Self-learning Robots using Evolutionary and Genetic Algorithms

Henrik Johansson
Master in Robotics
Mälardalen University
Västerås, Sweden
hjn08005@student.mdh.se

Jonas Karlsson
Master in Robotics
Mälardalen University
Västerås, Sweden
jkn08008@student.mdh.se

ABSTRACT

Human-like robots capable of talking to us, walking amongst us and assisting with tasks that can be hazardous or dangerous to us, or simply just perform tedious work, has been the dream of many for a long time, and even more so when the field of computer science started to grow. Since then the computational power of hardware has increased drastically, and continues to do so, but our attempts at creating this human-made mind, this artificial intelligence, are progressing slowly.

So how should we go about reaching this sky-high goal? There are many different approaches to the problem and one of them thinks back to how evolution brought us from simple single cell organisms to what we are now. It should be possible to imitate, if not the whole, then at least a part of such a process in our computers today if we just define the algorithm properly. This is where the main subject of this paper enters; Evolutionary and Genetic algorithms. In this survey we have collected various interesting experiments and facts involving the use of these algorithms in an effort to interest the reader in a subject that has potential for ground breaking results.

Keywords
Evolutionary Algorithm, Genetic Algorithm, Algorithms, Neuroevolution, Co-evolution, Self-learning robots, Fitness, Pathfinding, The Red Queen Effect, 3D morphology

1. INTRODUCTION

The artificial intelligence, commonly referred to as an agent, is the mind of a robot and makes decisions based on sensor data taken from the environment. The agent carries out the decision by telling actuators (motors, servos etc.) what to do [11]. A basic way of programming a robot is by considering all possible problems and situations the robot can be confronted with and implement a solution to each one of the cases. There are several situations when this is a bad approach: You are confronted with a problem with hundreds or thousands of variables as for example in a vision recognition system. The knowledge within the area of a problem is incomplete as might be the case if you want to program a walk cycle for a robot with arbitrary amount of legs. The environment in which a robot must act is unknown and you must map the region and preferable optimize the the way the robot moves within that region. A learning system often responds much better to changes in the environment than a system that follows a set of rules.

The paper starts off by narrowing down from 2.Biologically Inspired Learning to focusing on 3.Genetic Algorithm and how it works. Then we see it used in several experiment in 4.Genetic Algorithm. The survey is summed up in 5.Summary and ended with 6.Conclusions.

2. BIOLOGICALLY INSPIRED LEARNING

As a method of computational learning, computer scientists have been inspired by living beings ability to absorb information to act upon. The nervous system is the core component of the most animals brains [3] and representing 82% of the human brain mass [5]. An average human male has 78 - 94,2 billion neurons cells [5]. By learning how single neurons work we can construct artificial neural networks to create a learning and adaptable intelligence. A real neuron has a great number of dendrites that receives electrochemical stimulations from other neurons. Electrical stimulations are then transmitted upstream to the neuron. A neuron has great adaptivity because repeated activity of the connectivity increases the size of the dendrite's connections. The equivalent in an artificial neuron is a real value weighted input. Depending on what you want from the artificial neuron the weight is updated accordingly. The weighted inputs are evaluated with a stepper function which calculates the neuron output. This artificial neuron is an analogue but simplified model of a real neuron and is often used for finding patterns in data.

Evolution is another powerful natural phenomena that have adapted low intelligence organisms into beings capable of abstract thinking. Consider an initial population and a desirable property, further entitled as fitness. A number of parents are selected with a high fitness level. These parents produces a number of offspring with a genetic code that is a random mixture of the parents’ plus an individual mutation. The children form a new population and the algorithm is repeated.
There are several different implementations of evolutionary learning in computer science. The algorithms used are chosen and modified to fit the application. Neuroevolution is an implementation where the evolutionary idea is applied to a artificial neural network. The weights represent the genetic code and is evolved according to a fitness function. A common use of this Evolutionary Algorithm (EA), in the field of robotics, is in motor control [2]. Another implementation of the EA is Genetic Algorithm (GA) where the genetic code is represented as a string of numbers which encode the behaviour of the robot at hand. The algorithm is often ended when the fitness-level of most recent population is acceptably high.

