Neuroevolution algorithm for optimising a robotic manipulator trajectory

SIMONE MORETTINI MANUEL FRAILE RODRÍGUEZ
simonemo@kth.se | manuelfr@kth.se

January 11, 2021

Abstract

The purpose of this research project is to develop a machine learning solution for robotic manipulator trajectory optimisation. It is intended to use NeuroEvolution of Augmented Topologies (NEAT) algorithm to develop an Artificial Neural Network able to solve the inverse kinematics problem for a robotic manipulator with 6 rotational joints. It is expected to reach a solution that improves the performance of other state-of-the-art techniques and study future applications of the developed technology.

Keywords: NEAT, Inverse Kinematics, robotic manipulator, trajectory optimisation

Contents

List of Acronyms and Abbreviations

ANN Artificial Neural Network

CPPN Compositional Pattern-Producing Network

GA Genetic Algorithm

NEAT NeuroEvolution of Augmenting Topologies

1 Introduction

This project is built on the idea of creating a better and more optimised robot arm trajectory control algorithm. Usually, the approach used for controlling a robot arm is by solving the inverse kinematics. This consists of using the equations obtained by the mathematical model describing the system and, with them, evaluating the joint-related variables to obtain the desired position of the end effector[1]. Solving this problem can become complex if the robot has a high number of degrees of freedom because of singular configurations[2].

Alternative approaches have been studied to control robots, one of them is by using Genetic Algorithms (GAs) as presented in this overview[3]. GAs is a family of algorithms inspired by evolution. The main idea is to create a random solution and then evolve them by mixing them and mutating them as in the biology evolution [4].

The advantage of GAs is that they can produce novelty solutions and they perform well in adapting to the environment [5]. Hence, they can be used for different challenges with only small adaptations. There are very different possibilities inside this family of algorithms. In robotic arm control genetic algorithms[6], genetic programming[7] and Neuroevolution[8] have been used.

This project is focused on Neuroevolution since there are studies in its application to robotic manipulators but there is not a strict performance evaluation of them[8]. The main feature that distinguishes NeuroEvolution of Augmenting Topologies (NEAT) from other GAs is that the population that the algorithm evolves is composed by Artificial Neural Networks (ANNs). The advantage of using NEAT to create and train a ANNs is that there is no need to define the structure of the ANNs a priori. Just by deciding the number of inputs and outputs needed for the ANNs, the algorithm will create the network that will solve best the problem[9].

It is important to note the impact that this technology could have in advancing towards a more sustainable industry. Based on this compromise with green technology, one of the main parameters that is evaluated in the proposed solution, is energy consumption as it would be explained afterwards. Optimisation of energy consumption in mass industry application can have not only an economic benefit but, more importantly, a positive effect on the environment as less energy needs to be produced to achieve the same levels of production.

1.1 Theoretical framework

This project is theoretically based on three main concepts: genetic algorithms, robotic manipulator, and trajectory description in space. For each of these three points, there is a dedicated in-depth explanation that provides the reader with the theoretical knowledge needed for understanding this work.

1.1.1 The Algorithm

Evolutionary algorithms are inspired by biological evolution[10][11]. All the algorithms inside this category have some common characteristics such as a population composed of some solutions to the problem, a function to evaluate the fitness of a solution and a mechanism to create a new population using a set of solutions[12].

NEAT [13] is an algorithm that belongs to the family of evolutionary algorithms. In NEAT the individuals are ANNs. The algorithm execution procedure can be divided into the steps shown in Fig. 1. In step 1 the initial population is created with simple ANNs. In step 2 the evaluation of each individual is performed using a fitness function. If the stop criteria (based on the number of iterations or fitness value reached) is satisfied, the algorithm stops in step 3, otherwise it continues to step 4 where considering the fitness evaluated, a new population is created using crossover and mutation. The crossover consists of exchanging part of the networks between two individuals. On the other hand, the mutation consists of random changes as creating a new node, a new connection between two nodes or changing the weight of a connection.

