25th ABCM International Congress of Mechanical Engineering
October 20-25, 2019, Uberlândia, MG, Brazil

COB-2019-2025
JOINT ANGLES MINIMIZATION IN ROBOTIC ARM USING GENETIC ALGORITHMS

Aline de Cássia Magalhães
Rodrigo Maciel de Godoy
Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Minas Gerais, Brasil
alinecmag@gmail.com, godoyr888@gmail.com

Josué Silva de Morais
josuemorais@gmail.com

Abstract. *The study of motion in robots is called kinematics, and it is divided into two types according to what is desired. Direct Kinematics finds position and orientation from the joint data, and Inverse Kinematics finds the joints positions from the position and orientation of the claw. In the present article, based on the Denavit-Hartenberg method of Direct Kinematics, a genetic algorithm has been implemented to optimize the motion angles between two points, having the limitations of the joints given by the user and the position where the joint wishes to arrive. The minimum value of joint displacement is* 33% *lower with Radcliffe's method, keeping position error less than 1.0 cm, which is better in terms of minimizing joint angles. The Wright's method has smaller end effector position errors, but the joint angles are larger to ensure errors up to* $10^{-4} cm$ *accuracy.*

Keywords: *Forward Kinematic, Genetic Algorithms, Optimization, Robotics*

1. INTRODUCTION

Robotics is a field that is widely researched throughout their areas. Robotic arms are one in a vast options of robots and can be used industrially, assisting surgeries or physiotherapy, for example. They are composed of joints and links that are responsible for making the end effector reach the desired position and orientation in space with an objective (e.g., pick and place and welding). The kinematics are used to study these, among others, robots. It is the study of the motion and is subdivide in two fields: direct kinematics and indirect kinematics as shown in Fig. 1.

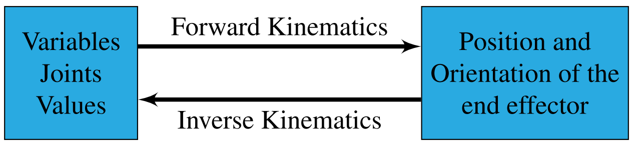

Figure 1. Representation of the inputs and outputs of each method in the Kinematics study.

Direct Kinematics calculates the position and orientation of the end effector from the joint variables. Conversely, Inverse Kinematics has the opposite relationship, finding the joint variables based on the position and orientation of the end effector, as shown by Mohammed and Sunar (2015) in their work.

When a robotic arm is designed, it is desirable for the end effector to reach his destination with the least possible effort. Thus, the present paper is based on Denavit-Hartenberg's Direct Kinematics method to implement a genetic algorithm that minimizes both joint displacement and position error. The final solution is a point that minimizes the two proposed objectives by obeying the constraint functions.

In the search for robotic arm optimization, several authors use the technique of genetic algorithms, such as Števo *et al.* (2014) which performs optimization according to operating time, angles and power consumption for a predefined robot with six degrees of freedom. Other authors such as Sharma and kaur (2011), Liu *et al.* (2007), Banga *et al.* (2007) and Kazem *et al.* (2008) in their publications have chosen to use Direct Kinematics and predefined robots due to various mapping factors of unique configurations.

In this sense, this article aims to bypass some restrictions of kinematics that, based on the initial Cartesian point and the end desired by the user, the smaller angles are calculated so that the effector reaches the destination requested with relatively small position error. The implementation must meet the precision requirements for robotic arms with any amount of joint as required by the user.

2. ALGORITHM IMPLEMENTATION

2.1 Kinematic Model

A manipulator has $n + 1$ links, with the base numbered as 0 and the end effector as n, according to Pazos (2002). Also, as shown by Tsai (1999), each combination of rotations and translations in a given ligament is called a homogeneous transformation, and for full manipulator movement, n homogeneous transformations occur.

