Introduction to Evolutionary Programming And Genetic Algorithms

After scientists became disillusioned with classical and neo-classical attempts at modelling intelligence, they looked in other directions.

Two prominent fields arose:

- Connectionism (neural networking, parallel processing)
- Evolutionary computing (Genetic Algorithms, Genetic Programming, etc…)

Artificial Intelligence takes example from natural solutions:

- Artificial Neural Networks → The Brain
- Fuzzy Logic → Reasoning and Experience
- Evolutionary Programming (inc. Genetic Algorithms) → Natural Selection and Evolution

The Principles of Natural Selection and Evolution

Selection:

- If there is a pool of various individuals, those who are **fit** enough to copy themselves survive, if not, they extinguish.

- Reproducing by copy means that the **fittest** individuals populate the environment while the unfit eventually go extinct.

- But this only works if we have **variety** to start with.

- Natural Selection happens by letting the individuals perform (i.e. “live”) in an environment where they have to solve a problem (“survive” for long enough to be able to reproduce)

The Principles of Natural Selection and Evolution

Evolution:

Can you evolve by copying? Can you adapt this way?

If an organism **copies** itself to reproduce how can it evolve?

> **Mutation**.
>
> It is a renewable source of **variety**.
>
> But it is dangerous and absolutely random therefore an *effective* but not very *efficient* way to evolve.

The Principles of Natural Selection and Evolution

Evolution:

If different and already tested good treats could be shared it would be easier!

> **Cross Over** ➔ Sexual reproduction
>
> It is much safer and not so random, therefore more *efficient* than mutation.
>
> But is it more *effective*?
>
> It does not provide **renewable variety**.
>
> (Once all combinations have been produced, there will be no more variety).
>
> So we still **NEED Mutation** to maintain **variety**.

The Principles of Natural Selection and Evolution Applied to Problem Solving

Problem Solving:

From observing Natural Selection and Evolution we can see that:

- Neither Selection nor Evolution is a **solution** to a problem.

- They provide a way to **search** for a solution by evolving it.

- Therefore Evolutionary Programming can be seen as a methodology for searching solutions rather than a solution in itself.

- The solution is searched by trying it in the actual problem rather than trying to find the inverse model of the problem!

  ➔It is a **Direct** solving method rather than an **Inverse** one.

Evolving Solutions to Problems

The Basic Genetic Algorithm

Symbolic AI vs Genetic Algorithms

- Most symbolic AI systems are very static, they can usually only solve one given specific problem
- If the problem were somehow to change, these systems could have a hard time adapting to them
- Genetic algorithms can combat these problems

- They are basically algorithms based on natural biological evolution
- The architecture of systems that implement genetic algorithms (or GA) are more able to adapt to a wide range of problems
- Genetic algorithms can be incredibly efficient if programmed correctly.

General Algorithm for Genetic Algorithms

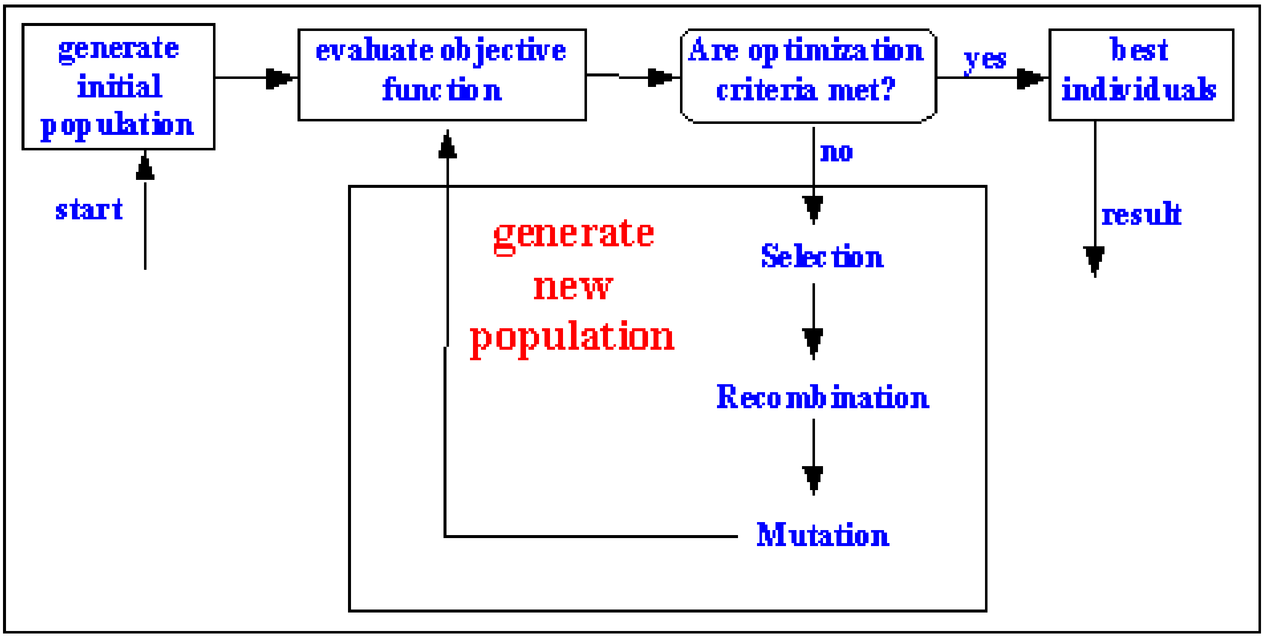
Genetic algorithms are not too hard to program or understand, since they are biological based.

