A GENETIC ALGORITHM LEARNS TO PLAY TIC-TAC-TOE

José Rodríguez
jose@ejristos.com
August 2020, Bogotá, Colombia.

Abstract

A *Genetic Algorithm (GA)* from *Artificial Intelligence (AI)* is implemented in *JavaScript (JS)* to test the idea that Evolution can learn to play *Tic-tac-toe*. A generous context with suitable programs is provided. We heavily rely on the Logic of Science: if a hypothesis leads to a mandatory prediction that results to be false, the hypothesis has been falsified and shall be abandoned.

Table of Contents

INTRODUCTION....2
THE RULES OF T3....2
HUMAN PROGRAMMED AI....2
EVOLUTION AS A CONSTRUCTOR OF INTELLIGENCE....3
LET US TEST EVOLVED AGENTS....8
TESTING THE EVOLUTIONARY THEORY....9
TESTING PHILOSOPHY....12
CONCLUSION....17
LIST OF PROGRAMS....18
TO KNOW MORE....19
....19

INTRODUCTION

From the stand of engineering, the purpose of Artificial Intelligence is to imitate the behavior of intelligent beings. From the stand of Science, the purpose of AI is to test ideas about the material basis of human actions, reactions, and thoughts. Since the human brain is made of neurons, neuronal networks are usually considered to be the basis of both human and artificial intelligence while their all-pervading success reinforces the idea that we think with our brain. This is false: death and its ensuing halting of useful brain activity show in the light of anthropological witness that we think with the spirit and that the body is its avatar. Nevertheless, our neurons enable automatic reactions that in general are very elaborated. But on the other hand, human beings react partly by instinct and partly by learned routines. Learning usually happens along an evolutionary process. This is a clear sign that Evolution shall be useful to solve problems whose solution might seem to be specific to human beings. So, Evolution is dared here to become a master in the *Tic-tac-toe* game also known as *T3* or T*hree-in-line*. Our discussions are supported by pertinent JS programs that run on most web browsers.

THE RULES OF T3

Program *T3WebDevSimplified2.html* allows you to play Three-in-line in solitary to learn the rules and get expertise. The program calculates scores and outputs the winner.

Exercise. Use the program *T3WebDevSimplified2.html* to review the rules of *T3* and to verify that you gain expertise through an evolutionary process: bit by bit, trying this and that and selecting the best.

HUMAN PROGRAMMED AI

A principal component of human intelligence is the capability to imagine, simulate, and judge situations that have not occurred as yet and that are relevant to the problem to be solved.

Astonishingly, program *T3MinimaxP5ToJS.html* shows us how AI can carry this to an unbeatable extreme at least with this game. Basically, you imagine all possible moves up to the end of the match beginning from your present state, compare final scores of all matches, and choose the best move. This is done automatically with the Minimax Theorem of Game Theory for games of zero-sum for two players, in which what one wins is lost by the other. This program offers you the opportunity to play here against an AI master. It is a JS adaptation of the original version *T3P5.html* that can be run with an internet connection because this program is not written in JS but in P5, a JS dialect of the Java superfamily, and so a translator to JS is needed that is downloaded from the Internet.

Exercise. Play with program *T3MinimaxP5ToJS.html* and test the following assertion that we accept: when playing against a master, one can never win. The most one can get is a tie. Intrigue: Does this depend on who is the first to play?

Exercise. Program *T3MinimaxP5ToJS.html is* in its present state very boring because to initiate another match one must reload the program. So, study program *T3WebDevSimplified2.html* above *or T3Cornellv3.html* below to see how one can very nicely program restarting a game and add this capability to *T3MinimaxP5ToJS.html*. Our answer will be included in the program *T3MinimaxGraded.html* below.