The homogeneous matrix that depicts position and orientation between the i and $i + 1$ links is given according to Eq. (1), since R_i^{i+1} is the matrix of rotation that presents the orientation and x_i^{i+1} the position after the move (Crane III and Duffy, 2008).

$$T_i^{i+1} = \begin{bmatrix} R_i^{i+1} & x_i^{i+1} \\ 0 & 1 \end{bmatrix} \tag{1}$$

According to Crane III and Duffy (2008), the matrix calculation is performed until $i + 1 = n$. To calculate the total motion of the handler, simply multiply all the matrices, as shown in Eq. (2).

$$T_0^n = T_0^1 T_1^2 \ldots T_{n-1}^n \tag{2}$$

Thus, it is noted that the homogeneous transformation matrix will differ according to the robot geometry. In order to provide a better understanding of the Denavit-Hartenberg method, procedures will be performed for a robot with flat geometry and two revolution joints, represented by Fig. 2.

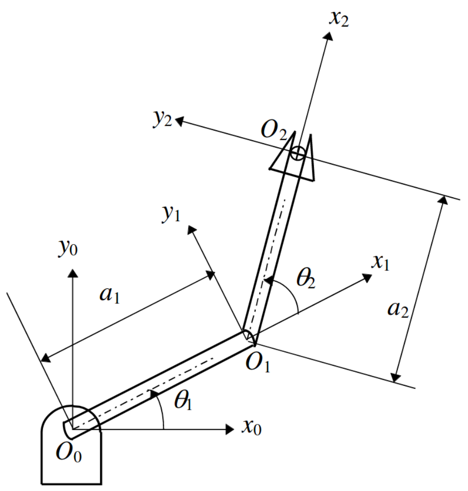

Figure 2. Flat manipulator with two revolution joints.
Available from: http://sites.poli.usp.br/p/eduardo.cabral/Cinem%C3%A1tica%20Direta.pdf

The Denavit-Hartenberg method uses four parameters to define motion between two consecutive links, which will be explained in the following topics. The parameters can be better explained in Siciliano *et al.* (2010), Ranky *et al.* (1985) and Radavelli *et al.* (2012).

- a_i: z_{i-1} and z_i wheelbase module over x_i;
- α_i: Angle between the axes z_{i-1} and z_i along x_i according to the right hand rule;
- d_i: x_{i-1} and x_i wheelbase over z_{i-1} from $i - 1$ to origin;
- θ_i: Angle between the axes x_{i-1} and x_i around z_{i-1} also according to the right hand rule.

These parameters are arranged in a table with their values for each link, according to Siciliano *et al.* (2010). For the example in Fig. 2, the Tab. 1 has the parameters for the joints.

Table 1. Denavit-Hartenberg Parameters for the Two Ligament Flat Manipulator.

Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	0	0	θ_2

Hence, it is possible to define the two partial homogeneous transformation matrices (T_0^1 and T_1^2), according to the Eq. (3) and Eq. (4).

$$T_0^1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & a_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & a_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$T_1^2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & a_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Finally, to obtain the homogeneous transformation matrix for the entire manipulator, simply multiply the partial matrices, as in Eq. (5).

$$T_0^2 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & a_1\cos\theta_1 + a_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & a_1\sin\theta_1 + a_2\sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Note, therefore, that the end effector orientation (highlighted in blue) is a rotation of $\theta_1 + \theta_2$ around the z_0 axis. The position (highlighted in red) returns x, y, and z in the first, second, and third line, respectively (Radavelli *et al.*, 2012).

Thus, the implemented algorithm calculates the homogeneous transformation matrix for the desired manipulator and returns the end effector position for the Denavit-Hartenberg parameters in each iteration of the genetic algorithm.

2.2 Genetic Algorithm

Genetic Algorithm is a method that is based on the genetic evolution of living things. This algorithm has in its theory definitions for chromosomes, genes, individuals, mutation and crossover (interaction of two individuals to form individuals for the new generation). Initially, a population of individuals is randomly generated. In each iteration, there is the selection step of the fittest, then the chromosome crossover occurs and, finally, the mutation to generate a new population (Whitley, 1994).