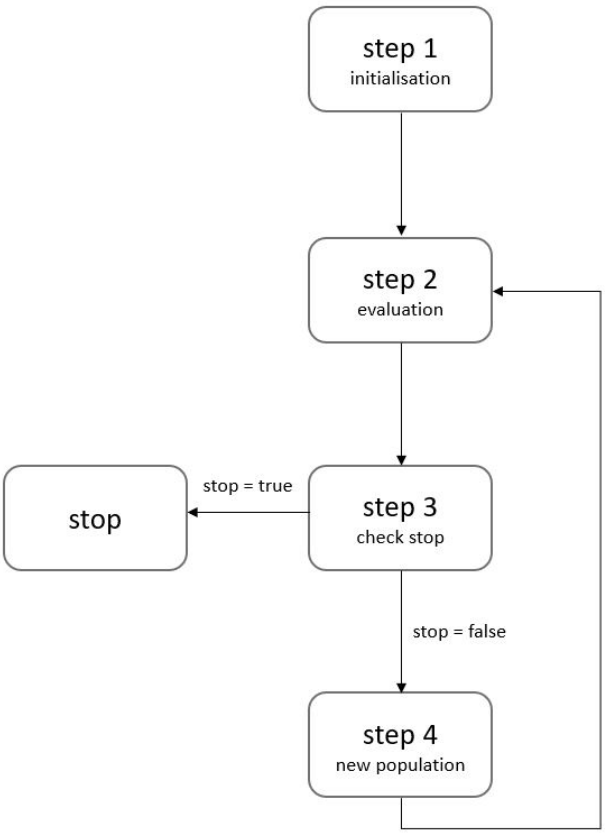

Figure 1: Flowchart for the NEAT algorithm.

1.1.2 The Robot

About the robotic arm manipulator, we are utilising the model of an ABB IRB 6400FHD [6]. In Figure 2 it is shown the different joints composing the robot, which are a total of 6, as well as their rotation directions.

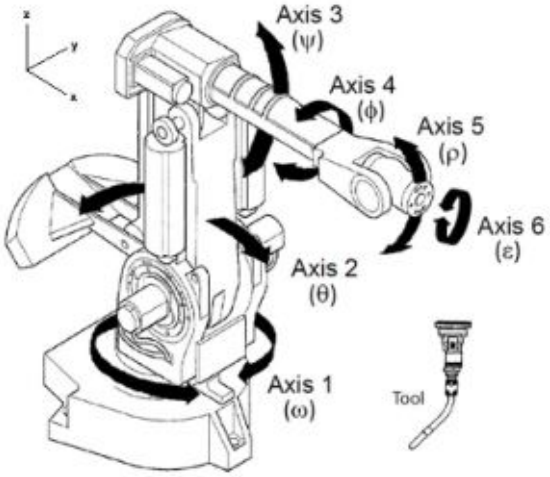

Figure 2: Robotic arm manipulator, based on the reference paper [6].

In Table 1 we can see the kinematic constraints of each joint[6]. Furthermore, the direct kinematics of the robot is based on the assumption that the end effector trajectory is computed in a clockwise coordinate system using the following transformation matrices [6]:

$$TM = A \cdot B \cdot C \cdot D \cdot E \cdot F \cdot G \cdot H \cdot I \cdot J \tag{1}$$

where the matrices are described in Table 2.

Table 1 Specification of robot motors.

Motor	Axis rotation [º]	Rotation Velocity [º/s]	Rated output [W]	Energy consumption [W·s/º]
$E_1(\omega)$	-180 ; 180	90	2800	31.1
$E_2(\theta)$	-70 ; 70	90	1900	21.1
$E_3(\psi)$	-28 ; 105	90	2400	26.6
$E_4(\phi)$	-300 ; 300	120	1000	8.3
$E_5(\rho)$	-120 ; 120	120	600	5.0
$E_6(\varepsilon)$	-300 ; 300	190	500	2.6

Table 2 Transformation matrices for the direct kinematic of the robot.

Rotation	Movement
$A = \begin{pmatrix} cos(\omega) & -sin(\omega) & 0 & 0 \\ sin(\omega) & cos(\omega) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$B = \begin{pmatrix} 1 & 0 & 0 & R1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
$C = \begin{pmatrix} cos(\theta) & 0 & sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & R2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
$E = \begin{pmatrix} cos(\psi) & 0 & sin(\psi) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\psi) & 0 & cos(\psi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	
$F = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) & 0 \\ 0 & sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$G = \begin{pmatrix} 1 & 0 & 0 & R3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
$H = \begin{pmatrix} cos(\rho) & 0 & sin(\rho) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\rho) & 0 & cos(\rho) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$I = \begin{pmatrix} 1 & 0 & 0 & R4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
$J = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\varepsilon) & -sin(\varepsilon) & 0 \\ 0 & sin(\varepsilon) & cos(\varepsilon) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	

1.1.3 The Trajectory

The main idea of this research project is to create ANNs able to provide the most optimal trajectory for a robotic arm manipulator through a Neuroevolution algorithm. Therefore, it is important to understand how we describe a trajectory in this study. For this project, a trajectory is understood as a series of points in space that together described the path to be followed by the robotic arm manipulator.