EVOLUTION AS A CONSTRUCTOR OF INTELLIGENCE

Human programmed AI is powerful as it is shown by the program *T3MinimaxP5ToJS.html*. Its great problem is that its existence is human dependent. Looking for independence, Evolution might be a partial solution. It is partial because it takes a lot of intelligence to discover that evolution, something that we practice by instinct, is important, and more intelligence and much expertise is necessary to program and connect it with a specific problem. How to do this regarding T3 was shown in 2009 by Nathan and Sajeev, who developed a program in C, *T3Cornell.c.* Some important claims that we extract from their discussion are the following:

- Evolution works.
- Evolution is not magical and so the mandatory evolution of perfection goes through a lengthy path whose possible

convergence to perfection is hidden behind unbearable computer time.

- Recombination is not included because it is expected to be irrelevant for the present situation because of the high degree of cloning and the low mutation rate that was used.

We can turn now to gain an understanding of the code:

Which is the main idea to convert Evolution into a gamer? We must pose to Evolution an optimization problem that encodes the commitment of getting a winner in the chosen game. But this is clear since in games one struggles for winning over opponents. So, fitness is defined as the number of victories in a tournament.

How are strategies and moves encoded that they could be handled by evolution, mutated, and recombined? The answer is this: a move is an assignation of a cell to a particular state of the board. So, one encodes each state of the board and assigns to it a position in the board wherein our mark will be posted. Now, board states are naturally encoded as numbers in base 3.

Example. Let us consider the following board state:

```
o – o
- x -
x o x
```

and the x-gamer must play. If his or her move is marking the second line, third column, the new board state will be

```
o - o
- x x
x o x
```

Now, instead of looking at the board as a 2D grid, one can look at it as a 1D row and begin counting from 0 to encode the proposed move:

o-o-x-xox→ move = mark the 5h site with x→ o-o-xxxox

Next, instead of letters, one can use numbers: blanks or hyphens are encoded by zeroes, the o’s are encoded by 1’s and the x’s are encoded by 2’s. With this interpretation, we can reread that move as follows:

101020212 → mark the 5th site with 2→ 101022212

So, the move can be encoded as
{5, 2}(101020212) = 101022212

where {5, 2} represents an operator that at the fifth slot puts 2. As we see, we have needed three complex elements to encode a move. This encoding is inefficient. To improve it, we can take into account that each state of the board is represented by a number in base 3 with 9 ciphers, although some numbers cannot represent a realizable state. For instance, the number 102012222 cannot represent a state for there are too many 2’s with respect to 1’s. Now, we define a strategy as a function that assigns a move to every board state. Strategies are efficiently encoded as follows:

Since there are 3^9 = 19683 possible numbers of 9 ciphers in base 3, we declare an array with 19683 entries and at each entry, we pose a number from 0 to 8. The entries of our array are numbered in base 10 while board states are numbers in base 3. That is why translation methods to change basis are implemented.

Now, how is the GA implemented? The GA is just an evolutionary environment in which arrays, strategies, or chromosomes struggle for the reproduction of the fittest. Each array or chromosome is given as score the number of times it wins in a tournament. That score defines proportionally the probability of reproduction.

Our transpiling of *T3Cornell.c* to JS can be seen in program *T3Cornellv3.html*.

Exercise. Run program *T3Cornellv3.html* and verify else reject our conclusion: after 10000 generations of evolution over a population of 100 individuals no improvement can be registered regarding the maximal score per generation.

The immediate explanation of our failure with *T3Cornellv3.html* might be simple: our transpiling is defective because the original code was intended to run in the company of an external hard human-machine interface that we ignored. So, we decided to make in JS our implementation of the original idea: it is the program *T3Cornellv4.html.* To detect whether or not Evolution can improve the capability of its agents, chromosomes that mimic an intelligent behavior, we measured the maximal number of victories per generation, the number of ties and victories, the number of double marks, forbidden moves, and the average number of moves per match.

Exercise. Run program *T3Cornellv4.html* and verify else reject our conclusion: after 10000 generations of evolution over a population of 100 individuals no improvement can be registered concerning the maximal score per generation.

