# Evolutionary Algorithms and Genetic Programming

"*Creativity, it has been said, consists largely of re-arranging what we know in order to find out what we do not know.*" **George Kneller**

So that's what George Kneller said defining creativity, anyone could find many other different wordings for definition of creativity, while vast majority ends up meaning more or less same, "*ability to create something new …*". But I guess in usual sense we also want that new thing to be something meaningful, if not useful. Often we define spark of creativity when we see spectacular inventions, non-obvious thinking pattern, like those of Leonardo da Vinci. Now the question is this creativity something that belongs to human only? Well, if the definition says creating something new demonstrates creativity, then other animals definitely have it. Like Chimpanzee is known to use tools, in lab environment Rats can solve maze. Shouldn't these be considered as creativity as well. Probably yes or may be of course yes. But we would also acknowledge that no Chimpanzee is known to have designed a helicopter like Leonardo did few hundred years back. As far as we know such level of creativity only belongs to members of human species. As human invented machines grow in computational power, surpassing human in many order of magnitude in their speed, a question arises how long computers will play dumb. Will there be a point when computers will be able to invent useful things, and quality wise those inventions will be as good as or better than their human counterparts.

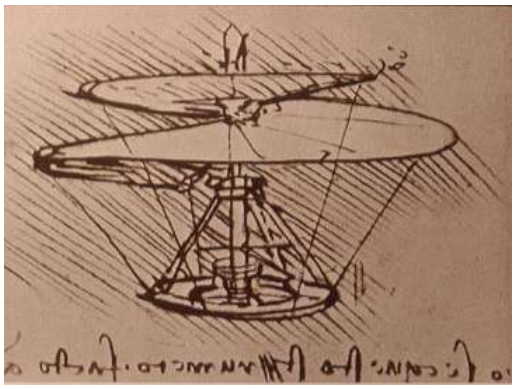


**Figure 1: Leonardo's flying machine, should computer be able to have such ingenuity?**

This is an important possibility likely to gain momentum is next decade or two. Exponential growth in nanotechnology and robotics that is taking place now, and expected to continue in next decade should open many avenues where machines are creative, and will be used to solve some of the original complex problems with their inventiveness.

One such avenue is use of ***Evolutionary Algorithms*** or ***Computation*** that clearly demonstrates machines ability to come up with creative solutions. Before I go on defining what evolutionary computation is or how it works, let's visit the ***Tone Discriminator*** evolved by Adrian Thompson's[1] machine in 1996. It uses fewer than 40 programmable logic gates, and no clock signal in an FPGA (Field Programmable Gate Array). This is an extremely small design for such a device, no known human designed such device exists that uses this little number of gates. Applying laws of evolution machine has come up with this design in roughly 5000 iterations, and it works perfectly. The interesting part is there appears to be some components in the board that are not connected with the rest of circuitry in any of the ways, while still influences overall operation. Figure below is from Adrian's paper, where the gray cells are not directly connected still are non-removable part of the Tone Discriminator. It's not well understood how these gray cells are influencing, may be their magnetic field

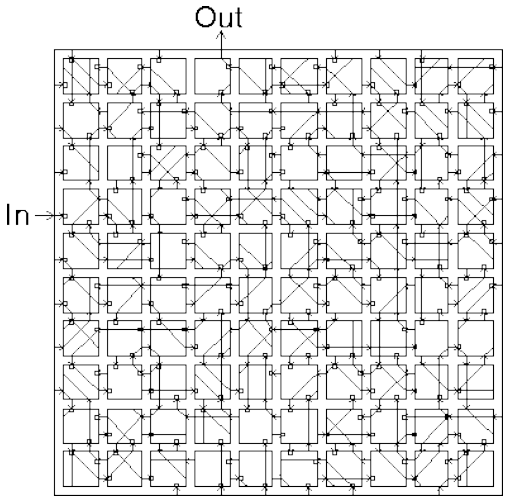



**Figure 2: Original circuit**

helps (which clearly a human designer won't consider), or something to do with the electrical load. Anyway the point is this evolutionary algorithm is able to design a circuit and outperform human in creativity.