Hence, to evaluate the performance of the trajectory obtained by our algorithms, we are going to take into consideration 4 parameters that are evaluated using the following objective functions: Energy (Eq.2), Operation time (Eq.3), Rotations (Eq.4) and Positioning accuracy (Eq.5). The equations consider a

consecutive couple of points of the trajectory to extract different information and then the sum of the result for each couple is made to have the final result.

$$E_{tr} = \sum_{j=1}^{N} \sum_{i=1}^{n} [(\alpha_{b,i,j} - \alpha_{a,i,j})] \cdot EC_{r,i} \tag{2}$$

$$T_{tr} = \sum_{j=1}^{N} \max_{i=1..n} [(\alpha_{b,i,j} - \alpha_{a,i,j})] \cdot RP_{r,i} \tag{3}$$

$$A_{tr} = \sum_{j=1}^{N} \sum_{i=1}^{n} (\alpha_{b,i,j} - \alpha_{a,i,j}) \tag{4}$$

$$D_{tr} = \sum_{j=1}^{N} \sqrt{[x_{w,j} - x_{ANN,j}]^2 + [y_{w,j} - y_{ANN,j}]^2 + [z_{w,j} - z_{ANN,j}]^2} \tag{5}$$

where $\alpha_{p,i,j}$ refer to the angle of the joint i in the j-th couple of points, instead p indicates the angle in the couple and it can be a if indicate the starting angle or b if indicate the reaching angle of a couple of the trajectory; EC is the energy consumption; RP is the rotation velocity; $x_{w,j}$ is the j-point in the objective trajectory; $x_{ANN,j}$ is the x of the j-point in the end effector trajectory obtained using the ANNs developed.

1.2 Hypothesis of the research

The main hypothesis of this research project is that the control obtained using the Neuroevolution algorithm has a higher performance than using a classical inverse kinematic approach or a genetic algorithm one. Moreover, the neuroevolution based algorithm will be able to create ANNs that can control the robot for any feasible trajectory if the right training dataset is used.

2 Research Methodology

The project uses the empirical method. The kinematic analysis of a robotic arm was developed and the NEAT algorithm was implemented for creating ANNs to control the robot. Using the kinematic and the evaluation criteria expressed in section 1.1 a comparison between the proposed approach with the state-of-the-art is made. This method permits to have a fair comparison with other proposed solutions in literature because the same hardware specifications and the same evaluation criteria as the ones proposed in the paper [6] were used. That allows us to validate the research results.

2.1 Procedures and Methodologies

The project consists of a software solution for robotic arm manipulator control. This program consists of three different modules that are interconnected. The first of them is the Neuroevolution algorithm that is implemented using an already existing library called NEAT-Python[14]. The second part contains all the function and mathematical equations that describe the robot and the kinematics to be studied. The third and last part is the fitness evaluation module that judges the quality of each network.

For the Neuroevolution algorithm, Compositional Pattern-Producing Networks (CPPNs) are chosen. CPPNs are an extension of the classical ANNs because they permit to use different activation functions in the same network[15]. The structure of our network consists of n inputs and m outputs where $n = 3$ is equal to the number of the end effector coordinates and $m = 6$ is the number of joint angles needed to control the robot.

The flowchart in figure 3 shows how the software solution is structured. The NEAT module is in charge of creating the neural networks' population. Each one of them will be used to evaluate the joint angles needed to obtain a certain trajectory for the end effector. Afterwards, the direct kinematics model evaluates the described trajectory using the angles suggested by the different CPPNs. At this point, the Fitness

Evaluation Module will grade the different angles of each joint and the trajectory made. The result of this will be used by the NEAT to create a new population. The parameters used for the NEAT are indicated in Table 3.

Table 3 Parameters used for the algorithm NEAT.

Parameter	Value
Number of initial hidden nodes	3
Initial connections	50% of the possible connections between the nodes
Probability to add a new node	0.3
Probability to add a new connection	0.2
Probability to remove a node	0.1
Probability to remove a connection	0.1
Activation function available	Sigmoid, Clamped
Compatibility threshold to consider two individuals of the same species	3.0
Number of generation to consider a specie without changes stagnated	70