If Evolution does not function, it might be badly implemented. So, our program *T3Cornellv4.html* might be defective. Hence, some tests are in order. To run a test, make a copy of the program and run your test over it. This is a permanent rule. The natural test is always to see what the program does, a task that moreover helps to understand the code. So, we command it to write specific information.

Exercise. Make a copy of the program *T3Cornellv4.html*. Set the printing flag REPORT_MATCH = TRUE. In that way, you can see how a match between two chromosomes is run. Verify else reject our claim: the match section is well devised and functions OK. This is a test for the method assessFitnessPop() because victories in a tournament define the fitness of each gamer.

Exercise. Use the various printing flags in program *T3Cornellv4.html* to run many diverse tests to verify our claim that the program is well designed and perfectly implemented. To test mutation, pose a high rate, say, 0.7. To test recombination, pose a high rate and shorten the chromosomes to, say, 20 chars. It might be good to mute the function *assessFitnessPop()*. This is done in JS by prefixing *//* to its call : *//*assessFitnessPop().

If Evolution does not function but we cannot prove that our software is defective, the possibility exists that we are measuring properties that remain almost constant while others do vary as a result of evolution. In this regard, we have not implemented the measuring of another descriptor to detect the effect of Evolution that was used by the Authors of the original code: the distribution of fitness. They claim that Evolution causes it to shift to a higher value. Their claim can be tested with program *T3Cornellv5.html*.

Exercise. Run program *T3Cornellv5.html* and test our claim that the effect of Evolution over the distribution of fitness is almost nil.

Our recurrent failure forces us to test the program as a whole.

Exercise. Connect Evolution to an easy problem, one that is already known to be rapidly solved by Evolution and verify that it works. Our answer is in program *T3Cornellv6.html*. After successfully solving this exercise, we gain confidence and felt invulnerable against doubts: our program is correct.

Comment and challenge. There is no improvement according to various descriptors and parameters. Does that imply that Evolution does not function? That may be the case but the contrary event would be more interesting: players continuously improve their performance but, say, the general ranking is static. This effect is known as the Red Queen Effect: in Alice's world, one runs and runs (evolves and improves performance) to remain in the same place (say, to remain in the third rank forever) unless a game-changer appears. Challenge: In the sequel, decide whether we have an instance of the Red Queen Effect or simply a very difficult problem that Evolution cannot attack. Or maybe something else?

When Evolution is well programmed but does not function, we face a problem with complexity. For if a problem would be easy, its associated landscape would have one or some few peaks and Evolution would rapidly climb to the summit. But when the density of peaks are too many, a long step in the wrong direction could lead to another peak and so on and on. Now, the magnitude of steps is regulated by the mutation and recombination rates. When, say, the mutation rate per char per generation is low, mutants slightly differ from their parents and so Evolution goes on tiny steps. But when mutation rate is high, mutants are totally different than parents and we say that Evolution goes on large steps. This last possibility is welcome for our case because descriptors tend to erratically move around certain average values. So, let us low the mutation rate so that steps could be very small to make fine-tuning possible, and climbing on high steep peaks realizable.

Exercise. Verify else reject the thesis that the associated landscape to T3 is very roughly and hence the mutation rate shall be adjusted to low values to allow for the exploration of promising peaks. In this regard, check our estimation of the performance of the mutation rate (per char per generation). Our solution is in program *T3Cornellv7.html.* According to our observations and when Population size = 100 and Recombination rate = 0.5, the values of mutation rate rank from useless to very-effective as follows: 0.5 < 0.2 < 0.1 < 0.05. < 0.02 < 0.01. Evolution produces great effects for the last three values of the

mutation rate. The program was run with a population size of 100 and a high recombination rate. That is why we keep from now on the mutation rate constant with a value of 0.01.

Challenge. The original Authors did not use recombination because of the extreme degree of cloning and the low mutation rate that they used. Is recombination pretty useless for our setting?

Challenge. Optimize the mutation and recombination rates at the same time. What about dynamic rates?