It is obvious that evolution has not only shaped our minds, but also our bodies. While most of the research in this field is done to evolve better intelligence, there is also research being done with something called 3D Morphology, which focuses on evolving the body structure. This is a branch of evolutionary robotics not as well established as the others since it is very hard to represent in real life. It is however fully possible to simulate in a 3D environment on a computer.

3. GENETIC ALGORITHM

The last section we mentioned the general idea of the GA, but how does it really work? Here we go a little more in depth into the main parts of the GA.

- Fitness
- Mating
- Mutation

3.1 Fitness

The fitness we briefly introduced earlier is an important part to decide the success of a species. It’s basically a sort of reward function and can be simple or advanced, depending on the goal of the species and the complexity of the environment. The function should contain the most important parameters such as distance for a path finding robot, or maybe time for a predator robot. Fitness is often the first part discussed in papers presenting experiments on evolutionary robotics. Some examples of such fitness functions are shown in figure 2.

\[
\Phi_{py} = \frac{1}{K} \sum_{k=1}^{K} \frac{x_k}{500}, \quad \Phi_{pr} = \frac{1}{K} \sum_{k=1}^{K} \left(1 - \frac{x_k}{500}\right)
\]

\[
EVAL_{i} = \frac{(1 \cdot \text{mediumActivation}) \cdot \text{maxStepsNo} - \text{stepLen} \cdot 2}{\text{maxStepsNo} - (1 + \text{distanceToTarget})}
\]

\[
f_1 = 1.0 + \frac{d_2 - d_1}{d_1 + d_2}
\]

\[
f_2 = 1.0 + \frac{d_3 - d_1}{d_1 + d_2}
\]

Figure 1: McCulloch-Pitts neuron architecture [4]

Figure 2: Three examples of fitness functions used in three different experiments [8] [10] [6]

fig 2a. "The fitness function \(\phi_i\) for species \(i\) was based only on the average time to contact over \(K\) tournaments" [6]

fig 2b. "The activation function chosen for this experiment depends on three parameters: the medium activation of the sensors ..., the number of steps until the target is reached ..., and the final distance to the target" [8]

fig 2c. "The creatures’ final distances to the cube are used to calculate their fitness scores. The shortest distance from any point on the surface of a creature’s parts to the center of the cube is used as its distance value. A creature gets a higher score by being closer to the cube, but also gets a higher score when its opponent is further away." [10]

All of the experiments that used these specific fitness functions will be presented later in the paper.

3.2 Mating

When all the members in the population have gone through simulation, the ones with the best fitness score are selected into a separate group where they then are picked randomly to create pairs\(^1\). These pairs are the parents that will create the offspring for the next generation (often replacing the ones with the lowest fitness scores), and is done by dividing each parents gene strings and copying and merging them into new gene strings.

Choosing which parts of the genome to copy is done with a genetic operator, for example crossover, which exists in different variants [1]. The standard crossover operator picks a random point in the gene strings (same location for both parents) and then interchanges everything behind that location between the parents to create two new strings, representing two children as figure 3a shows. Going one step further we can select two random points in the strings, and interchanging everything in between those location, as shown in

\(^1\)Operators using more than two parents exist, but are situational
such as GA instead of many non-biologically inspired ma-
plan has been achieved [14]. The advantage of using EA’s
some generations an more optimum or near optimum path
the parents’ genetic code with an additional mutation. After
als are paired. The pair produce offspring with a mixture of
distance to the target is rewarded, and the fittest individu-
the a path plan. A number of individuals are selected from
4. GENETIC ALGORITHM IN PRACTICE

3.3 Mutation
The last step of creating new offspring is the mutation, which
is an operator that mimics the natural process occurring
continuously in nature allowing species to branch out even
more than the mating procedure can. It is natures way of
test and experimenting with new things, but of course
at a greater risk since it’s not based on anything but ran-
doness. In GA, mutation can be done in a few different
forms depending on the type of data you are mutating. With
binary strings a random bit’s value can simply be inverted
to create a new string. When using integer or float values
however, this can lead to extreme changes if for example the
most significant bit is mutated. They are therefore typically
mutated by adding or subtracting a small value instead [9].

