Introduction to Artificial Intelligence

Unit # 13

Course Outline

- Overview of Artificial Intelligence ✔
- State Space Representation ✔
- Search Techniques ✔
- AI in Games ✔
- Machine Learning ✔
- Propositional and Predicate Logic ✔
- Probabilistic Reasoning ✔
- ~~Introduction to Robotics~~ (To be taught separately)
- **Evolutionary Algorithms**
- Natural Language Processing
- Reinforcement Learning

Remaining Lectures

- Evolutionary Algorithms (2 Lectures)
- AI in Video Games (1 Lecture)
- Presentation – HW 4 (1 Lecture)
- Hidden Markov Model/NLP (1 Lecture)
- Reinforcement Learning (1 Lecture)
- Revision (1 Lecture)
- Projects Presentations (2 Lectures)

- Robotics (Separate Sessions)

Evolutionary Computing

- Evolutionary Computing is a subclass of Artificial Intelligence that involves combinatorial optimization problems.
- Evolutionary computation uses iterative progress, such as growth or development in a population. This population is then selected in a guided random search using parallel processing to achieve the desired end. Such processes are often inspired by biological mechanisms of evolution.

Evolutionary Computing

- Evolutionary Algorithms
  - Genetic Algorithms
  - Evolutionary Programming
  - Evolutionary Strategies
  - Genetic Programming
- Swarm Intelligence
  - Particle Swarm Optimization
  - Ant Colony Optimization
- Other Nature Inspired Models
  - Artificial Immune Systems
  - Differential Evolution
  - Honey Bee Optimization

Overview of Evolutionary Algorithms

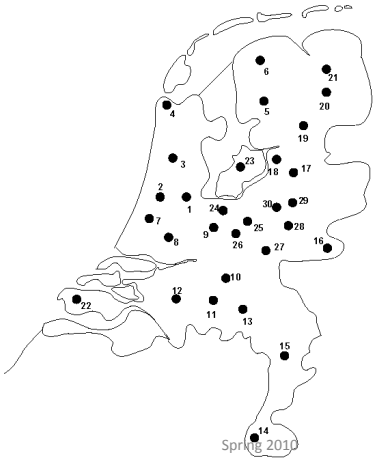
- A parallel search scheme that is inspired by biological evolution
- EAs have been used successfully for optimization problems in several fields
- They make little assumptions about the landscape
- Works for problem where classical schemes fail such as
  - When derivate of a function does not exist
  - The function is discontinuous

History

- Several efforts were started in parallel during 1960s
  - Evolutionary Strategies (Berlin Technical University)
  - Genetic Algorithms (University of Michigan)
  - Evolutionary Programming (UCLA)
- During 1990s the above communities agreed to the term "Evolutionary Algorithms".

Traveling Sales Person Problem

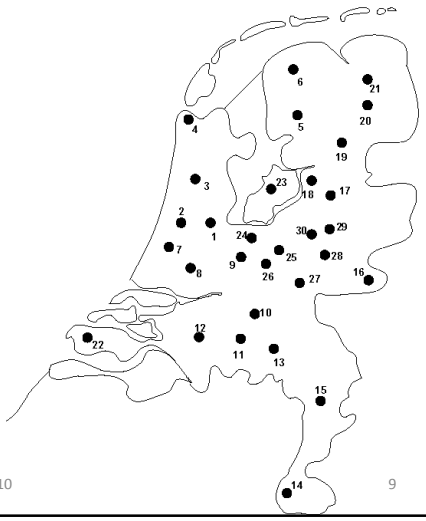
- Given a number of cities and the costs of traveling from one city to any other city, what is the cheapest round-trip route that visits each city exactly once and then returns to the starting city?