Challenge. Prove else disprove: Randomness can create agents with not too low dexterity that serve as raw material for evolution.

Challenge. Probe or rebut: Evolution is unable to produce champions that beat everyone.

Challenge. Prove else refute: one cannot get a better player thanks to tiny changes. Instead, one must change various items that coherently work. Hence, Evolution is not observed to be gradual but by jumps and since this happens rarely, the selection procedure tends to clone the very same champions and therefore populations tend to be highly homogeneous.

LET US TEST EVOLVED AGENTS

We have witnessed that Evolution functions and that it is useful to create AI agents from nothing, from randomness. Nevertheless, a problem remains: what are those agent worth? The question is important because although Evolution increases the capability to beat more and more opponents, one does not know what will happen to champions outside their clique. In this regard, we need to know by direct experience how well the best evolved AI agent plays. But we have a question about who is the best:

Intrigue: Given that the maximal score per generation oscillates, a question arises: who is better? The best of the last generation, or that with the best score through all the evolutionary process? Do we have an answer to this question if we clone the best of each generation to fully initialize the next generation?

Exercise. Add to program *T3Cornellv7.html* the possibility to get a match of the best AI agent of the last generation against a human being. Our answer extracts the best exponent of the last generation in program *T3Cornellv8.html*, which in program *T3Cornellv9.html* gets ready to fight against a human player. Verify else reject the expectation: the degree of dexterity of the evolved agent increases with the number of generations. Compare your results with ours given that we run our simulations over 100 individuals:

- Double marking was the general rule for 10, 1000, and 10000 generations.
- In all cases, the human player could win along any desired line but in some cases it was difficult and while we were searching for that, the Agent won us various times.
- Some first signs of intelligence appear: the Agent sometimes completes the lacking mark to get three in line, other times it interdicts three in line of the opponent. But this is darkened by the lack of intelligence to always pass from two-in-line to the required three. Anyway, the most salient feature of the very low intelligence of the evolved Agent is that it is stiff, rigid and does not learn from errors. It seems that Darwinian Evolution as implemented by the present program to be an opponent worth of respect shall achieve absolute perfection before engaging in championships.

Our partial conclusion is this: Evolution readily evolves an Agent that plays the game T3 as a beginner. Nevertheless, to gauge the effect of the number of generations over the developed skills was impossible for us with this program because once the mutation rate has been optimized we would say that Evolution does not function. That is why we need a more appropriate setting.

TESTING THE EVOLUTIONARY THEORY

Life is astonishing because of its all-pervading perfection. The Evolutionary Theory says that perfection can be reached beginning with mediocrity, advancing to good quality, raising to superior performance to escalate to excellence. Hence, the Evolutionary Theory predicts that the Evolution of Perfection is mandatory, i.e., the fossil record shall be filled in imperfect specimens. Nothing as this is observed. Instead, what is observed in nature is the impressive perfection of the coronavirus. So, the Evolutionary Theory is obviously false.

Now, one can get skilled in T3 after a not very long experience, so we expect that the agents synthesized by Evolution also get skilled with more and more generations in such a way that the Mandatory Evolution of Perfection is also observed. That is why we dare ourselves with the following goal which is also useful to gauge the cost of perfection which is measured in terms of the number of generations (given that we keep the population size constant).

Project. Devise an artificial master in *Three-in-line* that can adjust its cleverness to given levels. In that way, we can challenge Evolution with increasing complexity to see as expected that Evolution works and that perfection evolves, i.e., increases from nearly null to excellence.