4. THE ENVIRONMENT INFLUENCE
The result of a robot programmed with some sort of EA is
not determined upon the completion of the program, but
rather when the robot is placed in a certain environment,
because that’s when the robot starts learning what it should
do. It is therefore of importance how that environment is
designed in order for the robot to show promising results.

3D morphology, that we earlier described, is a typical branch
of evolutionary robotics where the environment can play a
crucial role. Back in 1994 an experiment was conducted by
Karl Sims where robots were competed against each other
to gain and keep control of a cube in one-on-one matches,
simulated in a 3D world [10]. The fitness of the robots in
this experiments was calculated to reward robots that not
only stayed as close to the cube as possible, but also could
keep the opponent as far away from the cube as possible,
figure 2c). Selection, mating and mutation was then made
on both the body shape of the robot, as well as its control
system. Mating was done using both crossover and asexual
operators.

The environment in such a case is obviously very important
to determine the shape and behaviour of the robots since
they were all aiming to control the cube. What would for
example happen if there were two cubes? Maybe the cube
was put in a cage or on top of some sort of tall structure.The
experiment would have, with no doubt, had a different out-
come. More complex environments of course requires a well
specified body and controller design system in order to de-
velop the appropriate body structure.

4.2 MULTIPLE ALGORITHM SOLUTIONS
This part of the paper focuses on the combination of EA’s
with other algorithms, both of similar type (itself included),
and of non-evolutionary types. This can lead to advantages
over single algorithm solutions, but obviously also increases
the complexity of the system.

4.2.1 Combining with other evolutionary algorithms
When combining more than one EA in a closed environment
it is possible to simulate co-evolution. Co-evolution is an
interesting subject when working with EA’s, and the basic
idea is to use two or more systems - each controlled by an
EA - and let them affect each others development.

This could either be multiple systems cooperating within
the same robot, like the body and the mind as in the ear-
lier mentioned experiment by Karl Sims [10], or it could be
different robots with systems independent of each other, co-
operating or competing in the same environment. Where
the robot earlier had a static environment to adapt to, it
now has to adapt to another entity, which in its turn adapts
to the first entity and so on.

An example of the latter is predator-prey experiments where
the predator evolves to catch the prey which in its turn
evolves to avoid the predator. One of the earlier experiments
with predator-prey robots was made by D. Floreano, S. Nolfi
and F. Mondada [6]. In 1998 they used two Khepera robots,
one being the prey and the other being the predator (figure
4). Some differences were implemented to simulate the fact
that predator and prey rarely are of the same species. The
prey had double the predators maximum speed, but also a
protuberance on top of the robot, making it easier for the
 predator to detect. Moreover, the predator had a better
vision system with a 360° visual angle.

The fitness for the robots were simply based on the average
time it took for the predator to catch the prey; the shorter
the better for the predator and the longer the better for
the prey (figure 2a). First simulated in a computer for quicker
progress and enabling easier analysis of the results, the
robots were later placed in a 47x47cm flat environment
and allowed to move freely to catch or avoid their adversaries
in one-on-one tournaments.

The results showed a fitness graph with oscillating charac-
teristics. The predator and the prey took turns in having the
best fitness. In other words, for some amount of generations,
the predators were better at catching the prey, followed by
some generations where the prey was better at escaping the
predators. This happened in intervals of roughly 10-20
generations (Fig 7 in [6]). Even simulation for 500 genera-
tions could not determine a winner of either the prey or the
 predator.

The phenomena is called The Red Queen Effect and was
proposed by Leigh Van Valen in 1973 and is based on the
idea that every positive effect for one species (increase in
fitness) has an equally negative effect distributed over all the
other species in the area, who then must continue to evolve
just in order to survive [12]. It is often mentioned when
doing work in the field of co-evolution, and traces of it can
interestingly enough be found in Karl Sims 3D morphology
experiment as well, though he himself does not mention it
in his paper (Fig 8 in [10]). Attempts have even been made
to see similarities in business models [13].