Permutation Representation: TSP

- Problem:
  - Given n cities
  - Find a complete tour with minimal length
- Encoding:
  - Label the cities 1, 2, … , *n*
  - One complete tour is one permutation (e.g. for n =4 [1,2,3,4], [3,4,2,1] are OK)
- Search space is BIG:

  for 30 cities there are 30! ≈ 10^{32} possible tours



TSP: Nearest Neighbor

	A	B	C	D	E	F	G	H
A	0	8	3	1	4	9	3	6
B	8	0	5	10	11	4	3	6
C	3	5	0	8	7	1	5	12
D	1	10	8	0	9	11	6	4
E	4	11	7	9	0	5	17	3
F	9	4	1	11	5	0	4	1
G	3	3	5	6	17	4	0	7
H	6	6	12	4	3	1	7	0

- Start with A: A – D – H – F – C – B – G – E Cost?
- Start with E: E – H – F – C – A – D – B – G Cost?
- Start with G: G – B – F – H – E – A – D – C Cost?

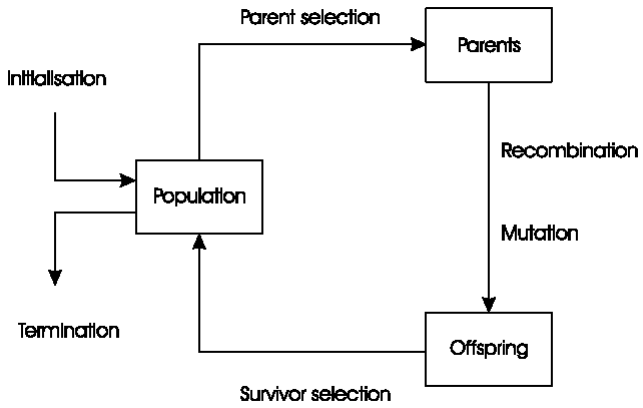
Evolutionary Algorithms

- Evolutionary Algorithms fall into the category of "generate and test" algorithms
- They are stochastic, population-based algorithms
- Variation operators (recombination and mutation) create the necessary diversity and thereby facilitate novelty
- Selection reduces diversity and acts as a force pushing quality

A Typical Evolutionary Algorithm Cycle

- Step 1: Initialize the population randomly or with potentially good ***solutions***.
- Step 2: Compute the ***fitness*** of each individual in the population.
- Step 3: Select parents using a ***selection procedure***.
- Step 4: Create offspring by ***crossover*** and ***mutation*** operators.
- Step 5: Compute the ***fitness*** of the new offspring.
- Step 6: Select members of population to die using a ***selection procedure***.
- Step 7: Go to Step 2 until termination criteria are met.

General Scheme of EAs



Initialize The Population

- Suppose the population size is 6

Candidate Solutions
A C B F H D E G
H B G E A C D F
A H G C B D F E
E G B C D H F A
F H A D C B E G
C D B A H E G F

Compute Fitness

Candidate Solutions	Fitness
A C B F H D E G	43
H B G E A C D F	52
A H G C B D F E	49
E G B C D H F A	47
F H A D C B E G	49
C D B A H E G F	56

	A	B	C	D	E	F	G	H
A	0	8	3	1	4	9	3	6
B	8	0	5	10	11	4	3	6
C	3	5	0	8	7	1	5	12
D	1	10	8	0	9	11	6	4
E	4	11	7	9	0	5	17	3
F	9	4	1	11	5	0	4	1
G	3	3	5	6	17	4	0	7
H	6	6	12	4	3	1	7	0

Selection Procedure

- Selection in evolutionary algorithms is the process of choosing which individuals reproduce offspring and which individuals survive to the next generation.
- When selection is used to choose which individuals reproduce, the process is referred to as ***pre-selection*** (parent(s) selection).
- When it is used to select the individuals that survive to the next generation it is called ***post-selection*** (survival selection).

Selection Schemes

- ***Fitness Proportional***: Individuals are selected based on their fitness in proportion to the other individuals in the population.
- ***Binary Tournament***: Two individuals are randomly selected from the population and compared. The one with the highest fitness is selected for reproduction. Then another two individuals are randomly selected and the best fit is kept as the mate to the first parent.
- There are many other schemes as well.

