Biologically-Inspired Approaches and Methods

alexandru.ionascu96@e-uvt.ro

June 2019

Abstract

This report covers bio-inspired algorithms / methods, analysis techniques, with open discussion for further engineering of self-organizing systems. There are multiple bio-inspired topics brought into discussion. Mostly we will focus on: (1) swarm intelligence: social behaviours of insects / animals, (2) cellular computing: with a concrete example, and (3) evolutionary computation and modelling for optimization and design.

1 Introduction

1.1 Selected Titles and Authors

- *Bio-inspired multi-agent systems for reconfigurable manufacturing systems*, Paulo Leitãoab, José Barbosaacd, Damien Trentesauxcd in *Engineering Applications of Artificial Intelligence*, Volume 25, Issue 5, August 2012, Pages 934-944

1.2 Additional Resources

Apart from the mentioned paper, this report follows the main concepts presented in *Biologically-inspired Milti-agent systems*[1] course from Harvard University. Several chapters from the book *The Nature of Code*[2] were useful for the fifth section of this report.

1.3 Motivation

There are three main reasons that drove me towards this report title. Firstly, my skepticism towards the biological inspiration coming from artificial neural networks. I used to believe that artificial neurons (from a neural network) are way too simple to be compared with the natural neurons. I believed that the biological inspiration was just an unsuccessful result of a reverse-engineering process, while others just simply call it platonicity.

I still don't believe that scaling artificial neurons to the same size of the brain (approx 100 billions) would result in a similar intelligence compared to humans. However, there is a key factor that changed my mind gradually while reading

a blog post about addictive thinking and cognition. The neurons can detect very simple patterns. Activation process becomes instantaneous for the learned patterns. Now, I believe that the only unsuccessful inspiration from the context of using artificial neurons is that we don't have self-organizing topologies, which is probably the most important role of the brain.

Secondly, evolutionary algorithms. There are plenty of ranges of applications, but mostly I was fond on the genetic algorithms. I find them very interesting since they can be pretty much applicable to arbitrary functions (not necessarily continuous, not necessarily differentiable). I believe that creativity and common sense observations are very helpful for a better convergence of the population. Modelling the environment (population size, mutations and their volatility) and the rules (how to make crossover meaningful rather that just simple mixing or averaging) are crucial, however my favourite one is the ability to create your own fitness function. If the problem's statement is usually descriptive, one can find ways to choose the initial population very close to the the optimum and thus saving generations of computations.

Lastly, there is the idea of modelling complex problems with simple rules. Nature after millions of years can deliver plenty of such examples and I believe that the industry is at just the beginning of understanding that.

2 Biology Concepts

Biology taught us that there may exist a considerable amount of power in creating things out of large numbers of inexpensive and imprecise parts with ephemeral life: millions of cells which cooperate to assemble an organism, bees and ants in colonies to forage wide areas and build complex shelters without a predefined plan and without supervision, and also fish schools, bird flocks with their highly efficient migration and evasion techniques.

Not only we have so many examples of great designs in nature, but these are the results of millions of years of refinements. And this process is still improving, time is the only limit.

The main idea is that systems can reach a better collective intelligence from individuals with a relatively simple set of rules and with simple capabilities. No surprise, we have plenty of examples already in engineering for designing systems with the same properties, starting from the computer networks to multi-robot systems in applications from various domains (e.g. agriculture), to new computational structures like nano-structures which are also self-assembling. We have plenty of reasons to believe there are more to come.

3 Swarm Intelligence

3.1 Particle Swarm Optimization

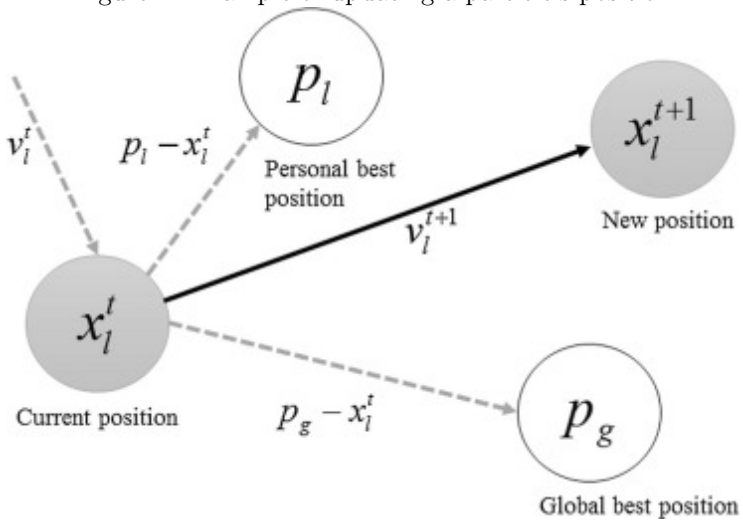
This optimization algorithm was first introduced in simulation social behaviour in the context of a bird flock or a fish school[5]. It is an iterative method where

we have a stopping criterion (an upper bound in the number of iterations / available time or a certain value found) at the beginning of the algorithm. We start with a population (swarm) of candidate solutions distributed randomly in the search-space. What we want is to move the particles meaningfully for our objective function.

