Nature-Inspired Algorithms

Author: Maciej Świechowski Ph.D.

Introduction

"Look deep into nature, and then you will understand everything better" – Albert Einstein

Since ancient times, nature has been serving as inspiration for humans in the shaping of our civilization and culture. A plethora of brilliant inventions happened just through careful observation of nature. For example, geckos' feet with millions of microscopic sticky hairs inspired researchers to develop a similar tape-like material. Various synthetic materials were invented after analysis of the natural counterparts, especially chitins and furs of animals. Many shapes of machines were inspired by animals, in particular, those animals which really excel at doing something – the fastest runners, the best swimmers or the most efficient energy savers. The dream of flight sparkled in our minds probably because we were envy of all those colourful birds flying around. The ultra-fast Shinkansen Bullet Train's nose was modelled by a bird-watching engineer Eiji Nakatsu, which helped to reduce power consumption and the noise problem and enabled faster speeds. We could keep going and not stop…

How does it apply to Computer Science?

While in natural sciences such as physics, chemistry, biology or geology, it is quite easy to grasp the concept of inspiration by nature (even the name "natural sciences" helps), it might not be so obvious in computer science / information technology. In the latter field, such an inspiration can be twofold: either in the "hardware" aspect – that is the way computer are built or the "software" aspect – the algorithms and data structures used within computer programs. In this article, I am focusing solely on the second case.

Let's start by defining what a computer algorithm is. You may look at the algorithm in a similar way to a cooking recipe. An algorithm is a set of steps to follow to solve a problem.

These steps then need to be translated to a code in a programming language – a representation understandable by a computer. There are many programming languages available out there.

An exemplar algorithm is how to find the greatest common divisor of two or more numbers, a shortest path between two vertices in a graph or how to draw a line between two points on a screen.

General Purpose (GP) Algorithms

Most nature-inspired algorithms are of general purpose. It means that they are not tailored for one task only, but can be adapted to a variety of problems. Think of them as a general recipe of baking a cake (but not any

particular one). General purpose algorithm share the common base – the steps that programmers need to put in the code. However, usually, those steps have some room of placing a problem specific requirements.

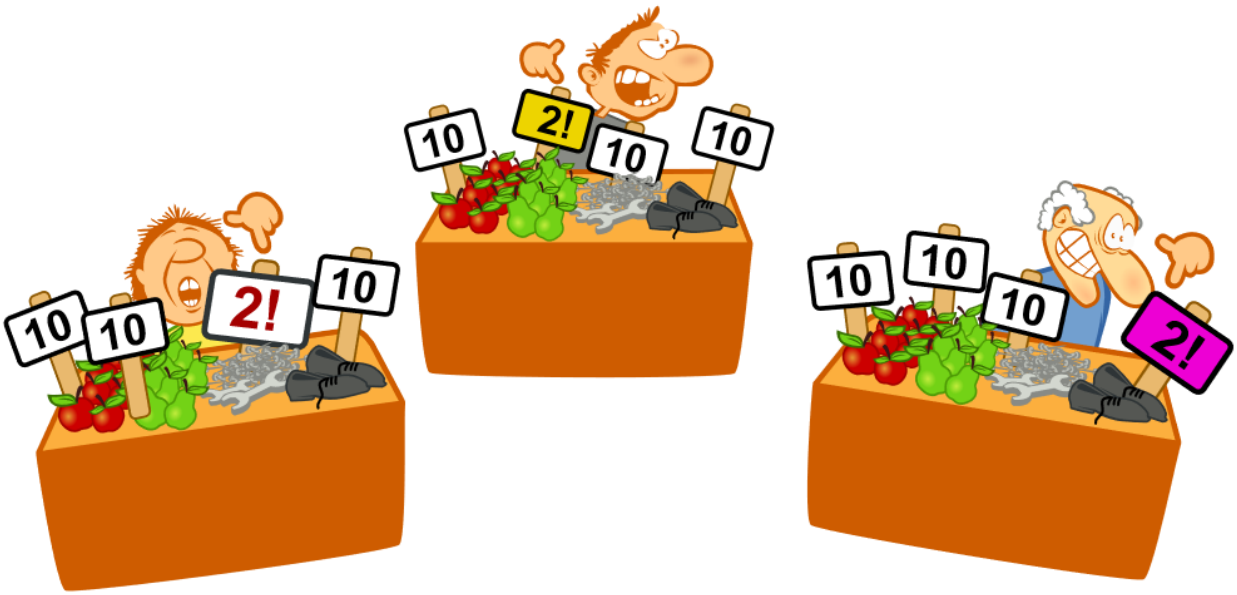
GP algorithms are neat, because once we have them, it is usually quicker to apply them to a particular problem than to figure out the solution from scratch. Moreover, GP algorithms are subject of studies themselves. In the literature, you will commonly find them under the name: metaheuristics. There is an established theory, continuously polished, on how they work, how efficient they are, and which GP algorithm is the most suitable for a particular problem.