Thinking in terms of real-life evolution helps.

The general algorithm for a GA is:

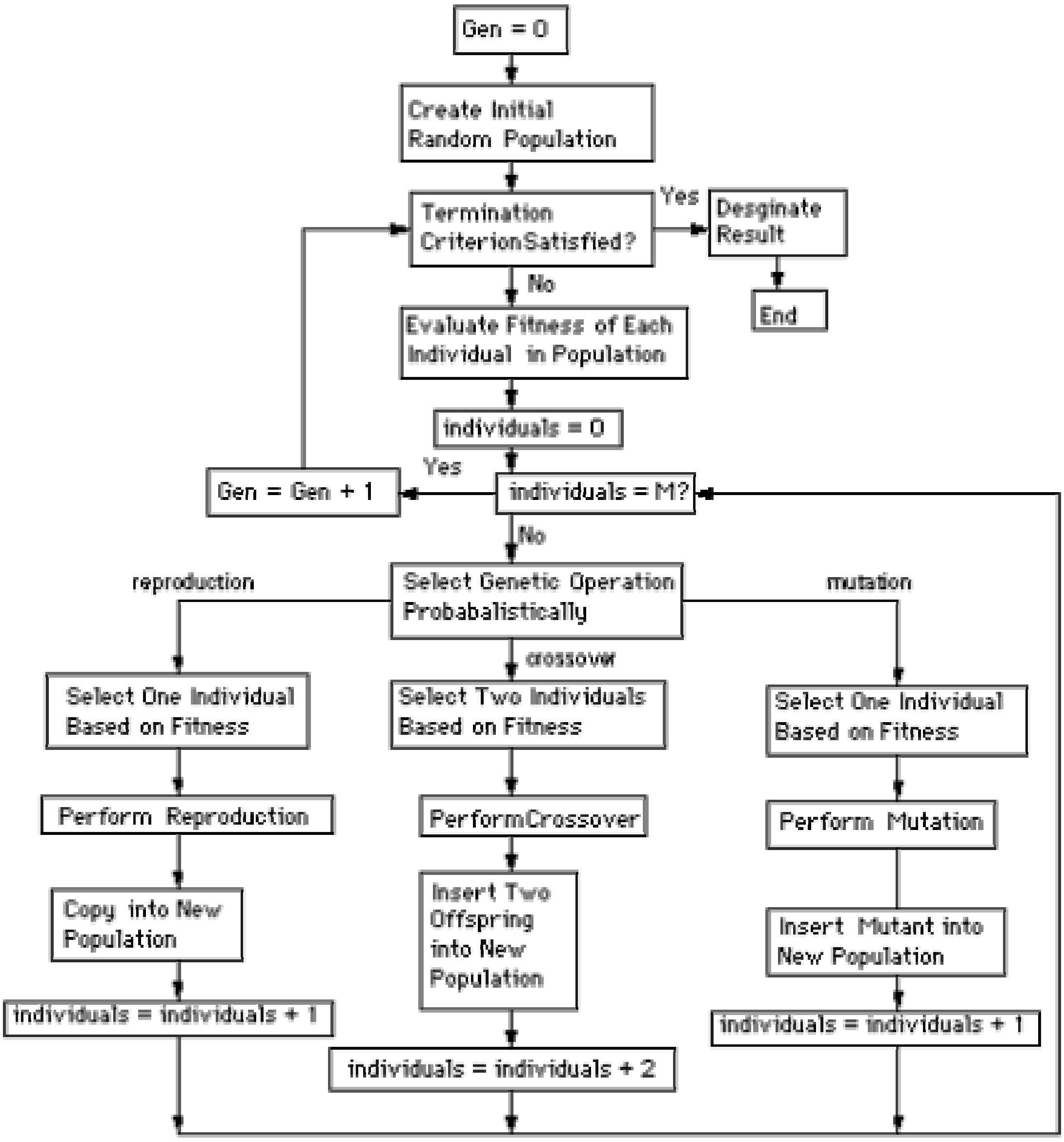
- Generate a large set of possible solutions to a given problem (**initial population**)
- Evaluate each of those solutions, and decide on a "**fitness level**" ("survival of the fittest")
- From these solutions breed new solutions (the next **generation**)
  - The parent solutions that were more "fit" are more likely to reproduce
  - While those that were less "fit" are more unlikely to do so
- Solutions are **evolved** over time, by repeating the process each **generation**.
- **Terminate** when a solution has been found or other **termination criteria** has been met