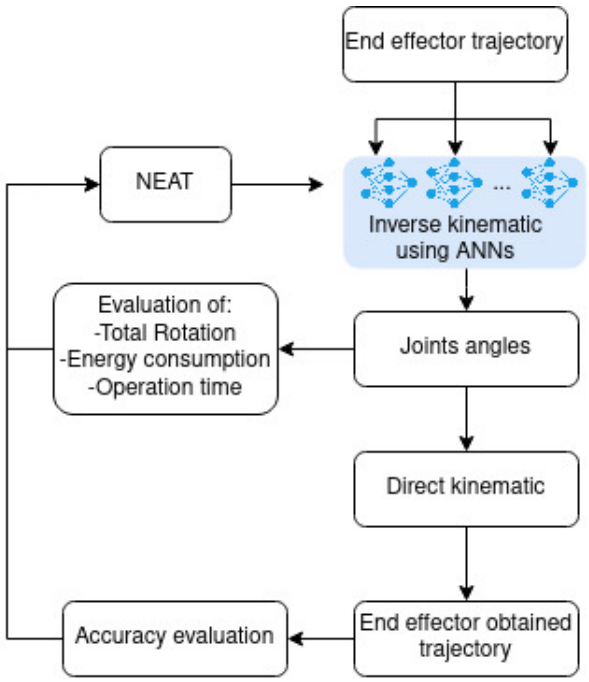

Figure 3: Flowchart for data collection and analysis.

The data used for training is the trajectory used in a previous study[6]. NEAT can accept only a scalar as fitness to evaluate an individual and the fitness functions were evaluated in the following way:

1. $Fitness_{accuracy} = D_{tr}$
2. $Fitness_{time} = D_{tr} + 0.06 \cdot T_{tr}$
3. $Fitness_{rotation} = D_{tr} + 0.005 \cdot A_{tr}$
4. $Fitness_{energy} = D_{tr} + 0.0005 \cdot E_{tr}$

In all the cases, the objective of the algorithm is to minimise the fitness function. After training, the best individual is evaluated using the objective functions and the results are compared with the result of the paper[6].

3 Results and Analysis

The results presented were obtained using a population of 250 individuals and running the NEAT for 600 generations. The number of generations was chosen based on the change in innovation. After multiple runs, 600 results to be the right number to obtain the best results. In the tables, the resulting objective functions are reported. Every row of the table contains the objective function evaluated between two consecutive points and the last row contains the accumulative value of the objective function considering the full trajectory.

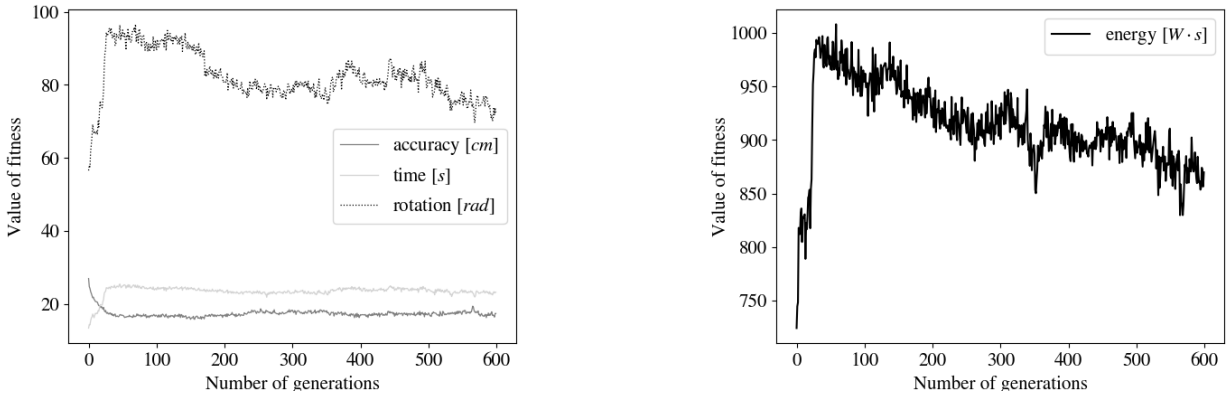

Figure 4: Graph of the objective functions obtained by optimising only the accuracy. As expected the accuracy decrease over the generations. The other objective functions are also decreasing though they are not considered in the evolution of the individuals.

Table 4 Objective functions result by optimising the accuracy.