Parent Selection

- Suppose we use binary tournament
  - 1 vs. 4 (1 is the winner as its fitness is better)
  - 2 vs. 5 (5 is the winner as its fitness is better)

Parent 1	A C B F H D E G	43
Parent 2	F H A D C B E G	49

Crossover

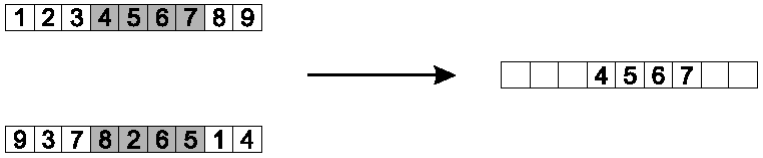
- Merges information from parents into offspring
- Choice of what information to merge is stochastic
- Most offspring may be worse, or the same as the parents
- Hope is that some are better by combining elements of genotypes that lead to good traits
- Principle has been used for millennia by breeders of plants and livestock

Order 1 Crossover

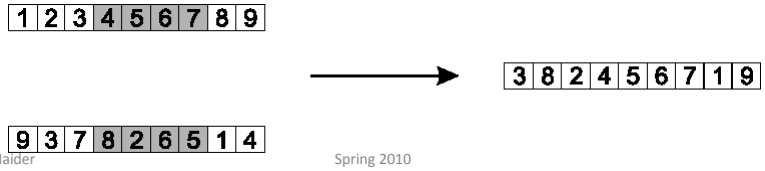
- Idea is to preserve relative order that elements occur
- Informal procedure:
  1. Choose an arbitrary part from the first parent
  2. Copy this part to the first child
  3. Copy the numbers that are not in the first part, to the first child:
     - starting right from cut point of the copied part,
     - using the **order** of the second parent
     - and wrapping around at the end
  4. Analogous for the second child, with parent roles reversed

Order 1 Crossover Example

- Copy randomly selected set from first parent



- Copy rest from second parent in order 1,9,3,8,2



Crossover Example

- We selected the parents in the previous slides

Parent 1	A C B F H D E G	43
Parent 2	F H A D C B E G	49

- Let's do crossover to produce two offspring. Suppose the crossover points are 3 and 5

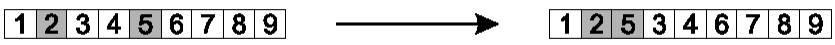
Offspring 1	D C B F H E G A
Offspring 2	F H A D C E G B

Mutation

- After creation of new individuals via crossover, mutation is applied usually with a low probability to introduce random changes into the population
  - Replace gene values lost from the population or not initially present
  - Evaluate more regions of the search space
  - Avoid premature convergence
  - Makes the entire search space reachable
- If applied at high probability, the offspring will lose their resemblance to their parents and the ability of the algorithm to learn from the parents will be lost

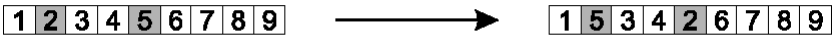
Insert Mutation for Permutations

- Pick two allele values at random
- Move the second to follow the first, shifting the rest along to accommodate
- Note that this preserves most of the order and the adjacency information



Swap Mutation for Permutations

- Pick two alleles at random and swap their positions
- Preserves most of adjacency information (4 links broken), disrupts order more



Mutation Example

- Perform swap mutation on the two offspring produced in the previous slide.

Offspring 1	D C B F H E G A
Offspring 2	F H A D C E G B

- For Offspring 1, swap 2nd and 4th gene.
- For Offspring 2, swap 2nd and 6th gene.

Offspring 1	D F B C H E G A	55
Offspring 2	F E A D C H G B	40

Crossover OR Mutation?

