/tspApplet.html

- Smart Rockets
  - http://www.blprnt.com/smartrockets/