Evolutionary Computation
Lecture 1: Introduction

Claus Aranha
caranha@cs.tsukuba.ac.jp

Department of Computer Science

July 17, 2013
In this course we will overview of the class of optimization algorithms called **Evolutionary Computation**.

The course will cover the basics of Genetic Algorithms, including theory, application and implementation. We will also discuss some common variations of GA, such as DE, GP and MOGA.
Course Goals

By the end of this course, the student should have a general view of the Evolutionary computation Field. We expect the student will be able to:

- Analyze an optimization problem and determine if it is possible to use some form of evolutionary computation method to it.
- When using a Genetic Algorithm, being able to choose appropriate operators and parameters from the literature.
- Be able to design the code for a simple Genetic Algorithm System.
- Be able to discuss theoretical concepts involving Evolutionary Computation.
Requirements and Reports

Grading

The course will be graded on a written report and a programming assignment. The assignments will be described in the last week, and the students will have 2 weeks to complete them.

Programming examples and exercises in the course will be mostly done in python or python-like pseudocode. We will use the py-evolve package a lot.

Most programming examples will be discussed in the second and third classes - I suggest you study a bit of python before then - it is fun!
Evolutionary Algorithms: The elevator pitch

- Machine Learning Method;
- An algorithm that auto-adapts to problems;
Evolutionary Algorithms: The elevator pitch

- Machine Learning Method;
- An algorithm that auto-adapts to problems;
- Auto-adaptation takes place through evolution;
- Solutions found by evolution are robust to changes in the problem;
Evolutionary Algorithms: The elevator pitch

- Machine Learning Method;
- An algorithm that auto-adapts to problems;
- Auto-adaptation takes place through evolution;
- Solutions found by evolution are robust to changes in the problem;
- Does not require too much knowledge about context;
- Deals well with multi-modal, discontinuous, high-dimensional problem spaces;
Evolutionary Algorithms: The elevator pitch

- Machine Learning Method;
- An algorithm that **auto-adapts** to problems;
- **Auto-adaptation** takes place through **evolution**;
- Solutions found by evolution are **robust** to changes in the problem;
- Does not require too much **knowledge about context**;
- Deals well with **multi-modal, discontinuous, high-dimensional problem spaces**;

How much of this is accurate?
An evolutionary Story

- Lamarck and the Giraffes;
- Darwin and the English Butterflies;
An evolutionary Story

- Lamarck and the Giraffes;
- Darwin and the English Butterflies;

The evolutionary computation analogy
Evolutionary Algorithm Example: Box2d Evo Cars
Outline of an Evolutionary Algorithm

<table>
<thead>
<tr>
<th>Problem</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solutions</td>
<td>Population</td>
</tr>
<tr>
<td>Operators</td>
<td>Search Heuristics</td>
</tr>
<tr>
<td>Utility</td>
<td>Fitness Functions</td>
</tr>
<tr>
<td>Iterations</td>
<td>Generations</td>
</tr>
</tbody>
</table>
What do Evolutionary Algorithms DO then?

1. Generate random solutions to a problem;
2. Select the best solutions by some metrics;
3. Create new solutions based on the selected best solutions;
What do Evolutionary Algorithms DO then?

1. Generate random solutions to a problem;
2. Select the best solutions by some metrics;
3. Create new solutions based on the selected best solutions;

Keep this in mind, we are going to repeat it many times.
Why do we care?

It is important to have a clear image about how recent ideas of evolution work. This can be useful for having new ideas regarding evolutionary algorithms, or to understand the world.
Why do we care?

It is important to have a clear image about how recent ideas of evolution work. This can be useful for having new ideas regarding evolutionary algorithms, or to understand the world.

Beware of forcing too much biology into your algorithms!
An individual and its genome

**Genotype**

The genetic code of an individual, determines how the development of that individual’s entire body will proceed. In general identical genome will result in almost identical individuals, and similar genomes will result in similar individuals.