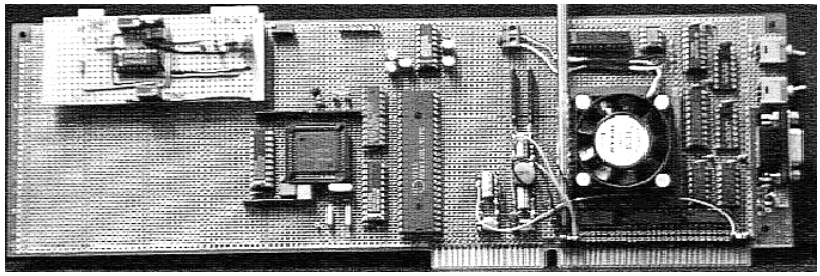


Figure 3: Circuit board used

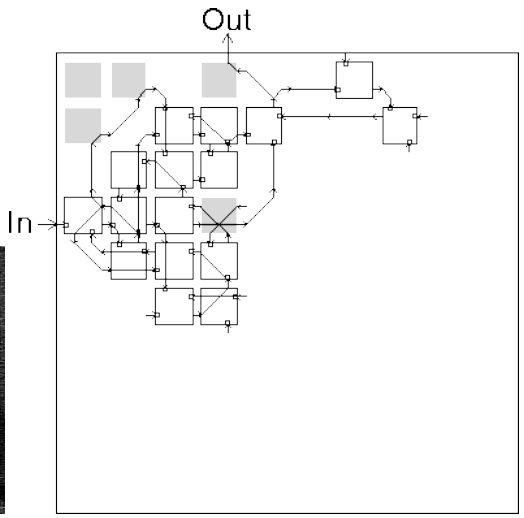



Figure 4: Evolved effective circuit

## Definitions

There are several meta-heuristic algorithms known today in computer science, including random optimization, simulated annealing, even greedy algorithm. One of them is evolutionary algorithms. They use mechanisms inspired by biological evolution, like reproduction, mutation, recombination and natural selection. Usually the problem space is described by a set of genome, then operators like mutation, reproduction are applied to create candidate solutions, and finally a cost function determines which solution to retain (fitness). These operations are applied repeated times and due to natural selection, candidate solutions improve over time.

Based on implementation details evolutionary algorithm may be divided into several categories, one being Genetic Algorithms, one other Genetic Programming among few more.

Genetic Programming gives solutions that are in the form of computer program. Fitness of the program is determined by their ability to solve a computational problem.

There has been many interesting solutions that were provided by evolutionary algorithms like the one example I gave above. Some of the solutions obtained by genetic programming are as good as already patented human inventions. I will discuss more on this later.

## Basics of Genetic Programming

Genetic programming can be thought as an automated invention machine which works by applying evolutionary algorithms in the space of computer programs. To illustrate an example let's have a simple program written in C:

```
int Add(int a, int b)
{
	int t = MAX_INT - a;
	if (t < b)
		return 0;
	else
		return (a + b);
}
```

Notice that we can construct a tree for this routine:

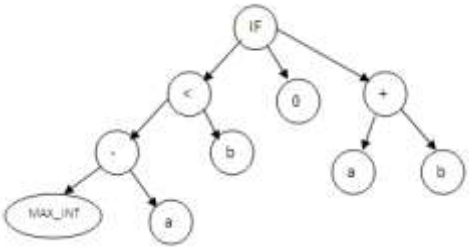


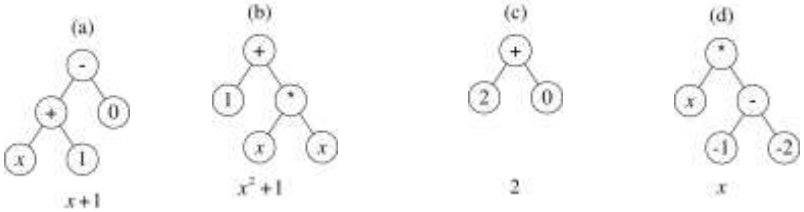

Figure 5:Program Tree

**Or: (IF (< (- MAX_INT a) b) 0 (+ a b))**

Given this type of construction we can create random programs using available functions such as +, -, *, /, %, IF_ELSE, along with terminals like a, b, or random constants. Random programs are syntactically valid, executable but could be of different size and shapes.

To use Genetic Programming we take an objective (such as find a computer program with one input, where the output is equal to input), figure out a terminal set, and a function set and then create a fitness function (for above objective we can have fitness as the absolute value of the difference between input and output). Once these preparatory steps are complete:

- Initial population of random programs are created, such as we can have (for above objective):




- Apply genetic operations based on their fitness, so a more fit individual gets more priority (for example(d) here wil get more priority):
  - **Reproduction**: copying individual as is.
  - **Mutation**: changing randomly picked entire sub-tree
  - **Crossover**: crossing nodes between two individual
  - Architecture altering operations
- With the new population iterate the process over and over, until reasonable solution is found.

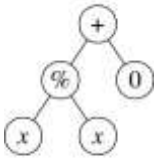


Figure 6:Mutation in (c) at *2* position

So this is briefly how genetic programming works. While the process appears fairly simple, it can give remarkable results in practice. John Koza[2,3] at Stanford (widely known as the father of GP) has shown many applications of GP in last few decades. Most of the information here I have presented is based on his papers.