Our solution goes as follows:

The first step of our project is to devise the graded AI master. An idea is this: the Minimax Algorithm has various loops so, the simplest solution is to truncate them. Nevertheless, we chose to adhere to the tradition in games of strategy which mimics the behavior of masters: they simulate in their head the development of the game until the final output beginning from any possible configuration to next take up the best move. Apprentices cannot do that or do it but very partially. A good indicator of the dexterity of an apprentice is the depth of his or her simulations. In our case, the apprentice scans in the first place a cell to see whether or not it is empty to next simulate the further development of the game up to one, two, …, n moves which measure the depth of the simulation. This setting resolves immediately how to devise a graded master according to the game tradition: one must truncate the depth of the simulation. It seems to range from 0 to 8, both included, so we have 9 possible levels of dexterity for the player that plays first. The difficulty of levels might be not homogeneous because different branches along the exploratory tree could have different depths.

Our implementation of a graded master is found in program *T3minimaxGraded.html.* The Minimax Theorem for Games implements what a real master does in the head so, we use the parameter *level* to regulate the depth of the simulations.

Exercise. Play with the graded master of the program *T3minimaxGraded.html* and check whether or not it is correct, i.e., whether or not increasing the level of the game

translates itself into a growing difficulty of winning or getting a tie.

Challenge. Evolve program *T3minimaxGraded.html* to get an irresistible attraction for small children.

Now that we have a graded AI master, we can invite it to the ring to bold Evolution to establish how costly is for it to develop winning skills. This is done in program *T3EvoOfDexterity.html*.

Exercise. Play with program *T3EvoOfDexterity.html* and verify else refute: evolution being the first gamer to play can reach level 6 of 7 in less than 30000 generations with a population size of 100. It could be as low as 6500 generations. But the double is necessary to advance to level 7 which means perfection (for the goal of neutralizing a master).

Exercise and challenge. Use program *T3EvoOfDexterity.html* to verify or reject: the Evolutionary Process of Evolution vs. T3 conforms to the footprint of medium complexity, which is:

- Slightly useful functions appear by mere randomness,
- which are quickly evolved to mediocrity (the half of maximum attainable function),
- which are then slowly evolved to good quality,
- which are then and then more slowly evolved to superior quality
- and then extremely slowly evolved into excellence.

Clarification. When complexity is low, the evolutionary process flies upwards nonstop from almost nothing to the summit. When the complexity is high, evolution is worst than randomness.

Challenge. Resolve the following objection: the alleged footprint of evolution that conforms to the evolutionary solution of medium complexity is fake. Instead, it is just the footprint of the program that proposes increasing levels of complexity to the gamer. We find the objection valid but we will assume that it is favorably solvable.

Exercise. Adapt program *T3EvoOfDexterity.html* that it could be run on cell phones and computers with small memory. Hint: The regulable part of memory depends on the population size. So, enable the User to choose its value. With our desktop, population

sizes higher than 200 might cause the System to collapse or clog. Our answer is in program *T3EvoOfDexterity2.html*.

Challenge. In our case, Evolution always plays in the first place and the Master in the second, hence the maximum level is 7. Our criterion for level promotion was to win or to get a tie. But for level 7, when competing with a master, one cannot win but only to lose or to get a tie indifferently of playing in first or second place. Nevertheless, Evolution almost always was the winner up to level 6. Prove that this is not a coincidence and explain why.

Challenge. Dexterity grows in jumps. Is it like a cup that is filled drop by drop until it overflows? Or, rather it is like making fire with wooden sticks, such that flames are not produced by intense blowing but by achieving good rearrangements of the wooden sticks?

TESTING PHILOSOPHY

The Evolutionary Theory is for most scientists the summit of science and the organizer of knowledge as a whole. So, it has grown into Evolutionism, a world view that competes with religion. This is the philosophical school of thought that teaches that “Evolution is open-ended, i.e., no one knows what will happen next, and therefore it has no limits to its creativity. Hence, Evolution cannot be proven to be false. And being the simplest natural explanation of the existence of everything in the living kingdom, it is the truth, the correct explication of our existence”.

Now, what is exactly the Open-Ended Evolution of Evolutionism? From the stand of simulations, it is Evolution in the long term, where super-computing is welcome. But in the sociology of Science, Open-Ended Evolution is a big show-case where you can exhibit the wonders of Evolution and hide at the rear your lack of courage to recognize that the Evolutionary Theory is obviously false because the complexity of life demands from Evolution to leave all-pervading tracks of the Evolution of Perfection, which are not found neither in your body nor in the fossil record.