There is a multitude of GP algorithms as well as there are many nature-inspired algorithms. One might ask: why is one not enough? Well, although such algorithms are universal, there are still problems which some algorithms will be better at and some others will be worse. So the next question arises: don't all of them solve the problem equally well? No, because the algorithms we are talking about are the so-called heuristic ones.

- Exact algorithms – they give the best possible solution to the problem. They are good for simple problems, but for many of problems (more complex ones), they are too slow to compute, hence, completely not applicable in practice.
- Heuristic algorithms – they solve a problem more quickly using some assumptions or simplifications. They find a so-called approximate solution, which might be the best possible one but does not have to be. When the solution is not optimal, it is usually still "good enough".

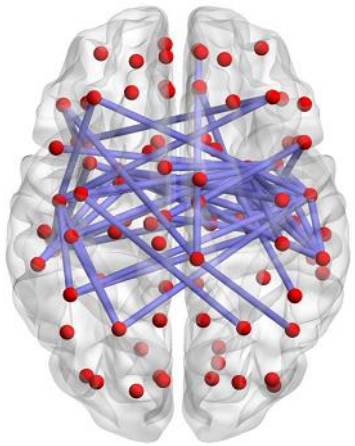
In 1997, a mathematician David Wolper and William Macready wrote an article called "No Free Lunch Theorems for Optimization". Simplifying their words, it says that there is no single best approximate (heuristic) algorithm. If we have two algorithms A and B, we can always find problems, in which either of them will be better. The performance of each algorithm is averaged across all possible problems.

The conclusion is that, we need a toolbox of various tools (algorithms) to repair a car (solve a problem).



A Couple of Nature-Inspired Algorithms

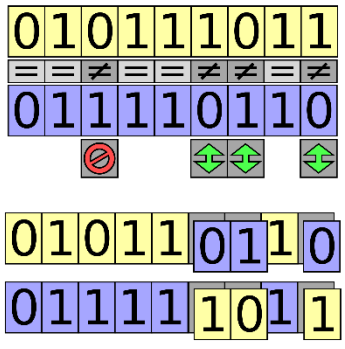
Artificial Neural Networks (ANNs)



The inspiration is a biological brain. Artificial Neural Networks are universal computational models used in computer science, especially in the areas of artificial intelligence and machine learning. Essentially, they are big data processing graphs. They compromise of nodes (neurons) and connections (synapses). Such a structure converts the input signal to the output signal, which is a loose analogy to transfer of electric impulses in biological brains. The idea is to build a network, represent a problem instance as the input signal and the solution as the output signal. Then ANNs are trained to provide a correct response to the input signal. The training can be performed, for instance, using examples (training samples) that we know what the correct output should be. Ideally, the network will learn how to respond for the unknown samples too. It is often quite difficult to find the proper balance in the network complexity, so it does not lose its ability to generalize. The issue of having learnt to properly respond to the training examples and failing to generalize is called network's overfitting. There is a rich mathematical apparatus beyond artificial neural networks, which is not covered here. Because of that, it is often not easy to explain why a trained network works. It often does solve a problem, but not provide insights about how it exactly the problem was solved. Therefore, Artificial Neural Networks are considered "black boxes" – they hide the internal details of operation.