A more complex program would typically make use of additional improvements like:

- Reuse of code by subroutine.
- Data structures could be added like Stacks, Queues, Lists, Rings.
- Automatically defined loops (ADL).
- Automatically defined recursions (ADR).

Other interesting techniques include Subroutine creation, deletion, argument duplication, branch duplication, all of these can create seemingly complex programs to solve a particular problem.

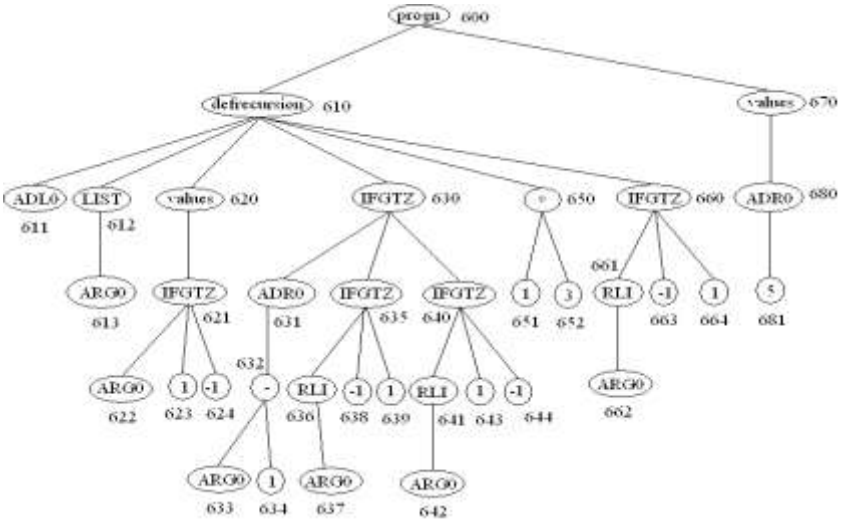



Figure 7:Genetic Program with loops and recursions

## A GP Example: Circuit Synthesis

This has been demonstrated by John Koza. Basic components are Registers, Capacitors, Inductors, Diodes, or Transistor. There are several topology modifying functions such as SERIES/PARALLEL division, VIA or FLIP. There would be some Development Controlling functions needed for circuit growth (see figure below) like END, NOP, SAFE_CUT etc.

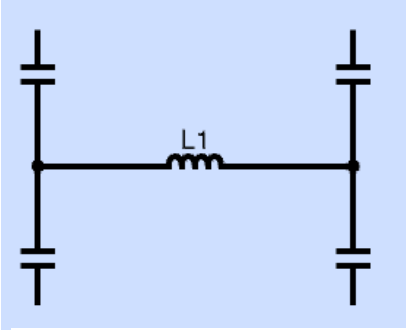



Figure 8:Parallel division

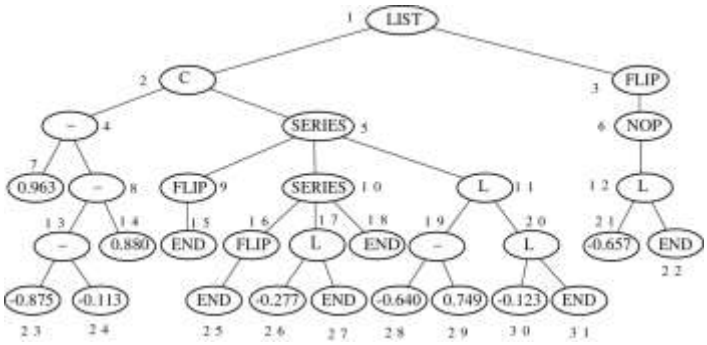



Figure 8b: Developmental GP

Once we have the preprocessing ready we can start with a program tree and an embryonic circuit, try evaluating its fitness. In this case circuit simulator is used to find the result of designed circuit. So for example if a low pass filter is tried to generate (or invent to be precise), GP algorithm ends up with a circuit that is surprisingly similar to George Campbell of AT&T invented in 1917 (US Patent 1227113). Genetic program also designs another circuit which is already known as Zobel filter (after Otto Zobel, as the inventor in 1925).

In the end many 20th century patented inventions were generated from genetic algorithm. Afterwards running the algorithm resulted in several 21st century inventions to be invented again by the machine.

What strikes me most is that Genetic Programming goes on to invent several novel circuit designs which are yet to be invented by a human.