- Exploration: Discovering promising areas in the search space, i.e. gaining information on the problem
- Exploitation: Optimising within a promising area, i.e. using information
- There is co-operation AND competition between them
- Crossover is explorative, it makes a big jump to an area somewhere "in between" two (parent) areas
- Mutation is exploitative, it creates random small diversions, thereby staying near (in the area of) the parent
- Only crossover can combine information from two parents
- Only mutation can introduce new information (alleles)

Survivor Selection

- Most EAs use fixed population size so need a way of going from (parents + offspring) to next generation
- Suppose the process shown in the previous slides is repeated twice to generate 4 more offspring.
- We have now a pool of these 12 solutions (6 parents + 6 offspring) and can run binary tournament to select 6 candidates for the next generation.
- These 6 selected candidates will form the parents pool in the 2nd generation.

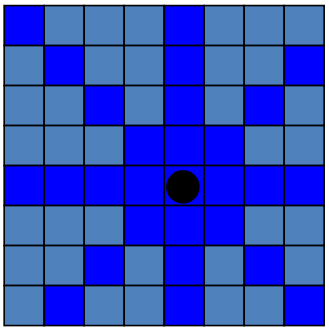
Parents	Fitness
A C B F H D E G	
H B G E A C D F	
A H G C B D F E	
E G B C D H F A	
F H A D C B E G	
C D B A H E G F	

Children	Fitness
D F B C H E G A	
F E A D C H G B	

A Typical Evolutionary Algorithm Cycle

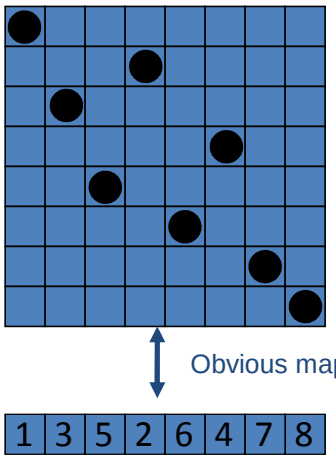
- Step 1: Initialize the population randomly or with potentially good ***solutions***.
- Step 2: Compute the ***fitness*** of each individual in the population.
- Step 3: Select parents using a ***selection procedure***.
- Step 4: Create offspring by ***crossover*** and ***mutation*** operators.
- Step 5: Compute the ***fitness*** of the new offspring.
- Step 6: Select members of population to die using a ***selection procedure***.
- Step 7: Go to Step 2 until termination criteria are met.

The 8-Queen Problem



Place 8 queens on an 8x8 chessboard in such a way that they cannot check each other

The 8-Queen Problem: Representation



The 8-Queen Problem: Fitness Evaluation

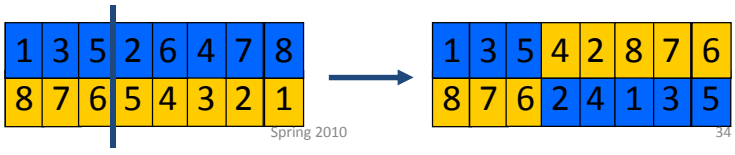
- Penalty of one queen:
  the number of queens she can check.
- Penalty of a configuration:
  the sum of the penalties of all queens.
- Note: penalty is to be minimized
- Fitness of a configuration:
  inverse penalty to be maximized

The 8-Queen Problem: Mutation

- Small variation in one permutation, e.g.:
  - swapping values of two randomly chosen positions

The 8-Queens Problem: Crossover

- Combining two permutations into two new permutations:
  - choose random crossover point
  - copy first parts into children
  - create second part by inserting values from other parent:
    - in the order they appear there
    - beginning after crossover point
    - skipping values already in child



Algorithm Design

- Design a representation
- Design a way of evaluating an individual
- Design suitable mutation operator(s)
- Design suitable recombination operator(s)
- Decide how to select individuals to be parents
- Decide how to select individuals for the next generation (how to manage the population)
- Decide how to stop: termination criterion