In general we can distinguish three simple rules that should lead us (but no guaranteed) to an acceptable solution:

- **Direction of the current motion**: Yet again, initially random, the velocity increases if the solution gets better, otherwise the current motion slows down or even turns back in an opposite direction. In a social behavior, it represents the personality of the individual and his intuition.
- **Personal best solution**: Technically, each individual has it's own memory. In general we reference the best solution, while other variants focus on a collection of top-N solutions.
- **Global best of swarm**: The social pressure is being taken into account. Each individual takes into consideration the intelligence of the swarm as a total. The social pressure can be the same across the swarm, or distributed randomly for individuals.

Figure 1: Example of updating a particle's position

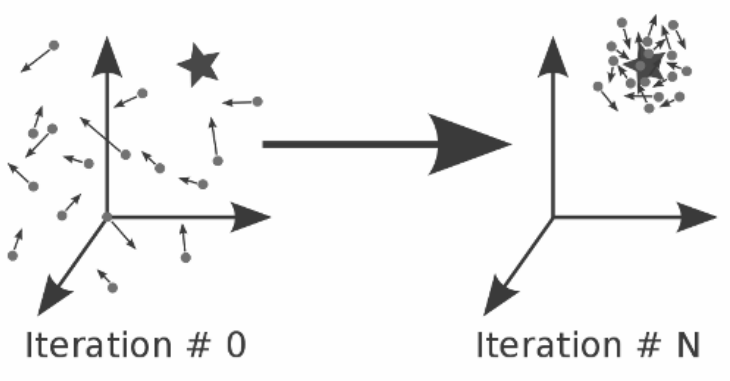


Basically, we have three main vectors, their weights and their linear addition gives us at each iteration the the new particle position. There are several topologies to improve the PSO algorithm. The biggest problem for any optimization method is to be stuck in a local optimum. Instead of using the global best, each individual would share his knowledge only in his neighbourhood. Another

one would imply that sharing information is within a subset of the population, simulating social circles, regardless of the geometric distance.

The rule of convergence can modeled also seen as the moment in time when all the individuals are reaching the same position, presumably the global optimum. Otherwise we may assume that no global optimum would be found.

Figure 2: Example of PSO convergence



3.2 Ant Colony Optimization

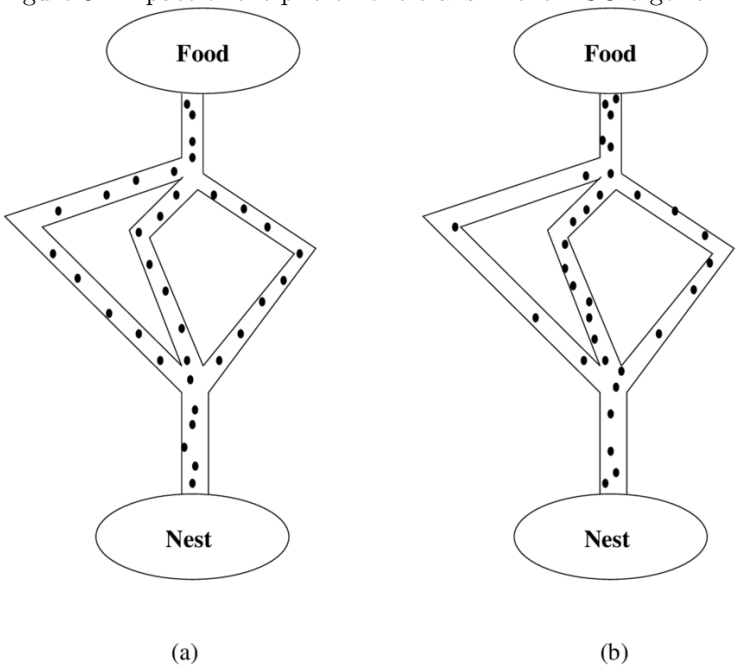
In nature, we have seen the way ants proceed for food finding. Ants of the most species initially start random walks and upon finding food, they return to their colony. Meanwhile they are laying down pheromone trails. The other ants may find the path, and will be more likely switch their random travelling to trail following, returning and also reinforcing it.

