# Evolutionary computation has been promising self-programming machines for 60 years – so where are they?

March 27 2018, by Graham Kendall

Credit: Julia M Cameron from Pexels

What if computers could program themselves? Instead of the laborious job of working out how a computer could solve a problem and then

writing precise coded instructions, all you would have to do is tell it what you want and the computer would generate an algorithm that solves your problem.

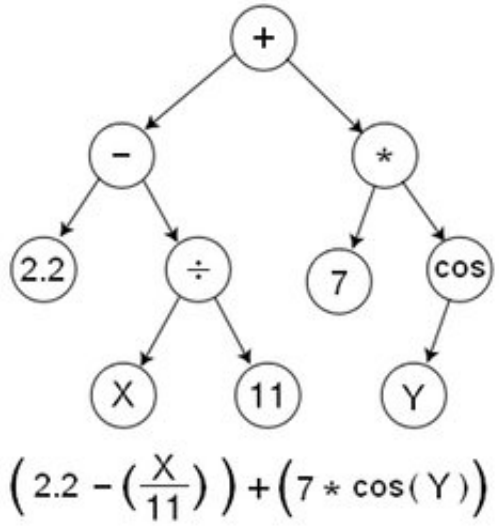
Enter evolutionary computation, which can be seen as a type of artificial intelligence and a branch of machine learning. First suggested in the 1950s, evolutionary computation is the idea that a computer can evolve its own solutions to problems, rather than humans having to go through a series of possibly complex steps to write the computer program ourselves. In theory, this would mean computer programs that might take weeks to program manually could be ready in a matter of minutes.

This idea enabled computers to solve complex problems that may not be well be understood and are difficult for humans to tackle. Computer scientists have used evolutionary computation on many problems, including formulating the best mix of ingredients for shrimp feed, portfolio optimisation, telecommunications, playing games and automated packing.

And researchers who have been studying evolutionary computation for over 60 years have made tremendous advances. It is even the subject of several scientific journals. Yet, as I noted in a recent paper, the idea still isn't used widely outside the research community. So why isn't evolutionary computing evolving faster?

## How does evolutionary computation work?

Evolutionary computation draws on Charles Darwin's principles of natural evolution, commonly known as survival of the fittest. That is, the weakest (less well adapted) members of a species die off and the strongest survive. Over many generations, the species will evolve to become better adapted to its environment.




Genetic programming tree. Credit: Wikimedia

In evolutionary computation, the computer creates a population of potential solutions to a problem. These are often random solutions, so they are unlikely to solve the problem being tackled or even come close. But some will be slightly better than others. The computer can discard the worst solutions, retain the better ones and use them to "breed" more potential solutions. Parts of different solutions will be combined (this is often called "crossover") to create a new generation of solutions that can then be tested and the process begins again.

Another important element of evolutionary computation, as with natural selection, is mutation. Every so often a small, random change is made to one of the solutions being tested. This means new potential solutions can be created that wouldn't be possible from just using crossover.

Hopefully a combination of crossover and mutation will produce new potential solutions that are better than their "parents". This might not happen every time, but as more generations are produced, better

solutions are more likely to emerge. It's not unusual for evolutionary computation to involve many millions of generations, just as natural selection can take many millions of years to noticeably alter a living species.

One of the most popular types of evolutionary computation is genetic programming. This involves one computer program evolving another working program to tackle a specific problem. The user provides some measure of what comprises a good program and then the evolutionary process takes over, hopefully returning a program that solves the problem.

We can trace genetic programming back to the late 1980s, with one of the main proponents being John Koza. But even though it has since made significant research advances, genetic programming is not used on a daily basis by commercial organisations or home computer users. Given how tricky it can be to develop software systems that work effectively and efficiently, it would seem sensible to get computers to help in the same way they are changing many other industries.

**Why hasn't evolutionary computation been adopted?**

The commercial sector hasn't embraced evolutionary computation as it has other technologies developed by researchers. For example, 3-D printing was invented in the 1980s and after a long period of development is now being used in industrial manufacturing and even by people in their homes. Similarly, augmented reality, virtual reality and artificial intelligence have emerged from the research community and become major products for big tech companies.

One of the key issues holding evolutionary computation back is the failure of researchers to focus on problems that the commercial sector would recognise. For example, computer scientists have intensively

studied how evolutionary computation could be used to schedule exam timetables or working out routes for vehicles.

But researchers often only study simplified versions of problems that are of little use in the real world. For example, many vehicle routing simulations involve calculating the distance between two points using a straight line. Vehicle routes in the real world rarely follow straight lines, and have to contend with one way systems, breakdowns, legal issues (such as how long before a driver must rest), time constraints and a whole lot more. However, this complexity is actually where evolutionary computation could help. If you can adequately define the problem as it occurs in the real world, then the evolutionary algorithm should be able to deal with its complexity.

Another problem is that the solutions evolutionary computation generates are often hard to explain. For example, even though a genetic programming system might create a solution with a perfect outcome, how it actually works might be a mystery to a human programmer as the system may have produced complex code that is difficult to interpret and understand.

An evolutionary computation system is also complex to implement and support and this may put off some commercial organisations. It would help if there was an easy-to-use framework that hid much of the underlying complexity. While these frameworks exist in the scientific community, they are not easily accessible by the commercial sector, never mind home users.

IBM's famous computer architect Frederick Brooks said that you cannot tackle increasingly large software development projects simply by throwing more people at them. It would be an immense help to the software development industry if, instead of having to manually develop every piece of a system, developers could specify the requirements of its

key parts and let an evolutionary process deliver the solutions.

This article was originally published on The Conversation. Read the original article.

Provided by The Conversation

Citation: Evolutionary computation has been promising self-programming machines for 60 years – so where are they? (2018, March 27) retrieved 7 May 2024 from https://phys.org/news/2018-03-evolutionary-self-programming-machines-years.html