According to Fogel and Atmar (1990), at first, the recombination did not make any modifications to the genes, only an exchange took place between them. One of the pioneering methods to implement mutation (a modification in genes) was " *Davis averaging crossover*". Kelly Jr and Davis (1991) proposed to average among some genes. Next, Radcliffe (1992) came up with a method where the new population has uniform values relative to its parents. The Radcliffe method is one of the objects of study in this paper, and uses a random β number between 0 and 1 to calculate new individuals according to the equations described in Eq. (6) and Eq. (7). In these, p_1 and p_2 are the individuals selected to generate new individuals for the next generation, and c_1 and c_2 are the individuals they generate.

$$c_1 = \beta p_1 + (1-\beta)p_2 \tag{6}$$

$$c_2 = \beta p_2 + (1-\beta)p_1 \tag{7}$$

Another method used in this paper was proposed by Wright (1991), in which individuals are points in Euclidean space. In this way, he circumvents the cases in which the change of genes results in a population with low fitness. Let p_1 and p_2 be the parents, the newly generated individual receives the fittest among the three children generated according to the equations Eq. (8), Eq. (9) and Eq. (10), where c_1, c_2 and c_3 are the next generation candidate.

$$c_1 = \frac{1}{2}p_1 + \frac{1}{2}p_2 \tag{8}$$

$$c_2 = \frac{3}{2}p_1 - \frac{1}{2}p_2 \tag{9}$$

$$c_3 = \frac{1}{2}p_1 + \frac{3}{2}p_2 \tag{10}$$

To ensure randomness in evolution, the algorithm generates a random number r between 0 and 1 and compares it to the probability of mutation (p_m). If $r < p_m$ mutation occurs, which in this case is simply the replacement of the individual with a randomly generated one.

The individual is a vector with the angles and displacements of each joint. The first fitness function is given by minimizing the difference between the desired position and the position given by the homogeneous transformation matrix of each individual. The other function is to minimize joint angles by obeying the constraint functions. Constraints are the limitations of each user-supplied joint and the maximum acceptable value for position error.

As this is a multiobjective optimization problem, the NSGA-II algorithm will be used, which is a genetic algorithm adapted for more than one goal (Deb *et al.*, 2002). This will use a distinct approach from the pure genetic algorithm only in the method of sorting the individuals in their population.

The NSGA-II algorithm uses the criterion of dominance, which states that having two random individuals (x_1 and x_2), x_1 dominates over x_2 if x_1 is not worse than x_2 for all goals and x_1 is better than x_2 in at least one goal. Mathematically it can be stated considering minimizing the objectives that x_1 is dominant if the relationship given by Eq. (11) occurs.

$$x_1 \leq x_2 \tag{11}$$

The choice of the NSGA-II algorithm is due to three main factors: existence of elitism (preservation of the best individual of the current population in the new population), the emphasis on non-dominated solutions and their diversity preservation mechanism called crowd distance.

The selection method begins by sorting non-dominated individuals, sorting them into fronts in ascending order of non-domination. From this classification, individuals can be selected for the crossing process.

The tournament selection process that has been implemented selects n individuals randomly forming a temporary subpopulation. From this subpopulation will be chosen the individual with the lowest rank value. If there is more than one individual with a minimum rank, the selection is made by the largest value of the crowd distance method, which calculates the value of the cuboid plotted between the points surrounding the point i ($i-1$ and $i+1$) according to Eq. (12) (Fortin and Parizeau, 2013).

$$\mathcal{F}_{dist}[i] = \mathcal{F}_{dist}[i] + \frac{\mathcal{F}[i+1].m - \mathcal{F}[i-1].m}{f_m^{max} - f_m^{min}} \tag{12}$$

In Equation (12), $\mathcal{F}_{dist}[i]$ is the value of crowd distance for the individual i, $\mathcal{F}[i-1].m$ is the value of m^{th} goal of i^{th} individual in front $\mathcal{F}$, f_m^{max} is the maximum value for goal m and f_m^{min} is the minimum value for goal m.

The chosen individuals must necessarily be within the feasible decision variable and objective space. The first is the region that has all the variables that obey the constraints imposed by the optimization problem. The second is the region of the possible values for the M goals, the so-called Pareto curve. This curve is the optimal not dominated solutions between the points of this region.

2.3 PROGRAM IMPLEMENTATION

The graphic part of the program was implemented using Unity 3D software, and has three screens: one for assembling the robot with constraints for each joint, another for running the program and the last where the user manually moves the robot to set its initial and final point.

In the assembly screen, pictured in Fig. 3, it is possible to insert according to the geometry desired by the user torsion joints, rotational joints, linear joints and links. With each joint or link inserted, the user sets the maximum and minimum possible value for each joint and the size of the links to form the complete geometry of the robot to simulate.