**Phenotype**

The expression of the genome into an individual. A physical body that interfaces with the environment.
An individual and its genome

Genotype

The genetic code of an individual, determines how the development of that individual’s entire body will proceed. In general identical genome will result in almost identical individuals, and similar genomes will result in similar individuals.

Phenotype

The expression of the genome into an individual. A physical body that interfaces with the environment.

There are also some “Developmental” issues, but we don’t need to go deep into them right now.
What is Evolution
Evolution is the change in the distribution of genotypes in a population over time.

What about survival of the fittest?
Darwin’s theory of natural selection explains how the gene distribution changes. In other words, it is the difference between evolution and the theory of evolution.

(For example, another “theory of evolution” was Lamarck’s theory...)
Evolutionary Computation and its Relatives

- **Evolutionary Computation;**
  General term for computer methods using some form of evolutionary analogy.

- **Evolution Strategies;**
  Early evolutionary method, non-populational.

- **Evolutionary Algorithms;**
  Used with many different senses;

- **Genetic Algorithms;**
  Classic evolutionary method, individuals in the population encode candidate solutions for the problem.
Elevator Pitch

The Field and Lingo of Evolutionary Computation

Evolutionary Computation and its Relatives

- Genetic Programming;
  More “recent” method where the individuals encode programs that can be used to solve the proposed problem.

- Differential Evolution;
  Even more recent method with specialized mutation and crossover operators for genomes composed of real values.

- Interactive Evolution;
  The Fitness function is provided by a human user. Normally used in artistic domains (Evolutionary Art).

- Evolutionary Neural Networks and others;
Evolutionary Computation Lingo

Beware of mixing algorithmic terms with biological terms!

- Iteration;
- Convergence;
- Epigenetic;
- Diversity;
- Generational, non-generational;
- Genotype and Phenotype;
- Exploration and Exploitation;
- Local Search and Global Search;
- Overfitting;
- Genetic Drift;
- Candidate Solution;
Looking at Evolutionary Computation from a Algorithmic point of view

Evolutionary Computation methods as Meta-heuristic search algorithms.
Search Algorithms

Search Algorithms enumerate all the States of a problem, in order to find a goal, or optimal state.

- Breadth First Search;
- Depth First Search;
- Random Search;
- Etc;
Search Space

Search spaces are defined as the set of all the possible states for a problem. For interesting problems, the search space is usually not enumerable.

Search space qualities

Implicit, explicit, Unimodal, Multimodal, Continuous, Discontinuous, Monotonic, non-monotonic, etc
Heuristics

Problem: Finding the shortest path on a maze
How can we improve on the simple DFS, if we are looking for a way out on a grid-like maze?
Heuristics

Problem: Finding the shortest path on a maze

How can we improve on the simple DFS, if we are looking for a way out on a grid-like maze?

Hint: Dijkstra’s Algorithm
Heuristics

Problem: Finding the shortest path on a maze
How can we improve on the simple DFS, if we are looking for a way out on a grid-like maze?

Hint: Dijkstra’s Algorithm

Heuristics bring extra information that can be used to guide the search effort.
Heuristics: Example

What would be a good heuristic search for:

- A Sudoku Puzzle?
- Climbing a mountain in an unknown area?
- Playing a connect-4 game?
Heuristics: Example

What would be a good heuristic search for:

- A Sudoku Puzzle?
- Climbing a mountain in an unknown area?
- Playing a connect-4 game?

Notice that the two last examples do not include complete information about the problem.
Why is EC a search heuristic?

The basic genetic algorithm:

- Generate multiple candidate solutions;
- Select the best solutions;
- Generate new candidate solutions based on the best solutions;
Why is EC a search heuristic?

The basic genetic algorithm:

- Generate multiple candidate solutions;
- Select the best solutions;
- Generate new candidate solutions based on the best solutions;
What is EC doing, from a search-space point of view?