Trajectory section	Rotation [rad]	Energy consumption [Ws]	Operation time [s]	Position accuracy [cm]
P10→P1	8.03	37.88	2.00	0.81
P1→P2	3.69	13.81	1.05	1.42
P2→P3	14.66	206.74	5.00	1.02
P3→P4	0.87	18.42	0.56	0.99
P4→P5	13.85	189.57	5.00	0.93
P5→P6	18.38	217.83	5.00	0.85
P6→P7	0.16	3.30	0.10	1.19
P7→P8	8.40	141.09	2.00	0.96
P8→P9	0.22	4.59	0.14	1.18
P9→P10	16.10	138.27	5.00	1.24
Total	**84.36**	**971.50**	**25.85**	**10.59**

In Table 4 the objective function is presented for the case in which the fitness function of the NEAT is only considering the position accuracy. From the graphs in Figure 4 is possible to see that by optimising the accuracy, also the other objective function decreases over time.

Since the positioning accuracy is essential to use the robot, it is always considered in the fitness function but, in the following cases, it is combined with the other objective functions.

The results of the attempt to obtain a time-optimal trajectory are presented in Table 5 and in the graphs in Figure 5 is possible to see how the objective functions value change during the generations. As expected the total operation time is lower than in the previous case at the cost of a bit higher accuracy.

Optimising the rotation results in the objective functions presented in Table 6 with a change in time presented in the graphs in Figure 6. In this case, as expected, the total rotation is lower than in the first case where the rotation was not considered in the fitness function.

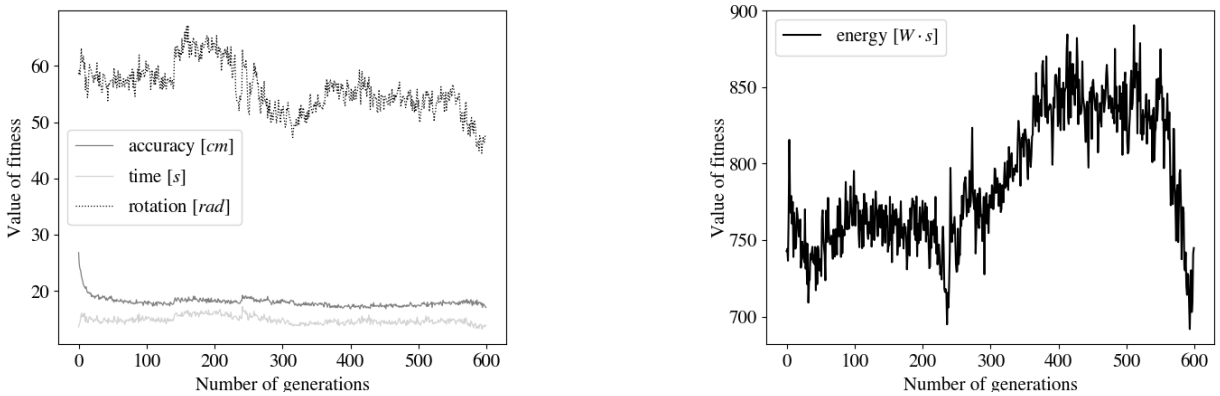

Figure 5: Graph of the objective functions obtained by optimising the operational time and accuracy. In the left figure the time and the accuracy decrease over the generations as expected. In the right figure, the energy does not follow any particular behaviour.

Table 5 Objective functions result by optimising the operational time and accuracy.

Trajectory section	Rotation [rad]	Energy consumption [Ws]	Operation time [s]	Position accuracy [cm]
P10→P1	4.38	41.53	1.90	0.78
P1→P2	0.00	0.00	0.00	1.60
P2→P3	8.80	149.38	2.00	1.20
P3→P4	0.82	17.50	0.51	1.02
P4→P5	7.33	118.65	2.00	0.97
P5→P6	11.74	164.15	2.00	0.85
P6→P7	0.35	2.89	0.17	1.19
P7→P8	9.24	153.58	2.00	0.93
P8→P9	0.29	3.69	0.09	1.25
P9→P10	1.99	42.05	1.27	1.42
Total	**44.94**	**693.42**	**11.94**	**11.21**