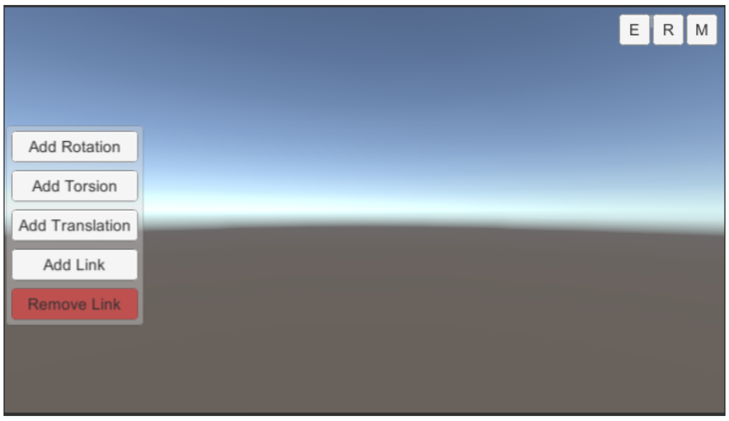

Figure 3. Screen for creating robot geometry.

After inserting the links and joints with their respective restrictions, the user can choose to perform manual tests on each joint, being able to visualize with each movement the real position of the robot in space. This can be done in the manual control screen, illustrated in Fig. 4, where the user clicking the "set" button can determine the current manipulator position as the start or end for the algorithm.

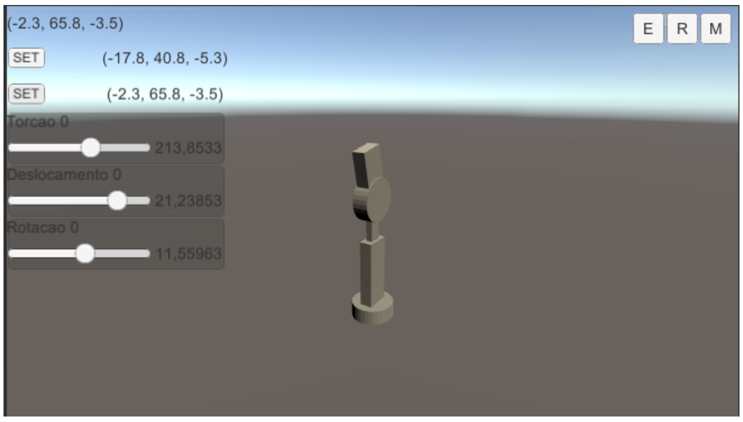

Figure 4. Screen for manual joint control.

The last screen (Fig. 5) allows to start the algorithm, based on the values defined in creation screen. Once the algorithm is started, the user will be able to make new changes only when the search comes to an end.

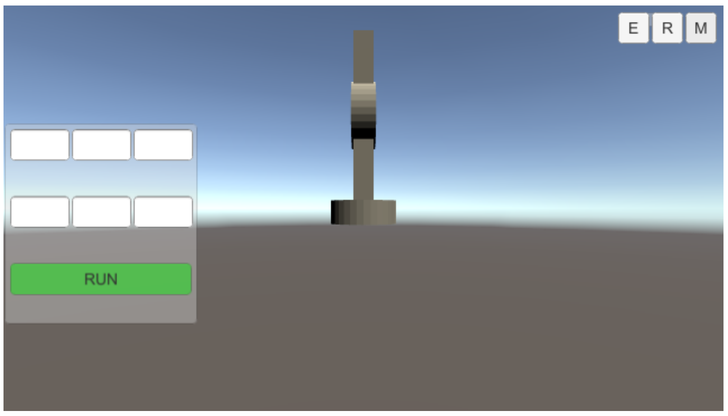

Figure 5. Screen to start algorithm.

2.4 Optimization Parameters

The optimization problem addressed in this article is given by the f_1 and f_2 functions of Eq. (13) and Eq. (14) respectively. In Equation (13), $pos_{desired}$ is the position that was entered by the user as final for the handler and pos_{AG} is the position obtained through the homogeneous transformation matrix for the individual generated by the algorithm.

$$f_1 = min(pos_{desired} - pos_{AG}) \tag{13}$$

$$f_2 = min\left(\frac{\sum_{j=1}^{num} \mathcal{J}}{num}\right) \tag{14}$$

In Equation (14), j is the index of the current individual, num is the number of joints inserted, and $\mathcal{J}$ is the value of j^{th} joint. Therefore, f_1 and f_2 are the goals to be optimized.