4.2.2 Combining with other non-evolutionary algo-
rithms

In an article about path planning for mobile robots by S.
Chin Yun, V. Ganapathy and L.Ooi Chong [14], an im-
proved GA is proposed. Their proposal uses two separate
algorithms in combination with GA: Obstacle Avoidance Al-
gorithm and Distinguish Algorithm. The Distinguish Algo-
rithm checks so that the proposed path planned by the GA
and the Obstacle Avoidance Algorithm is feasible. In this
way the GA consider all feasible paths and optimize the fi-
nal path. Simulations and real-time implementations with
an Team AmigoBot™ robot showed good results.

GA is often used in optimization problems where the map-
ing of the environment is done by other algorithms. In a
GA is applied to the ant colony optimization (ACO) algo-
rithm.

The ACO algorithm is a sort of swarm behaviour where two
kinds of agents exists: explorers and workers. An agent is
considered a worker when it returns to the base with food
and considered a explorer if it still is in search for food.
In nature, ants use pheromones to tell other ants where
they’ve been. Ants follow the trail with the highest amount
of pheromones. In their implementation the agent is likely
to use an artificial pheromone intensified path generated by
an earlier agent that have completed a route, but is not
compelled to.

The GA is used in this example to make a natural selection
of the agents based on how short the path it took was. This
selection generates offspring that compose a new stronger
and more optimized generation. The GA also contributes
with a dynamic path optimization for changing environment
due to evolution’s ability to adapt.

published an article describing an experiment to compare
two sets of neural networks on a mobile robot in order to
see which one performed best in path finding. One neuro-
evolutionary network that adapted according to the EA
once (when conceived) for every generation. The other net-
work however used not only a similar network for motor con-
 trol but also a reinforcing network connected to sensors that
gathered data about the direction to the target, distance to
the target and if the robot hit a wall. This reinforcement
network used the sensor data to generate correctional values
that was applied to the motor control network. The EA was
used to evolve the characteristics of the reinforcement net-
work once for each individual meanwhile the motor control
network self-adapted due to the feedback during the lifespan.
figure 5 The results showed that the agent with the simple
neuroevolution algorithm found the target early but had a
hard time to optimize the path. The reinforced neuroevo-
 lution algorithm however struggled for a greater number of

Figure 4: The Khepera robots used in the predator-
prey experiment
Figure 5: Structure of reinforced neural network

generations to find the target but when it did it fast found a near optimum path. The reinforced neuroevolution agent was also much more adaptable to where in the environment it was initially placed.

5. SUMMARY

Evolutionary and GA’s were created inspired by nature and the biological world. It is fascinating how creatures can adapt so well to new environments and is a trait that would be useful to have in robotics. We have gone behind the scenes of how the GA operates with fitness, mating and mutation.

Experiments have been presented showing various interesting results such as robots designing their own shape, the Red Queen effect and ant-mimicking path optimization to name a few. We have discussed how EA’s can be used, alone or combined with other algorithms - both of the same type and completely different types - to create more intriguing results. Results that solves problems in a way we unlikely would have solved them ourselves, but appear to work for the challenge the robot is set up against.

6. CONCLUSIONS

EA’s such as neuroevolution and GA can, when implemented in a smart way, provide a optimal or near optimal behaviour and is robust to changes in the environment. Since evolution only occurs once for an individual every generation time is a limiting factor which has made simulations an important part of the development of the the research field of artificial intelligence in robotics. There is however research using the slow robustness of evolution and the method of reinforcement learning that provides a learning system with correctional learning during the individuals life span.

While the GA in itself can evolve interesting results, it’s when co-evolved with other GA’s a most fascinating behaviour can be produced. Even when just two robots were placed in a simple environment like the predator-prey experiments showed, lifelike behaviour could immediately be observed in the form of the Red Queen effect. When co-evolution was applied in both a cooperative and a competitive manner in the 3D morphology experiment, the lifelike behaviour was even more apparent.

What this means for the future is hard to say. Evolutionary robotics is still a young field of research with high ambitions and much to show.

7. REFERENCES