Exercise, Challenge, and Warning. Rewrite program *T3EvoOfDexterity2.html* that it could serve as an example of Open-Ended Evolution, i.e., if you get out of patience, you must be

able to stop execution and see printed results. Our answer is in *T3EvoOfDexterity3.html.* This program implements a clock-based task manager: for each beat of a clock, you must do a certain task. Beat after beat, a loop results. If the stop-task button is clicked, the loop ends. Challenge: Optimize the duration of the clock beat for this program on your specific processing unit. Warning: If the duration of the beat of the clock is too short, calls will pile up and everything will mesh around although this might be difficult to notice and wrong conclusions would emerge unnoticed.

Exercise. Run program *T3EvoOfDexterity3.html.* What happens once level 7 has been achieved? How do you explain that? This is our answer:

After arriving at level 7, the level counter rises to more and more levels with a separation of more than 3000 generations within consecutive levels. What is the meaning of this given that levels past 8 are not related to the skills of the graded master? This means that our GA can synthesize a strategy that can compete with our master but that does not know how to win under general circumstances. Were that the case, Evolution would always get the maximum, a tie for our case, generation after generation. But this does not happen. All this sounds like a contradiction:

Exercise. Explain how a half-cooked Evolution can compete with a master. Our answer is to emphasize that our master has a problem: it as stiff as a steel rail because it always assumes that its opponent is a real maximizer under all circumstances. So, the behavior of the master is not that complex, and Evolution can find a strategy to neutralize it. That strategy determines the values of 5 entries in the corresponding chromosome but it might lack the correct specification of hundreds of other moves. Our evolved agent is exactly an idiot-savant that only knows its specialty, to neutralize a master, but it can fail in other situations. Say, if the agent already has two-in-line plus the opportunity to get three-in-line in the next move, the agent does not fill the appointed cell but can commit double marking somewhere else. Anyway, the work done by Evolution is not easy because it is improbable to find the 5 correct entries among nearly 20000 that must lead to a winner.

Exercise. Our discussion implies that to learn to play and to neutralize a master seems to be quite different goals, where the second is too easy as compared with the first, which belongs in

the realm of Open-Ended Evolution. Devise and run the machinery to directly test this.

Our answer:

Our machinery is in program *T3EvoOfDexterity4.html.* The idea that guided us was the following: The Master of the Minimax Theorem certainly knows how to do the best for any possible board configuration. So, we use it to calculate a Master strategy, a chromosome with the information on how to win or at least on how to lose the least. We guess that Evolution tends towards this chromosome but not perfectly because some few configurations might have various winning moves. That is why we measured the distance between best agents and the master chromosome with the expectation that it must decrease with time. The following was the experimental result: the distance does not converge to near zero. Instead, the distance from the best chromosome to the Master chromosome oscillates near 2700 without a tendency during nearly 80000 generations when the level marker reads 10.

Our failure leads us to suspect that Evolution cannot synthesize a true master. A direct test was in order: we examined the best agent after 100000 generations with program *T3Cornellv9.html*. We found that double marking is still persistent, that almost every modality of three in line can be gotten and that signs of intelligence also are present shoulder to shoulder with signs of stupidity. Our verdict is: there is no difference in the capability to play T3 between the best Agent of generation 1000 and that of generation 100000.

Our negative impression of the positive effect of Evolution over evolved AI agents impulse us to test the idea that Evolution might be worse than randomness:

Exercise. Program a match of randomness against the graded master of program *T3EvoOfDexterity4.html.* Decide whether or not Evolution is worse than randomness to learn to neutralize a graded master. Our solution is in program *T3EvoOfDexterity5.html.* Our result: Randomness and Evolution are indistinguishable.