By the time, the pheromone trail starts slowly to evaporate. This process is proportional with the travel time for the whole path. A shorter one, in comparison, gets passed over for multiple times, and thus the density becomes higher the shorter ones. The evaporation implies the following advantage: the avoidance of converging to a local optimal solution.

Supposedly there were no evaporation at all, the first ants would make the first paths chosen to be excessively attractive to the next ones. Considering that, the exploration would be considerably constrained. Much of the research about the importance of the pheromone evaporation is yet still to be explored in the future, but the artificial systems have already benefited vastly.

The most important result is that if a single ant from the colony discovers a good path to a food source, then there is a much better chance for the rest of them to follow that path, and the feedback will benefit for many ants so everyone will be following a single path. The main idea of the ant colony algorithm is to implement this behavior with "simulated ants" which explore randomly the problem space[4].

Figure 3: Impact of the pheromone trails in the ACO algorithm



4 Evolutionary Algorithms

Evolutionary algorithms are practical applications of trying to implement evolutionary aspects similar to the ones found in nature. The major difference consists in defining the fitness of an agent. Nature most common fitness function consists in survival's capacity.

We can define evolutionary algorithm as a cooperative search method based on evolution as an optimization over a very complex landscape. Evolutionary algorithms are iterative methods that rely on evolution of a population over time. The representation of an individual is characterized by genes and fitness function. Over time, the individual suffer variation from a cross over function. At each of each generation only the fittest survive into the next one, through a process called selection.

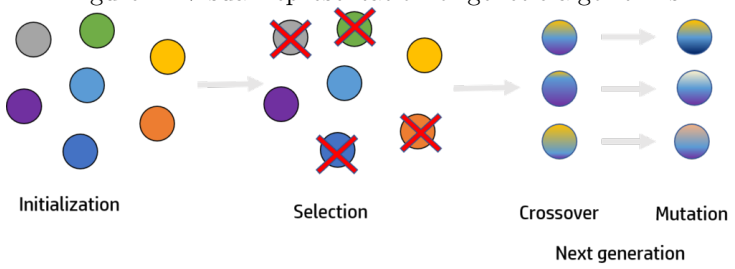
Specific classes of evolutionary algorithms:

- **Genetic algorithms**: are the most popular category of evolutionary algorithms. They imply recombination (cross-over) and each individual has it's own characteristic through individual mutation.
- **Genetic programming**: The solution of this method is actually a computer program that can have different structure at each generation.
- **Evolutionary Programming**: It is similar to genetic programming but

with fixed syntactic structure, however numerical parameters can evolve inside the generated program.

- **Differential evolution**: It is used for numerical optimization problems based on vector differences.
- **Neuroevolution**: It involves training and generating topologies for artificial neural networks with evolutionary algorithms.

Figure 4: Visual representation of genetic algorithms



5 Cellular Automatons

This section has the most interesting history. It all started with John von Neumann thinking very far into the future. He introduced the idea of the cellular automaton as a solution to mining operations at large scale. However, it was not for regular mining, it was indented for outer space mining where it would be almost impossible to build machinery and keep it's mechanical maintenance on asteroids or even distant moons.

Hence, a viable solution would be self-replicating robots or machines. They should work theoretically with no human intervention. In this way the cellular automata was defined for the first time, later out turning out to be used in multiple fields. In simple terms, there are simple three rules to define a cellular automata:

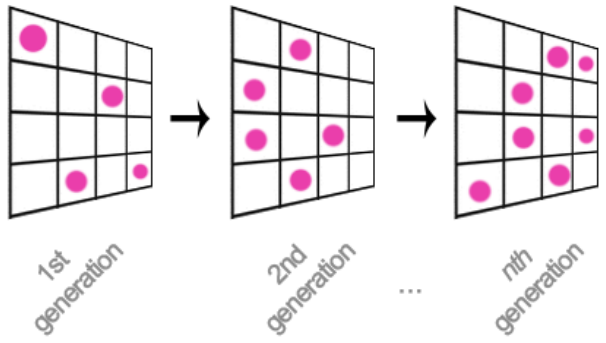
- There must be an elementary cell with a number of states. (e.g. ALIVE_STATE or DEAD_STATE)
- By applying a set of rules, the particular cell will evolve.
- There will be a next generation with the next state of each cell, thus the process can be applied again multiple times.

The complexity of the system can be extremely high, even though the cells and the rules are unsurprisingly simple.