For each joint entered by the user, two constraint equations are generated so that the joint value remains between the values requested by the user. These constraints are calculated according to the equations given in Eq. (15) and Eq. (16), where lb is the lower limit, ub is the upper limit and $\mathcal{J}$ the value generated for the joint, which may be α, θ or d, if the joint is for rotation, twist or linear, respectively.

$$g_{min} = lb - \mathcal{J} \tag{15}$$

$$g_{max} = \mathcal{J} - ub \tag{16}$$

Another constraint that the individual must obey is the accuracy value of the position error, which was used in this paper as 1. Therefore, the individual is penalized if the position error ($\mathcal{E}_p$) is greater than accuracy value ($\mathcal{A}$) according to Eq. (17).

$$g_{acc} = \mathcal{E}_p - \mathcal{A} \tag{17}$$

Thus, there will be $2j + 1$ constraint equations. The $\mathcal{P}$ penalty is given by Eq. (18), where r_p is the penalty constant. In this work r_p has a value of 10^{10}.

$$\mathcal{P} = \sum_{p=1}^{2j+1} max(0, g_p) r_p \tag{18}$$

The function's final value for goal m (F_m) is the minimization value plus the penalty, according to Eq. (19).

$$F_m = f_m + \mathcal{P} \tag{19}$$

The best optimization parameters were defined based on the sensitivity analysis involving the number of generations, the number of individuals in each population, the probability of crossing and the probability of mutation. The values used in the sensitivity analysis are described in Tab. 2 and were tested for each of the two types of crossover (Wright and Radcliffe).

Table 2. Values used in sensitivity analysis for each optimization parameter.

Parameters	**Sensitivity Test Values**
Number of Generations	1000, 3000, 5000, 7000, 9000, 11000
Number of Individuals	10, 20, 30, 40
Crossover Probability (p_c)	0.6, 0.7, 0.8
Mutation Probability (p_m)	0.05, 0.15, 0.25, 0.35, 0.45, 0.55

Three different manipulator models were used for sensitivity analysis, totaling 3240 tests. These were performed with a random seed value of 42. The manipulators for the analysis have their geometry illustrated in Fig. 6, being (1) the first test, (2) the second test and (3) the third test.

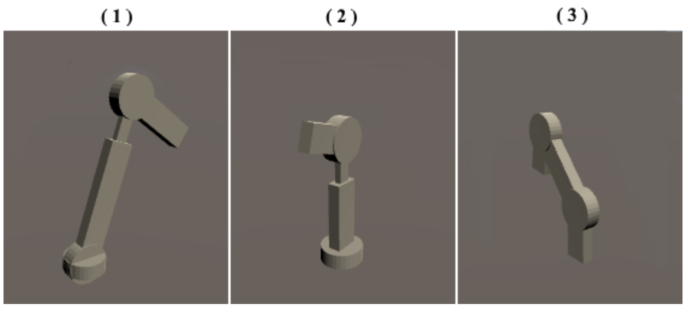

Figure 6. Manipulators used for sensitivity analysis.

The starting and ending positions chosen for each of the sensitivity analysis tests are found in Tab. 3. The link and constraint values for the joints of each of the handlers are listed in Tab. 4.

Table 3. Desired Positions for Each Handler.

	Initial Position	**Final Position**
First Manipulator	$(2.0, 91.0, -4.8)$	$(-3.2, 90.9, -4.1)$
Second Manipulator	$(-14.9, 29.1, -13.0)$	$(6.8, 63.8, -10.7)$
Third Manipulator	$(0.0, 74.4, 10.8)$	$(0.0, 71.7, 21.0)$

Table 4. Table of values used for manipulators sensitivity analysis.