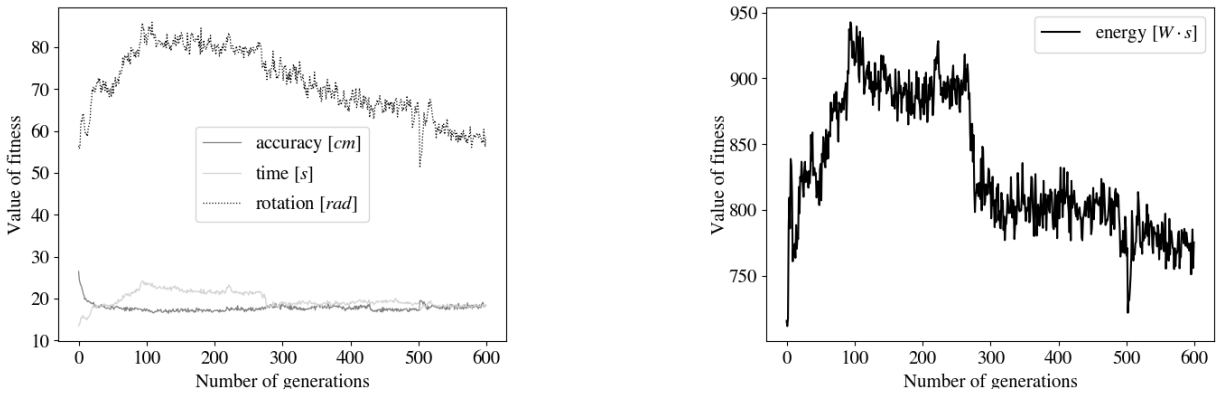

Figure 6: Graph of the objective functions obtained by optimising the rotations and the accuracy. By optimising the rotation, all the objective functions are optimized by consequence. That is an expected behaviour based on the equations of the objective functions.

The optimisation of the energy did not produce the expected result. As it is possible to see from Table 7, the energy consumption is higher than in the other executions. The graphs in Figure 7 show that the NEAT was able to optimise the energy to a low value but obtaining at the same time a too high positioning

Table 6 Objective functions result by optimising the rotation and accuracy.

Trajectory section	Rotation [rad]	Energy consumption [Ws]	Operation time [s]	Position accuracy [cm]
P10→P1	11.18	101.84	5.00	0.90
P1→P2	0.00	0.00	0.00	1.30
P2→P3	7.62	122.64	2.00	0.99
P3→P4	2.35	22.24	0.93	0.26
P4→P5	2.52	78.46	1.61	1.33
P5→P6	5.89	119.63	2.00	0.78
P6→P7	0.00	0.07	0.00	1.21
P7→P8	7.84	129.41	2.00	1.35
P8→P9	0.00	0.00	0.00	1.36
P9→P10	15.88	133.63	5.00	1.24
Total	**53.28**	**707.92**	**18.54**	**10.72**

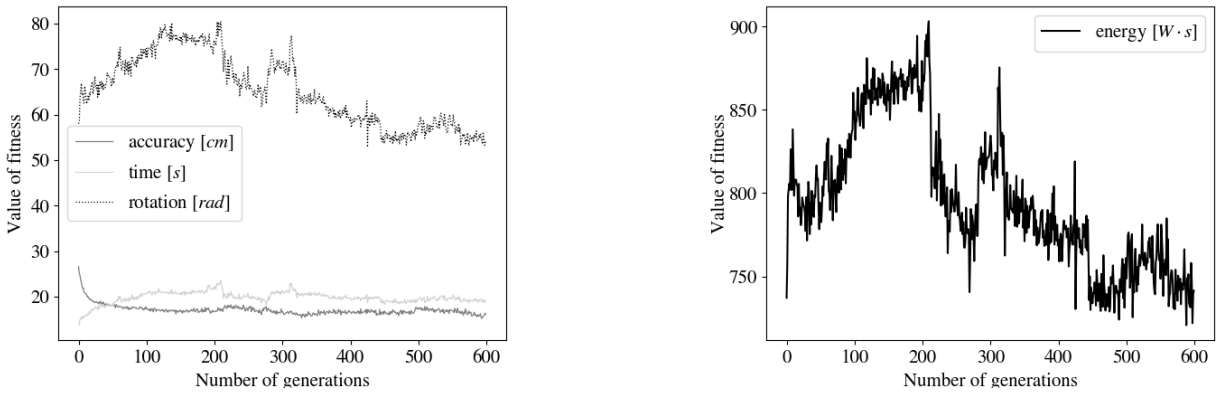

Figure 7: Graph of the objective functions obtained by optimising the energy consumption and the accuracy. NEAT is not able to optimize the energy as expected.

Table 7 Objective functions result by optimising the energy consumption and accuracy.

Trajectory section	Rotation [rad]	Energy consumption [Ws]	Operation time [s]	Position accuracy [cm]
P10→P1	10.61	89.73	5.00	1.19
P1→P2	0.68	21.17	0.43	0.76
P2→P3	18.21	147.67	3.16	1.23
P3→P4	0.00	0.00	0.00	1.05
P4→P5	17.80	145.87	3.16	0.97
P5→P6	14.13	135.79	3.16	0.81
P6→P7	15.37	122.78	5.00	1.41
P7→P8	14.95	161.07	1.93	1.61
P8→P9	2.35	6.12	0.71	1.72
P9→P10	2.94	30.25	0.52	1.24
Total	**97.04**	**860.44**	**23.06**	**11.99**

accuracy and so the best individual is not the one with low energy consumption.