In the most of the examples, we think of it as a grid (like a chess board). The simulation of the cellular automata can be made even manually, on the paper.

We initialize the grid with cells, dead or alive. We apply repeatedly the defined rules (usually we are looking to the neighbourhood of one cell at a time), and the memorize the next state for the grid of the second generation. This process replaces the current grid and can be repeated as many times. Generation after generation, we simulate growing, adapting, evolving from the previous one.

Figure 5: Example of a 2D cellular automata



Cellular automata have been successfully applied in random number generation, cryptography and computer vision.

The most famous examples are given by Stephen Wolfram with his 1D grid called *Rule 90* and John Conway with the famous *Game of Life* which is based on 2D grid.

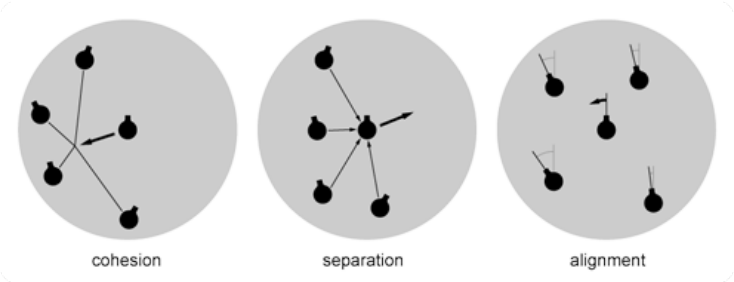
5.1 Boids

We will proceed with an example of 3D grid based algorithm. It is not actually a cellular automata in the strict sense but it works by the same principles. Back in the 1980s there was proposed the problem of simulating natural flock of birds or of fishes in the sea. We can state the following rules as:

- Cohesion: birds gravitate in the direction of another group of nearby birds.
- Separation: no two birds can be closer to a determined small distance.
- Alignment: the direction of the flight is consistent among neighbouring birds.

Note: if we consider each bird as an XYZ coordinate in 3D space, then this is the same approach as a regular automata as defined at the beginning of this section.

Figure 6: Bird flocking rules



6 Related Work

As mentioned in introduction, I believe there are ways to adept simplicity from plenty of rules that lead to complex natural systems. The importance of the bio-inspired methods seems to be very important in multiple domains, more recently in machine learning. Uber research provided studies to show simpler topologies in deep learning providing better results with genetic algorithms[3]. Transfer learning and reinforcement learning benefited from evolutionary algorithms, and the next directions are yet to come.

I prefer the bio-inspired models with reduced complexity, since my biggest fear is the lack of descriptive characteristics of the current state-of-the-art predictive models. We know they work, we have basic idea why, but I believe they don't provide much more insights.

The swarm intelligence algorithms have been successfully applied in a lot of practical problems. I've noticed good results on routing algorithms (graph-based) and there are multiple variants that lead to good results (even on NP-hard problems like TSP[6][7]) while maintaining the algorithm simple and intuitive.

Cellular automatons are expected to make a breakthrough in robotics. So far, cryptography benefited from it, probably more than any other field. They represent a very powerful concept, however we must reverse-engineer to obtain the desired rules, which is a much more complicated process.

7 Conclusions

During the writing of this report, I changed my views regarding this topic. My appreciation for complex systems that start from much simpler rules increased significantly. I am aware that interactions in nature are exponentially more

unpredictable than what we were able to compute so far. Nature is a great example, we may never be able to learn everything from it, nature is the oldest, the wisest entity yet, it is hard to ever beat it or tame it.

Even though bio-inspired methods seem to not contain any remote future perspectives, in time they developed other key concepts that were necessary to survive: robustness and anti-fragility to a wide range of external factors, even the ones which never existed before.

8 References

1. **Biologically-inspired Multi-agent systems**, Radhika Nagpa, *Fred Kavli*, **COMPSCI 289**, Harvard University

2. **The Nature of Code**, *Daniel Shiffman*

3. **Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning**, *Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, Jeff Clune*, Uber AI Labs

4. **Ant Colony Optimization**, *Marco Dorigo*, *Thomas Stützle* at MIT Press, 2004

5. **Particle Swarm Optimization**, *J. Kennedy, R Eberhart* Proceedings of IEEE International Conference on Neural Networks (1995)

6. **Ant Colony Optimization for the Traveling Salesman Problem Based on Ants with Memory**, *Bifan Li, Lipo Wang, Wu Song*, Fourth International Conference on Natural Computation (2008)

7. **Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm**, *Xuesong Yan, Can Zhang , Wenjing Luo, Wei Li, Wei Chen and Hanmin Liu*, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 2, November 2012