General Algorithm for Genetic Algorithms



General Algorithm for Genetic Algorithms

Flowchart for Genetic Programming



General Algorithm for Genetic Algorithms

Create a Random Initial Population

- An initial population is created from a **random** selection of solutions
- These solutions have been seen as represented by **chromosomes** as in living organisms
- The **genetic information** defines the behaviour of the individual
- A **chromosome** is a packet of genetic information organised in a standard way that defines completely and individual (solution)
- The genetic principles (way in which that information encodes the individual) enable the individuals to evolve in a given environment
- The genetic structure (way in which that information is packed and defined) enables the solutions to be manipulated
- The **genetic operands** (way in which that information can be manipulated) enables the solutions to reproduces and evolve

General Algorithm for Genetic Algorithms

Evaluate Fitness

- A value for **fitness** is assigned to each solution (chromosome) depending on how close it actually is to solving the problem

- Therefore we need to define the problem, model it, simulate it or have a data set as sample answers

- Each possible solution has to be tested in the problem and the answer evaluated (or marked) on how good it is

- The overall *mark* of each solution relative to all the *marks* of all solutions produces a fitness ranking

General Algorithm for Genetic Algorithms

Produce Next Generation

- Those chromosomes with a **higher fitness** value are more **likely** to reproduce offspring
- The population for the **next Generation** will be produced using the **genetic operators**
- Reproduction by **Copy** or **Crossing Over** and **Mutation** will be applied to the chromosomes according to the selection rule
- This rule states that the fitter and individual is, the higher the **probability** it has to reproduce
- Note that this works with **probabilities**!
- Why give a probability rather than choosing explicitly the best individuals?

General Algorithm for Genetic Algorithms

Next Generation or Termination

- If the population in the last generation contains a **solution** that produces an output that is close enough or equal to the desired answer then the problem has been solved.

- This is the ideal **termination criterion** of the evolution

- If this is not the case, then the new generation will go through the same process as their parents did, and the **evolution** will continue

- This will iterate until a solution is reached or another of the **termination criteria** is satisfied

- A termination criterion that **always must be included is Time-Out** (either as computing time or as number of generations evaluated)

- Since one drawback of Evolutionary Programming is that is very difficult (impossible most of the time) to know if the ideal termination criterion is going to be satisfied, or when

Evolutionary Programming

Difference between various names:

- What is a **Genetic Algorithm**?
  - Named by John Holland in the 70's.
  - A string of 1's and 0's to encode different solutions in the form of vectors of values or parameters.

- What is **Genetic Programming**?
  - Named by John Koza in the early 90's.
  - Evolving LISP programs using the GA principle.

- What is the **Genetic Paradigm**?
  - John Koza realised that not only programs could be "evolved" but other elements like equations, sets of rules, etc…
  - Then the application field exploded.

- What is **Evolutionary Programming**?
  - So… at the end, nearly any type of computing tool could be "evolved" in some way using the GA principles
  - The most generic term came out.