Genetic Algorithms and Evolutionary Algorithms

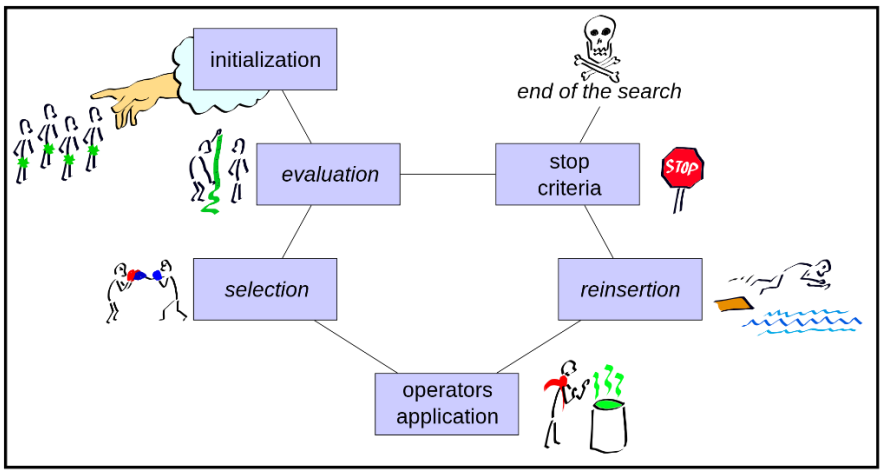
Genetic algorithms (GA) are a subset of evolutionary algorithms (EA). The first ones draw inspiration from the Darwinian's natural selection and genetics, while the latter ones refer to any kind of evolution.



Both GA and EA belong to the class of population-based methods. They operate with a population of individuals also referred to as phenotypes. Each individual encodes a candidate solution to the given problem. A common representation is an array of bits. At the start, the initial population usually consists of random solutions, possibly even invalid or bad quality ones. The idea is to employ the survival of the fittest process to "breed" new solutions until an acceptable one is found (or we ran out if time).

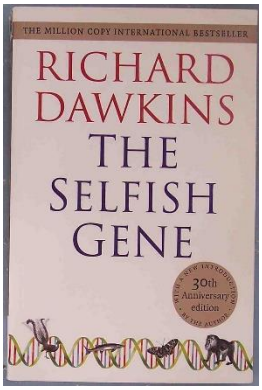
This is done thanks to selection pressure and genetic operators. The selection, as the name implies, selects the best individuals from the current generation to the next generation based on their fitness computed by the so-called fitness function, which is typically provided by the programmers.

The standard genetic operators are crossover and mutation. Crossover is the way how parent solutions are combined to create a child solution. Mutation affects a chromosome of an individual to maintain genetic diversity. Mutation is usually applied with a small probability for each chromosome.



General schema of an Evolutionary Algorithm (EA)

Memetic Algorithms



Memetic algorithms build upon genetic algorithms and add additional layer to them. The term "meme" was coined by Richard Dawkins and denotes "an idea, behavior or style that spreads from person to person within a culture". What it says in the context of genetic algorithms vs. memetic algorithms is that information between individuals does not have to be transferred only between generations, but it can be exchanged also during the lifetime of the individuals.

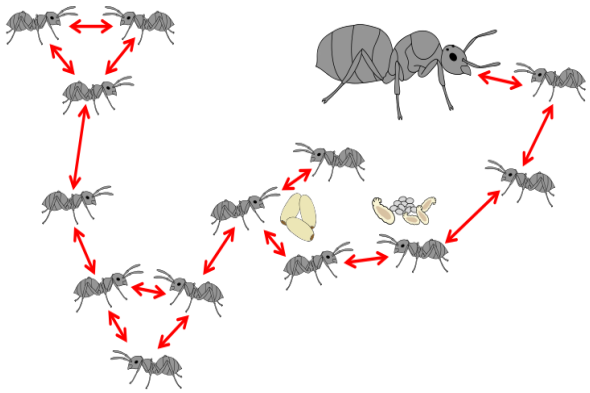
The idea is typically applied in a form of local improvement procedures for a chosen set of individuals. Such improvement procedures are often implemented with the help of some domain knowledge for a problem.

Ant Colony Optimization

This algorithm is an example of the so-called swarm intelligence methods. Many relatively simple organisms produce sophisticated behavior. The algorithm was proposed by an Italian PhD student – Marco Dorigo. The initial inspiration was the way how ants find a path between their colony and a piece of food. At the beginning, they behave rather randomly. When they find food, their return to the colony laying down pheromone trails. The more pheromone trails there are on a particular path, the more likely it is for the next ant to follow the same route. After some time, all ants will be following the shortest path between the colony and the food. When suddenly and obstacle is placed on this path, the ants will be initially stopped by it, but after some time, new pheromone trails will be placed to avoid this obstacle.