The individuals in a population are a sample from the search space. The evolutionary algorithm (implicitly) estimates the shape of the search space based on the fitness value of the individuals. New samples are taken from the search space based on the (implicit) estimate from the previous sample.

Let’s try to visualize this in a plane;
Comparing points of view about Evolutionary Computation

Evolutionary Analogy
- Individuals adapt to the problem;
- Operators are bio-inspired;
- Genome represents knowledge;
- Captures the imagination;

Search Meta Heuristic
- Sampling the search space;
- Operators manipulate the search algorithm;
- Representation defines the search space boundaries;
- Makes (most) reviewers happy;
Expectations of EC as a search heuristic

Generally, Evolutionary Computation usually performs well in “discontinuous” or “hard” search spaces. We can consider this to be because of the large steps taken by the crossover operator.

Evolutionary Computation is also considered to be able to generate “creative” solutions. This is usually attributed to the fact that although the fitness function is set, the way to satisfy the fitness function is not set.
Some extra concepts

- No Free Lunch Theorem
- The Curse of Dimensionality

It is important to know when GA can be the answer to a hard problem, and when it cannot.
Outline of a Genetic Algorithm

- Define the genome representation and the fitness function, based on the problem that you want to solve;
- Generate random individuals for the first generation;
- Select individuals with the highest fitness value, based on the selection operator;
- Generate new individuals, based on the crossover and mutation operators;
- Repeat the above until a pre-defined stop criteria is met;
Fitness Function

The first decision that must be made when designing an Evolutionary Computation algorithm is, generally, the Fitness Function.

- This is the part of the algorithm that will guide the state search;
- An imprecise definition of the fitness function may lead to the algorithm “cheating”;
- Pay close attention if you are going to define a minimizing or maximizing fitness function;
- Analysis of the fitness function may say something about the search space.
Genome Representation

By defining your genome representation, you define the entire search space. Make sure to define a search space that include the location for your optimal solution. (Good and Bad examples)
Population

The initial population is usually defined randomly. Although it is not unheard of initializing the population to certain values, or to insert a few seeds into the population. Just be aware that adding a few viable individuals will strongly bias your search.
Selection

The selection operator will guide the search, by directing the sampling of new solutions, based on previous results, as measured by the fitness function.

One way to define the selection operator is to simply take the best individuals in the population. Can you see why this would create more problems than it would solve?
Roulette Selection (or Fitness Proportional Selection), attributes a probability of selection to each individual based on their fitness value.
Tournament Selection

Tournament Selection randomly selects a fixed number of individuals from the population as a first step. On the second step, it selects the individual with the best fitness from those selected in the first step.

How does the number of individuals pre-selected affects the result?
Elitism

In Elitism, we set aside a number of individuals with the best fitness value, and copy them directly to the next generation.

Elitism is intended to avoid losing information to the random nature of EC. Is there any ways to prevent that loss of information?
Random Selection

Always remember to compare your evolutionary algorithm with the random selection. If your problem can be solved by random selection, maybe it is too simple for evolutionary algorithms.
Crossover Operators are the main way that new individuals are generated from current ones. The Crossover operator is strongly dependant on the decision made for the Genome representation.
Mutation Operators are used to add “new information” to the population. They modify existing individuals in order to add new combinations of genomes not yet existing in the population.
Implementation Concerns regarding Evolutionary Algorithms

- Object Oriented implementation;
- General Fluoxgram of an Evolutionary Algorithm
Libraries for Evolutionary Computation

- py-evolve;
- ECJ - evolutionary computation in Java;
- Open-Beagle - evolutionary computation in C++
EvoCircle

A simple evolutionary program is used to calculate load balancing of eigenvalue calculation methods.
Adding a bit more to your evolutionary portfolio.

- Genetic Programming,
- Multi Objective Genetic Algorithms,
- Parallel Genetic Algorithms,
- Differential Evolution,
- Memetic Algorithms,
- Et cetera.
Real valued representation and operators;
Tree-based representation and operators;
Mini-Python tutorial