4 Discussion

The expected result was to obtain an ANN able to control a robot with higher accuracy, lower energy consumption, lower operation time and lower total angular changes than the results obtained by the paper [6]. The results obtained prove wrong our assumptions as it shown in Table 8 where the GA and NEAT method are compared. The main difference appears in the energy consumption values where a very high difference is present. For the other objective functions, the difference is smaller. For all objective function, the method implemented perform worst than the method took for comparison. About the energy consumption, the value proposed in [6] seems less truthful than the obtained ones. The average consumption of energy for every degree of movement for the joint actuators is around $15W * s/^{\circ}$. This value should be smaller than the total energy consumption obtained using *GA* for a full trajectory where the amount of rotations is more than one degree but that does not happen on Table 8 where GA obtains energy consumption lower than $15W * s/^{\circ}$.

Table 8 Result obtained in this project, NEAT, compared with the results obtained using GA[6].

Optimization type	Rotation [rad]		Energy consumption [Ws]		Operation time [s]		Position accuracy [cm]	
	GA	NEAT	GA	NEAT	GA	NEAT	GA	NEAT
Position accuracy	-	84.36	-	871	-	25.85	-	10.59
Operation time	50.12	44.94	53.32	693	7.33	11.94	-	11.21
Rotation	19.74	53.28	10.66	707	8.21	18.54	-	10.72
Energy consumption	25.01	76.49	8.6	933	7.59	23.06	-	10.73

From the graphs proposed in the previous section is possible to see that NEAT improves the quality of the network through multiple generations but is not able to converge to a good value. The best average positioning accuracy obtained in one execution is 1.059 cm. A robotic manipulator in most cases needs to be very precise and this value is not acceptable to use this ANN created for trajectory planning.

Precedent studies[16] show that ANNs can be used for evaluating the inverse kinematic of a robot with good accuracy. This means that the not accurate final results obtained come from the evolution created by NEAT and not directly from the ANN. In Figure 8 one of the best ANN created during this project is shown. It is possible to notice that the network is simple since there are only 6 hidden nodes. In [17] the networks used are composed of at least five neurons for each joint. This demonstrates that the network obtained using NEAT is not enough complex to resolving the inverse kinematic with acceptable accuracy.

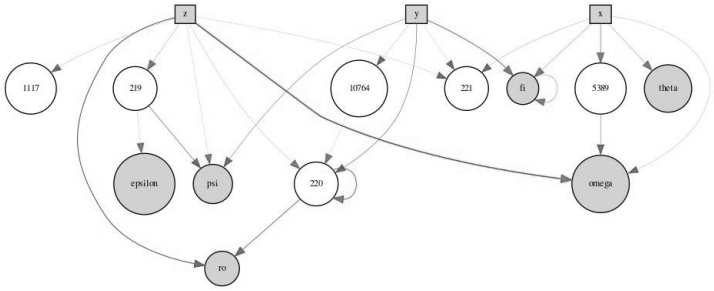

Figure 8: Scheme of an ANN produced by the NEAT algorithm. The squares are the input nodes, the grey circle are the output nodes and the other circles represent the hidden nodes.

In this work, there was the objective to compare the results obtained with the result of a previous work[6]. The objective function's values obtained in this project are worse than the ones proposed in the reference paper. So from what we obtained until now, the GA results better than NEAT in trajectory optimisation. One main problem of our approach is known to be related to the training. Until now, the

only trajectory is being used for training but, since the individuals in this project are ANNs, that can cause over-fitting that could explain the results.

5 Conclusions

After discussing the results of this project, several conclusions were reached.

- It has been demonstrated that the NEAT algorithm is not able to perform better than other GAs in robotic manipulator control. Although, some evaluation parameters are close to the results obtained in our reference paper [6], we do not consider that our results are as good as they should be to affirm that NEAT is better in robotic manipulator control.

- The main issue that our implementation of NEAT has shown is that the reached ANN's complexity is not enough to cope with such a complex problem as the inverse kinematics is. Tuning the parameters to reach more complex architectures could be a solution for this problem, but the tendency to simplify this algorithm will always be a drawback.

- Moreover, it could be considered that the training data bench is not diverse enough to reach a solution that performs a better generalisation. Improving the data source can explain the good but not enough performance of the proposed implementation. In addition to this, there is the possibility that tuning of the algorithm could be improved for making it not to converge to local minima. In any case, tuning is limited to the real capabilities of the algorithm.