ACO algorithms are similar. They are population-based, like GAs, but units in the populations are called (virtual) ants instead of phenotypes. Each ant iteratively finds a solution to the problem (e.g. a shortest path in a graph) while laying pheromone trails (e.g. on the edges of the graphs). The pheromone has some evaporation rate to avoid being stuck in the local optimum. In the next iteration of the algorithm, the choice of the ants (e.g. which edge to traverse next) depends on the amount of pheromone trails deposited. There are many variations of

the algorithm, for example, including elite ants or multiple orthogonal colonies, but the idea remains the same.



Flower Pollination Algorithm / Honey Bee Algorithm

The inspiration is drawn from honey bees. Their colonies can span over vast amount of areas and yet they are extremely efficient in harvesting nectar or pollen from flowers. This is another prominent example of a population-based / swarm-intelligence method. It navigates within the search space of a problem in a similar fashion to bees. The population contains a number of scouts, which are constantly searching for new flower patches. These scouts move quasi-randomly in the surrounding area (the algorithm performs analogical local search here). The scouts return to hive and deposit the food found. If a food source was particularly rich (for the algorithm: a high quality solution was found) then the bees go to the area of the hive called "dance floor" and performed the so-called waggle dance. This dance informs other hive members about the location of the newly discovered food source and possibly more bees are recruited to investigate it. The colony is able to quickly switch attention on the most profitable food sources. Once the source has been exploited, it is abandoned and the search is focused on somewhere else. Many recruited bees quit the job of being scouts until a new exceptional discovery is made. The only condition for the application of the method is that some topological distance between the solutions can be defined.

Particle Swarm Optimization

Another population-based algorithm. Here, the population consists of particles. The method was first intended for simulating social behavior, e.g., of a flock of birds or school of fish. Like with ants and genes, each particle encodes a candidate solution. A particle has its position and velocity. The position is the actual encoding of the solution. The velocity is a change in a solution towards a different one.

During the process, particles move around the possible space. The velocity of each particle is affected by its previous velocity (physical inertia) and forces from attractors. Attractors can be: 1) a particle's best position, 2) position of the best neighbor and 3) global best position in the swarm. Naturally, a procedure which can evaluate a given position (solution) is required.

Simulated Annealing

Simulated Annealing is an optimization algorithm inspired by the annealing process in metallurgy. The process is based on rapid heating and then slow controlled cooling of a material to maintain the best crystal structure.

In computer science, the metaphor is as follows: first try to explore as much space as possible (search globally) and once you have some solutions, focus more on exploitation. The longer the algorithm runs, the less impactful changes it is allowed to make while doing the search.

Cuckoo Search

The inspiration is drawn from birds called Cuckoos and their brooding parasitism. Female cuckoos find external host nests to dump their eggs. They usually find such nests to match the color of their eggs with the existing ones. Birds often observe the candidate nests beforehand and choose the highest quality ones. In the CS algorithm, each egg in a nest corresponds to a candidate solution to the problem the algorithm tries to solve. There are multiple nests as well as multiple cuckoos laying eggs, so the algorithm can be classified as population-based or swarm-intelligence based. Each cuckoo lays one egg at a time in a randomly chosen nest. The nest's host can find out that a cuckoo egg has been with some probability. If it happens, it either gets rid of the egg or builds a new nest. The algorithm is equipped with mechanisms to maintain both exploitation and exploration of the search space.

Some other examples of nature-inspired algorithms:

- Firefly Algorithm – inspired by fireflies and their communication system based on flash signals.
- Krill Algorithm – one of the newer algorithms, inspired by the herding behavior of krill (small animals living in the oceans)
- Cellular Automata – a model of computation (simple computer), especially famous for the Game of Life which can be define using such an automaton. The game features simple rules when a cell is created and when it is destroyed.
- Artificial Immune Systems – a rule based algorithm inspired by the biological immune systems found in living organisms that protect them against diseases

Conclusion

Nature-inspired computing is a hot topic in computer science. It loosely connects together a remarkable number of areas including biology, genetics, social behavior, engineering, mathematics machine learning, artificial intelligence. If more romantic arguments do not speak to you, let's look it from a more formal point. The Earth is approximately 4.5 billion years old. Scientists believe that earliest appearance of life on Earth dates back to 4.28 billion years ago. The first animals are believed to appear around 600 million years ago. That is a lot of time to evolve to adapt to living on this planet. We happen to wander on the very same rock called planet Earth, so why not drink from this fountain of the ancient knowledge?