	Joints	**Setted Values**
First Manipulator	T_0	$-60° \leq \alpha \leq 60°$
	R_0	$0° \leq \theta \leq 45°$
	a_0	$50cm$
	d	$0 \leq d \leq 46cm$
	R_1	$-90° \leq \theta \leq 45°$
	a_1	$33cm$
Second Manipulator	T_0	$0° \leq \alpha \leq 360°$
	a_0	$30cm$
	d	$0 \leq d \leq 25cm$
	R_1	$-120° \leq \theta \leq 120°$
	a_1	$20cm$
Third Manipulator	a_0	$21cm$
	R_0	$0° \leq \theta \leq 30°$
	a_1	$37cm$
	R_1	$-10° \leq \theta \leq 20°$
	a_2	$18cm$

The data obtained in the sensitivity analysis are saved in json extension files containing all optimization parameters and target values. A Python script has been implemented that reads this data file and performs the statistical mode calculation among the parameters of the best individual of each handler. The results obtained by him, as the best optimization parameters, are used to analyze the results.

The number of individuals and generations differs for each handler, so in the tests for results the highest values were used for these parameters. Thus, the following optimization parameters were used: 9000 generations, 50 individuals, crossover probability of 0.6 and 0.55 for mutation probability.

3. RESULTS

With the parameters obtained as often better by the sensitivity analysis, a new test was performed with a robot with distinct geometry, illustrated by Fig. 7. This geometry was chosen because the degree of freedom is greater than those used before.

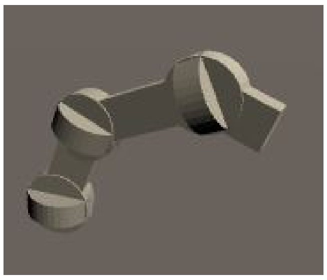
Figure 7. Manipulator used to test the sensitivity analysis's parameters.

Link and constraint values for this manipulator are in Tab. 5. The starting position of this is $(-22.1, 65.2, -16.3)$ and its end position is $(7.7, 54.5, 32.0)$.

Table 5. Values used in the manipulator for parameter validation.

Joints	**Setted Values**
T_0	$0° \leq \alpha \leq 360°$
R_0	$-30° \leq \theta \leq 30°$
a_0	$24cm$
T_1	$-90° \leq \alpha \leq 90°$
R_1	$0° \leq \theta \leq 60°$
a_1	$32cm$
T_2	$0° \leq \alpha \leq 180°$
R_2	$0° \leq \theta \leq 45°$
a_2	$15cm$

This handler was trained with each type of crossover and with random seeds from 40 to 51, with both included. Goal values f_1 and f_2 are shown in Tab. 6 as a point (f_1, f_2) on the feasible objective space.

Table 6. Results obtained in the final tests.

Seed	**Radcliffe Method**	**Wright Method**
40	$(0.49714, 32.12101)$	$(0.34541, 78.34688)$
41	$(0.62393, 15.80179)$	$(0.44789, 23.59036)$
42	$(0.04897, 87.69691)$	$(0.55309, 70.30331)$
43	$(0.36136, 97.41549)$	$(0.20493, 15.87122)$
44	$(0.34541, 78.34688)$	$(0.43360, 74.25003)$
45	$(0.73667, 80.82767)$	$(0.14501, 69.12104)$
46	$(0.44789, 23.59036)$	$(0.42723, 90.05093)$
47	$(0.20493, 15.87122)$	$(0.49714, 32.12101)$
48	$(0.62841, 28.73126)$	$(0.55407, 96.93008)$
49	$(0.14501, 69.12104)$	$(0.34053, 25.22050)$
50	$(0.51997, 30.23568)$	$(0.04897, 87.69691)$
51	$(0.42723, 90.05093)$	$(0.36136, 97.41549)$

The method proposed by Radcliffe (1992) has an average position error of $0.41557cm$, while the method proposed by Wright (1991) has an average error of $0.36327cm$. Therefore, the second method is more suitable if the main objective is to minimize the error in the final position. Although Radcliffe's method has a higher position errors, it prioritizes the minimization of the joints, with their values of approximately 15% lower than using Wright's method.

So when it comes to robots used for pick and place activities in industries or even performing ultrasound examinations or other activities that do not require high accuracy, the Radcliffe's method meets relatively well as it requires less joint displacement while maintaining the error within an acceptable range. However, if used for high precision activities such as insertion of screws and rivets or robots used to perform surgeries, it is advisable to use the Wright method which, although using greater displacement of the joints, ensures a better accuracy of the position of the end effector.