- Finally, it should also be considered the possibility that our reference paper [6] has inaccuracies. In this case, our paper could demonstrate more realistic results for robotic arm manipulation based on Genetic Algorithms.

Based on the previous reasoning, it is possible to propose future work to be followed. Although the results have not been positive and most possibilities of the NEAT algorithm have been exploited, there is still room for research projects that aims to obtain better results.

The main major area to continue to investigate would be tuning. This is a manual procedure that has a high complexity as given values are arbitrary. For this reason, there is room for different tuning propositions that can improve the results having clear one of the main concepts discovered in this research: NEAT's algorithm tendency to simplification affects negatively the results and must be reduced as much as possible.

In the field of data, there is room for improving the training data with better and more diverse trajectories. In this research, the algorithm has only be trained with one trajectory (the one proposed in the reference paper). This would be, in any case, a next step after a better tuning configuration has been discovered as improving the data bench gains especial importance for generalisation.

Lastly, it would be interesting to contact the reference paper's writers to validate their results and get to know if this research project could be, in fact, a more reliable and realistic approach to Genetic Algorithm implementation for robotic manipulator control.

References

[1] T. P. Singh, P. Suresh, and S. Chandan, "Forward and inverse kinematic analysis of robotic manipulators," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, no. 2, pp. 1459–1468, 2017.

[2] K. Tchon and R. Muszynski, "Singular inverse kinematic problem for robotic manipulators: a normal form approach," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 93–104, 1998.

[3] Q. Wang, P. Spronck, and R. Tracht, "An overview of genetic algorithms applied to control engineering problems," vol. 3, 12 2003. doi: 10.1109/ICMLC.2003.1259761. ISBN 0-7803-7865-2 pp. 1651 – 1656 Vol.3.

[4] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[5] P. K. Yadav and N. Prajapati, "An overview of genetic algorithm and modeling," *International Journal of Scientific and Research Publications*, vol. 2, no. 9, pp. 1–4, 2012.

[6] S. Števo, I. Sekaj, and M. Dekan, "Optimization of Robotic Arm Trajectory Using Genetic Algorithm," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1748 – 1753, 2014. doi: https://doi.org/10.3182/20140824-6-ZA-1003.01073. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016418653

[7] K. Ward, M. Siddique, L. Maguire, and T. Mcginnity, "Genetic programming control of an articulated robotic manipulator," *Journal of Intelligent Systems*, vol. 17, 12 2008. doi: 10.1515/JISYS.2008.17.S1.109

[8] T. D'Silva, "Neuroevolution of a robot arm controller," *The University of Texas at Austin, Department of Computer Sciences*, Oct. 3, 2005.

[9] K. O. Stanley and R. Miikkulainen, "Efficient evolution of neural network topologies," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 2. IEEE, 2002, pp. 1757–1762.

[10] N. M. AL-Salami, "Evolutionary Algorithm Definition," *American Journal of Engineering and Applied Sciences*, vol. 2, no. 4, pp. 789–795, Apr. 2009. doi: 10.3844/ajeassp.2009.789.795. [Online]. Available: http://www.thescipub.com/abstract/?doi=ajeassp.2009.789.795

[11] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, 2016, pp. 261–265.

[12] D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part 1, fundamentals," *University computing*, vol. 15, no. 2, pp. 56–69, 1993.

[13] K. O. Stanley and R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002. doi: 10.1162/106365602320169811 _eprint: https://doi.org/10.1162/106365602320169811. [Online]. Available: https://doi.org/10.1162/106365602320169811

[14] A. McIntyre, M. Kallada, C. G. Miguel, and C. F. da Silva, "neat-python," https://github.com/CodeReclaimers/neat-python.

[15] K. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, pp. 131–162, 06 2007. doi: 10.1007/s10710-007-9028-8

[16] L. Aggarwal, K. Aggarwal, and R. J. Urbanic, "Use of artificial neural networks for the development of an inverse kinematic solution and visual identification of singularity zone (s)," *Procedia Cirp*, vol. 17, pp. 812–817, 2014.

[17] M. Fouz, A. Bayoumy, and S. F. Rezeka, "Neural-networks-based inverse kinematics for a robotic manipulator," in *International Conference on Aerospace Sciences and Aviation Technology*, vol. 15, no. AEROSPACE SCIENCES & AVIATION TECHNOLOGY, ASAT-15–May 28-30, 2013. The Military Technical College, 2013, pp. 1–18.