Exercise. Explain why Evolution is inoperative to synthesize good players. Our answer: Evolution is good when quality can be gathered bit by bit, by small change, a condition that requires that variants could be continuously graded according to a useful descriptor, one with evolutionary connotation. So, if one variant

is good, a tiny change could improve quality. But when quality varies erratically in the vicinity, Evolution is inoperative. This is the case in this game, for if a move to cell (2,1) leads to a winner, nothing concretely can be predicted concerning surrounding cells. That depends on the concrete situation that is defined by the whole board. So, to be good or bad is an ensemble property and hence there is no individuality to be selected for. In short: Evolution is predicted to be inoperative.

Challenge: Relate the present encoding to synthesize agents to play Three-in-line with the impossibility of Evolution to disentangle a thread, a problem that was studied somewhere else.

Caution: Randomness is worth the most high respect.You can study a match of Randomness against the Master and you will highly praise the intelligence of Randomness. As in the following one that promoted Randomness to level 8:

```
Level = 7, GENERATION = 68600, Master : X, Randomness : O
 O: Board state 0 -> Mark position 2 -> cell = (2 , 0)
 X: Cell: (1,1)
 O: Board state 891 -> Mark position 8 -> cell = (2 , 2)
 X: Cell: (2,1)
 O: Board state 11826 -> Mark position 3 -> cell = (0 , 1)
 X: Cell: (0,0)
 O: Board state 11831 -> Mark position 7 -> cell = (1 , 2)
 X: Cell: (0,2)
 O: Board state 12092 -> Mark position 1 -> cell = (1 , 0)

 X | O | X |
 O | X | O |
 O | X | O |
```

Exercise. Resolve the following mystery: We witnessed a positive performance of Evolution in the fine-tuning procedure of the mutation rate. So, How can Evolution be indistinguishable from randomness? Our Answer: Evolution is as good as Randomness but only when it is tuned to its maximal expression but in other settings, it performs worse than randomness so, it can evolve for the better.

Challenge. Test our explanation of the aforementioned mystery.

Challenge. Prove that for the game Four-in-line, that you must invent, Evolution is definitely worse than Randomness. Clearly explain why and test your explanation.

Exercise. Propose an exit to the following terrible problem: Science has been consistently preaching that the gap between inanimate matter and the man regarding the computer power of the human brain was filled by Evolution for which there is plenty of time, nearly 20 billion years. In the light of the present study, this seems ridiculous: Evolution might tirelessly work during an eternity to just produce mediocrity. This is of course a subjective, hurried generalization of the Author. So, which is the path of research to convert these subjective, hurried verdicts into objective and authoritative discussions that with pleasure will be accepted by all parties? Our answer: our study shows one simple thing: everything that you say about Evolution must be proved and tested, else you are deluding yourself and everybody.

CONCLUSION

We implemented in JS a GA of the literature to synthesize agents to play the game Three-in-line. We proved that Evolution can find a strategy to neutralize a master in that game. Thanks to a graded master, we also verified indirectly a prediction of the Evolutionary Theory for problems of medium complexity: The Evolution of Perfection is mandatory beginning from mediocrity, advancing to good quality, raising to superior performance, to finally escalate to excellence. Or, the fossil record shall be filled in imperfections else the Evolutionary Theory is false. Now, to neutralize a master amounts to a specification of 5 entries in a vector with almost 20000 and to fill in them with the appropriate moves. So, this achievement has nothing to do with playing good, a capability that needs the specification of various hundreds of suitable entries. This task was expected to be done by Evolution bit by bit during a long time, a realm that is known as Open-Ended Evolution. But when we tested this illusion, we found that Evolution is inoperative. This means that there is no difference between the capability of the best agent of generation 1000, which was rather low, and that of generation 100000. Or, randomness also neutralizes the master and it has the same footprint as Evolution when passing the levels posed by the graded master and when going into the realm of Open-Ended Evolution. That is why we conclude that mediocrity is the objective hallmark of Open-Ended Evolution. This immediately leads to a falsification of the Evolutionary Theory that can be rejected by just opening and closing your left hand: it is perfect.