4. CONCLUSION

This paper aims to present a proposal to minimize the joint angles of a robot, maintaining a position error of less than $1.0cm$ from the joint values of any robot whose geometry is chosen by the user. For this was used a method based on genetic algorithm for multiobjective optimization problems (NSGA-II) to find the joint values that minimized the two fitness functions of the problem. One of the fitness functions (f_1) is based on the Direct Kinematics Denavit-Hartenberg algorithm and the other (f_2) is an average of the joint displacement values.

A comparison between the two crossing methods in the genetic algorithm was performed (Radcliffe and Wright). Wright's method proved to be more effective in terms of position error, while Radcliffe's method has better minimization of joint displacements, having a position error of approximately 12% greater.

5. REFERENCES

Banga, V., Singh, Y. and Kumar, R., 2007. "Simulation of robotic arm using genetic algorithm & ahp". *World Academy of Science, Engineering and Technology*, Vol. 25, No. 1, pp. 95–101.

Crane III, C.D. and Duffy, J., 2008. *Kinematic analysis of robot manipulators*. Cambridge University Press.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., 2002. "A fast and elitist multiobjective genetic algorithm: Nsga-ii". *IEEE transactions on evolutionary computation*, Vol. 6, No. 2, pp. 182–197.

Fogel, D.B. and Atmar, J.W., 1990. "Comparing genetic operators with gaussian mutations in simulated evolutionary processes using linear systems". *Biological Cybernetics*, Vol. 63, No. 2, pp. 111–114.

Fortin, F.A. and Parizeau, M., 2013. "Revisiting the nsga-ii crowding-distance computation". In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, pp. 623–630.

Kazem, B., Talib, A., Kazem, B.I., Mahdi, A.I. and Oudah, A.T., 2008. "Motion Planning for a Robot Arm by Using Genetic Algorithm". Technical Report 3. URL `https://www.researchgate.net/publication/228613178`.

Kelly Jr, J.D. and Davis, L., 1991. "A hybrid genetic algorithm for classification." In *IJCAI*. Vol. 91, pp. 645–650.

Liu, T.K., Chen, C.H. and Tsai, S.E., 2007. "Optimization on robot arm machining by using genetic algorithms". doi: 10.1117/12.784513.

Mohammed, A.A. and Sunar, M., 2015. "Kinematics modeling of a 4-DOF robotic arm". In *Proceedings - 2015 International Conference on Control, Automation and Robotics, ICCAR 2015*. ISBN 9781467375238. doi: 10.1109/ICCAR.2015.7166008.

Pazos, F., 2002. *Automação de sistema 6 robótica*. Axel Books.

Radavelli, L., Simoni, R., De Pieri, E. and Martins, D., 2012. "A comparative study of the kinematics of robots manipulators by denavit-hartenberg and dual quaternion". *Mecánica Computacional, Multi-Body Systems*, Vol. 31, No. 15, pp. 2833–2848.

Radcliffe, N.J., 1992. "Non-linear genetic representations." In *PPSN*. pp. 261–270.

Ranky, P.G., Ho, C.Y. and Ranky, P.G., 1985. *Robot modelling: control and applications with software*. IFS (Publications).

Sharma, G.S. and kaur, A., 2011. "Optimization of Energy in Robotic arm using Genetic Algorithm". *International Journal of Computer Science and Technology*, Vol. 2, No. 2.

Siciliano, B., Sciavicco, L., Villani, L. and Oriolo, G., 2010. *Robotics: modelling, planning and control*. Springer Science & Business Media.

Števo, S., Sekaj, I. and Dekan, M., 2014. "Optimization of robotic arm trajectory using genetic algorithm". In *IFAC Proceedings Volumes (IFAC-PapersOnline)*. ISBN 9783902823625. ISSN 14746670. doi:10.3182/20140824-6-ZA-1003.01073.

Tsai, L.W., 1999. *Robot analysis: the mechanics of serial and parallel manipulators*. John Wiley & Sons.

Whitley, D., 1994. "A genetic algorithm tutorial". *Statistics and computing*, Vol. 4, No. 2, pp. 65–85.

Wright, A.H., 1991. "Genetic algorithms for real parameter optimization". In *Foundations of genetic algorithms*, Elsevier, Vol. 1, pp. 205–218.

6. RESPONSIBILITY NOTICE