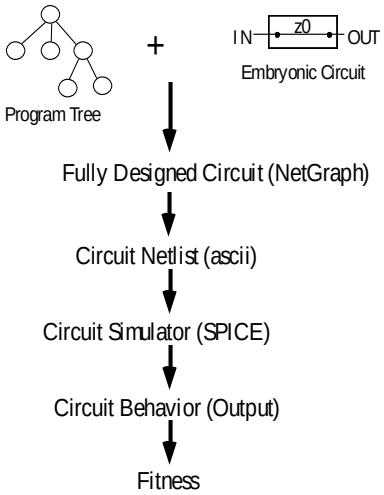



Figure 9: Fitness calculation

## Other examples

And of course it's not limited to circuit designs here's as example of re-engineering biological cell metabolic pathway.

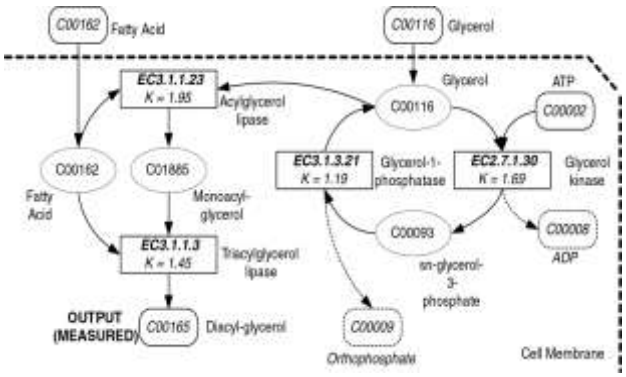



Figure 11: Original metabolic pathway

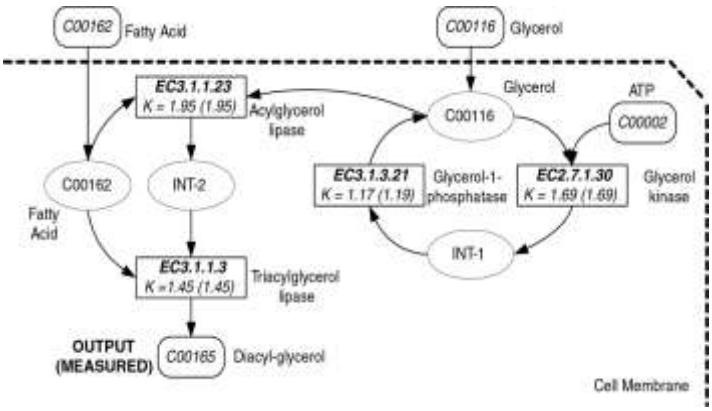



Figure 10: Evolved new pathway

NASA evolved a satellite antenna[4] (launched in 2004) using Genetic Programming. Apparently there is a whole list of applications where Genetic Programming gives the known optimum results compare to other existing algorithms:

- Mechanical design, control
- Bio-informatics
- Classification
- Data mining
- System Identification
- Forecasting
- Analyzing Genome, Protein data
- Application in Financial Sector
- Areas where human programming is difficult like, parallel programming, cellular automata, swarm intelligence.

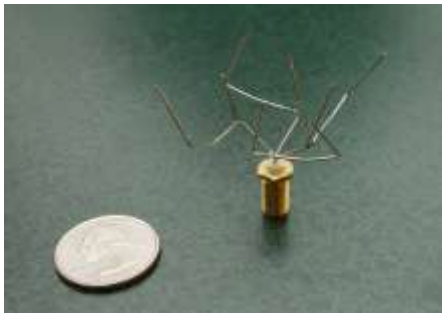


Figure 12: NASA Evolved Antenna

## Remarks

One of the disadvantages of Genetic Programming would be the time required find a solution. A decades back it would have been difficult to solve some of the problems due to lack of computing power. To re-invent some of the 21st century patents using Genetic programming takes 1000 Pentium II level machines substantial amount of time. And no free lunch theorem shows while genetic programming is able to find optimum solution some of the times, they can be outperformed by more field specific algorithms. In the end Genetic Programming is an interesting area in computer science particularly when problem space not clearly understood or less developed, it can come up with spectacular results, and in many cases finding field specific algorithm could be daunting, may be Genetic Algorithm itself could be used for that.

## References

1. Artificial Evolution in the Physical World. Evolutionary Robotics: From Intelligent Robots to Artificial Life 1997: Adrian Thompson.
2. Koza, John R. 1990a. Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems. Stanford University Computer Science Department technical report STAN-CS-90-1314. June 1990.
3. Koza, John R., Keane, Martin A., Streeter, Matthew J., Mydlowec, William, Yu, Jessen, and Lanza, Guido. 2003. Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers. ISBN 1-4020-7446-8.
4. An Evolved Antenna For Deployment On Nasa's Space Technology 5 Mission: Jason D. Lohn, Gregory S Hornby, Derek S. Linden.