LIST OF PROGRAMS

- T3 rules: *T3WebDevSimplified2.html*
- AI Expert: *T3MinimaxP5ToJS.html*
- AI Expert in P5: *T3P5.html*
- Evolution as a gamer, original code: *T3Cornell.c.*
- Free transpiling of *T3Cornell.c* into JS: *T3Cornellv3.html.*
- JS implementation of the original idea: *T3Cornellv4.html.*
- To study the effect of Evolution over the distribution of fitness: *T3Cornellv5.html.*
- Connect Evolution to an easy problem: *T3Cornellv6.html.*
- Perturb the mutation rate to optimize it*:* *T3Cornellv7.html*
- Extract the best exponent from the last generation: *T3Cornellv8.html.*
- Make a match between the best exponent of the last generation and a human player: *program T3Cornellv9.html.*
- Graded master: *T3minimaxGraded.html.*
- Graded master vs. Evolution: *T3EvoOfDexterity.html*.
- Graded master vs. Evolution for limited memory devices: *T3EvoOfDexterity2.html*.
- Open-Ended Evolution vs. Graded master: *T3EvoOfDexterity3.html.*
- To learn to play and to neutralize a master are quite different goals: *T3EvoOfDexterity4.html.*
- Randomness and Evolution are equally good: *T3EvoOfDexterity5.html.*

TO KNOW MORE

- The Logic of Scientific Discovery, Karl Popper (1959).
  https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwikxcb1oafrAhWqwFkKHXbTC-4QFjABegQIARAB&url=https%3A%2F%2Fedisciplinas.usp.br%2Fpluginfile.php%2F4250035%2Fmod_folder%2Fcontent%2F0%2FTextos%2FPopper%252C%2520The%2520Logic%2520of%2520Scientific%2520Discovery.pdf%3Fforcedownload%3D1&usg=AOvVaw02ymmb711caQjNvfaHlwr7

- The game Tic-tac-toe:
  https://en.wikipedia.org/wiki/Tic-tac-toe
https://math.stackexchange.com/questions/485752/tictactoe-state-space-choose-calculation

- Game Theory and the Minimax Theorem:
  https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/

- Tic Tac Toe AI with Minimax Algorithm for P5
The Coding Train / Daniel Shiffman
https://thecodingtrain.com/CodingChallenges/154-tic-tac-toe-minimax.html
https://www.youtube.com/watch?v=trKjYdBASyQ&t=1342s

What is P5?
It is an easy and powerful language of the Processing family and Java superfamily.
https://p5js.org/
https://editor.p5js.org/codingtrain/sketches/0zyUhZdJD
https://processing.org/

- Evolution as a gamer, Original work and code:
Evolutionary Algorithms: Tic Tac Toe
Nathan and Sajeev (2009)
Cornell University, School of Electrical and Computer Engineering.
https://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/f2009/nic4_sck76/nic4_sck76/index.html

https://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/f2009/nic4_sck76/nic4_sck76/files/Evolution.c

- The overwhelmingly challenging perfection of the coronavirus: Novel 2019 coronavirus structure, mechanism of action,antiviral drug promises and rule out against its treatment
  J Biomol Struct Dyn. 2020 : 1–10.
  https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7196923/
  Subramanian Boopathi, Adolfo B. Poma, and Ponmalai Kolandaivel

- Open-Ended Evolution:
  An Overview of Open-Ended Evolution: Editorial Introduction to the Open-Ended Evolution II Special Issue. Norman Packard (Protolife) (2019)
  https://arxiv.org/pdf/1909.04430.pdf

- Evolution is incapable of disentangling a thread:
  Vol 18: The Evolutionary Theory is obviously false, 14/XI/2019
  https://ejristos.com/dl/javaevol18main.pdf

All links were alive by 19/VII/2020.