Introduction to Evolutionary Computation

The EvoNet Flying Circus

Brought to you by (insert your name)

The EvoNet Training Committee

Some of the Slides for this lecture were taken from the EvoNet Flying Circus

Found at:
- www2.cs.uh.edu/~ceick/ai/EC1.ppt

This week

- Pattern space
  - What is this concept?
  - What does it have to do with searching and evolutionary algorithm?
- Introduction to evolutionary algorithm
  - Introduction to their place in AI
  - Basic concepts
  - Representation
**Pattern space**

- A pattern space is a way of visualizing the problem.
- It uses the input values/parameters as coordinates in a space.
- So 2 parameters 2-D plot.
- So 3 parameters 3-D plot.
- Any more parameter the same idea by hard to visualise.

**Pattern space in 2 dimensions**

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The AND function:
- 1
- 0

3-D pattern space
A lot of techniques in AI use this idea of a pattern space. In evolutionary algorithm the idea is to search this pattern space to find the best point.

Q What is the most powerful problem solver in the Universe?

A The (human) brain that created "the wheel, New York, wars and so on" (after Douglas Adams)

A The evolution mechanism that created the human brain (after Darwin et al.)

Building problem solvers by looking at and mimicking:
- brains → neurocomputing
- evolution → evolutionary computing
**Taxonomy**

- Computational Intelligence or Soft Computing
- Neural Networks
- Evolutionary Algorithms
- Fuzzy Systems
- Evolutionary Programming
- Genetic Algorithms
- Genetic Programming
- Classifier Systems

**History**

- L. Fogel 1962 (San Diego, CA): *Evolutionary Programming*
- J. Holland 1962 (Ann Arbor, MI): *Genetic Algorithms*
- I. Rechenberg & H.-P. Schwefel 1965 (Berlin, Germany): *Evolution Strategies*
- J. Koza 1989 (Palo Alto, CA): *Genetic Programming*

**The Metaphor**

- **EVOLUTION**
  - Individual
  - Fitness
  - Environment

- **PROBLEM SOLVING**
  - Candidate Solution
  - Quality
  - Problem
The Ingredients

- $t$
- $t+1$
- reproduction
- selection
- mutation
- recombination

The Evolution Mechanism

- Increasing diversity by genetic operators
  - mutation
  - recombination

- Decreasing diversity by selection
  - of parents
  - of survivors

The Evolutionary Cycle

- Population
- Replacement
- Selection
- Parents
- Recombination
- Mutation
- Offspring
Domains of Application

- Numerical, Combinatorial Optimisation
- System Modeling and Identification
- Planning and Control
- Engineering Design
- Data Mining
- Machine Learning
- Artificial Life

Performance

- Acceptable performance at acceptable costs on a wide range of problems
- Intrinsic parallelism (robustness, fault tolerance)
- Superior to other techniques on complex problems with
  - lots of data, many free parameters
  - complex relationships between parameters
  - many (local) optima

Advantages

- No presumptions w.r.t. problem space
- Widely applicable
- Low development & application costs
- Easy to incorporate other methods
- Solutions are interpretable (unlike NN)
- Can be run interactively, accommodate user proposed solutions
- Provide many alternative solutions
Disadvantages

- No guarantee for optimal solution within finite time
- Weak theoretical basis
- May need parameter tuning
- Often computationally expensive, i.e. slow

Journals

- BioSystems, Elsevier, since 1986
- Evolutionary Computation, MIT Press, since 1993
- IEEE Transactions on Evolutionary Computation, since 1996

Summary

EVOLUTIONARY COMPUTATION:
- is based on biological metaphors
- has great practical potentials
- is getting popular in many fields
- yields powerful, diverse applications
- gives high performance against low costs
- AND IT’S FUN!
The Steps

In order to build an evolutionary algorithm there are a number of steps that we have to perform:
1. Design a representation
2. Decide how to initialize a population
3. Design a way of mapping a genotype to a phenotype
4. Design a way of evaluating an individual

Further Steps

5. Design suitable mutation operator(s)
6. Design suitable recombination operator(s)
7. Decide how to manage our population
8. Decide how to select individuals to be parents
9. Decide how to select individuals to be replaced
10. Decide when to stop the algorithm

Designing a Representation

We have to come up with a method of representing an individual as a genotype. There are many ways to do this and the way we choose must be relevant to the problem that we are solving.
When choosing a representation, we have to bear in mind how the genotypes will be evaluated and what the genetic operators might be.
Example: Discrete Representation (Binary alphabet)

- Representation of an individual can be using discrete values (binary, integer, or any other system with a discrete set of values).
- Following is an example of binary representation.

<table>
<thead>
<tr>
<th>CHROMOSOME</th>
<th>GENE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 0 0 1 1</td>
<td></td>
</tr>
</tbody>
</table>

Phenotype could be integer numbers

<table>
<thead>
<tr>
<th>Genotype:</th>
<th>Phenotype:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 0 0 1 1</td>
<td>= 163</td>
</tr>
</tbody>
</table>

Phenotype:
- Integer
- Real Number
- Schedule
- ...
- Anything?

8 bits Genotype

Phenotype:

\[ 1^2 + 0^2 + 1^2 + 0^2 + 0^2 + 0^2 + 1^2 + 1^2 = 128 + 32 + 2 + 1 = 163 \]
Example: Discrete Representation
(Binary alphabet)
Phenotype could be Real Numbers
E.g., a number between 2.5 and 20.5 using 8
binary digits

```
1 0 1 0 0 1 1
```

= 13.9609

```
x = 2.5 + \frac{163}{256} (20.5 - 2.5) = 13.9609
```

Example: Discrete Representation
(Binary alphabet)
Phenotype could be a Schedule
E.g., 8 jobs, 2 time steps

```
1 2 3 4 5 6 7 8

1 2 1 2 1 1 2 2
```

Example: Real-valued representation
- A very natural encoding if the solution we are
  looking for is a list of real-valued numbers,
  then encode it as a list of real-valued
  numbers! (i.e., not as a string of 1’s and 0’s)
- Lots of applications, e.g., parameter
  optimisation