CEEL - ISSN 2596-2221
Universidade Federal de Uberlândia
25 a 29 de novembro de 2019

MINIMIZAÇÃO DO ERRO DE POSIÇÃO EM BRAÇOS ROBÓTICOS MODULARES UTILIZANDO ALGORITMO GENÉTICO

Aline de Cássia Magalhães*[1], Rodrigo Maciel de Godoy[1] e Josué Silva de Morais[1]

[1]FEELT - Universidade Federal de Uberlândia

Resumo **- O estudo do movimento em robôs é denominado cinemática, a qual é dividida em dois tipos de acordo com o que se deseja manipular. A Cinemática Direta encontra posição e orientação a partir dos dados das juntas, enquanto a Cinemática Inversa encontra as posições de junta dadas posição e orientação do efetuador. No presente artigo, baseado no método de Denavit-Hartenberg da Cinemática Direta, foi implementado um algoritmo genético multiobjetivo para otimizar o deslocamento das juntas entre dois pontos e o erro de posição, tendo as restrições das juntas e as posições onde o manipulador deve chegar dadas pelo usuário. O mínimo valor da média de deslocamento das juntas é 33% menor com o método de Radcliffe, mantendo o erro de posição menor que 1.0 cm, sendo este método melhor em termos de minimização dos ângulos de junta. O método de Wright possui menores erros de posição do efetuador, porém os ângulos das juntas são maiores para garantir erros com até 10^{-4}cm de precisão.**

Palavras-Chave **- Algoritmos Genéticos, Cinemática Direta, Otimização, Robótica.**

POSITION ERROR MINIMIZATION IN ROBOTIC ARM USING GENETIC ALGORITHMS

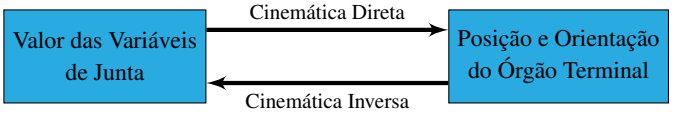
Abstract **- The study of motion in robots is called kinematics, and it is divided into two types according to what is desired. Direct Kinematics finds position and orientation from the joint data, and Inverse Kinematics finds the joints positions from the position and orientation of the claw. In the present article, based on the Denavit-Hartenberg method of Direct Kinematics, a multiobjective genetic algorithm has been implemented to optimize the motion angles between two points and the position error, having the limitations of the joints given by the user and the position where the manipulator 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.**

I. INTRODUÇÃO

A Robótica é uma área largamente pesquisada em seus diversos âmbitos. Braços robóticos são um dos vários tipos de robôs e podem ser utilizados em indústrias, no auxílio à medicina, fisioterapia, dentre outras aplicações. Esses consistem em um combinado de elos e juntas que fazem com que o efetuador chegue à posição desejada e execute algum tipo de trabalho (e.g. soldagem, *pick and place* e cirurgias). Cinemática é como se denomina o estudo do movimento nessas máquinas, e é subdividida em dois segmentos: Cinemática Direta e Cinemática Inversa, como ilustra a Figura 1.

Figura 1: Representação das entradas e saídas de cada método da Cinemática.



A Cinemática Direta calcula a posição e orientação do efetuador a partir das variáveis de junta. Por sua vez, a Cinemática Inversa tem relação oposta, encontrando as variáveis de junta tendo como base a posição e orientação do órgão terminal [10].

Quando um braço robótico é projetado, é desejável que o efetuador chegue ao seu destino com o menor esforço possível. Assim, o presente artigo é baseado no método de Denavit-Hartenberg da Cinemática Direta para implementar um algoritmo genético que minimize tanto os deslocamentos das juntas quanto o erro de posição. A solução final é um ponto que minimize os dois objetivos propostos obedecendo às funções de restrições.

Na busca por otimização de braços robóticos, diversos autores utilizam a técnica dos algoritmos genéticos, como Števo *et al.* [17] que realiza a otimização de acordo com o tempo de operação, os ângulos e consumo de energia para um robô predefinido com seis graus de liberdade. Outros autores como

*alinecmag@gmail.com

Sharma and kaur [15], Liu *et al.* [8], Banga *et al.* [1] e Kazem *et al.* [6] em suas publicações optaram por utilizar Cinemática Direta e robôs predefinidos devido a diversos fatores, principalmente à dificuldade de mapeamento decorrente da existência de configurações singulares.

Neste sentido, o presente artigo visa contornar algumas restrições da Cinemática para que, tendo como base o ponto cartesiano inicial e o final desejados pelo usuário, sejam calculados os menores ângulos para que o efetuador chegue ao destino requisitado com erro de posição relativamente pequeno. A implementação deve atender aos requisitos de precisão para braços robóticos com qualquer quantidade de junta, conforme a necessidade do usuário.

II. IMPLEMENTAÇÃO DO ALGORITMO

A. *Modelo Cinemático*

Um manipulador possui $n+1$ ligamentos, sendo a base numerada como 0 e o órgão terminal como n, de acordo com Pazos [11]. Ainda, conforme Tsai [18], cada combinação entre rotações e translações em um determinado ligamento é chamada transformação homogênea, sendo que para o movimento do manipulador completo, ocorrem n transformações homogêneas.

A matriz homogênea que retrata posição e orientação entre os ligamentos i e $i+1$ é dada conforme a Equação 1, sendo R_i^{i+1} a matriz de rotação que apresenta a orientação e x_i^{i+1} a posição após o movimento [2].

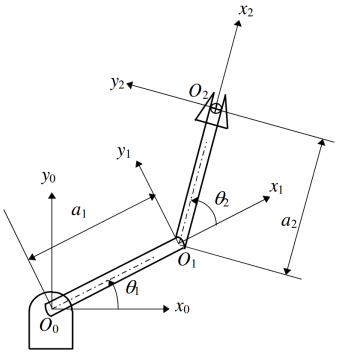
$$T_i^{i+1} = \begin{bmatrix} R_i^{i+1} & x_i^{i+1} \\ 0 & 1 \end{bmatrix} \tag{1}$$

De acordo com Crane III e Duffy [2], o cálculo das matrizes é realizado até que $i+1 = n$. Para calcular o movimento total do manipulador, basta multiplicar todas as matrizes, como mostra a Equação 2.

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

Assim, nota-se que a matriz de transformação homogênea irá diferir conforme a geometria do robô. A fim de prover melhor entendimento do método de Denavit-Hartenberg, serão realizados os procedimentos para um robô com geometria plana e duas juntas de revolução, representado pela Figura 2.

Figura 2: Manipulador plano com duas juntas de revolução. Disponível em: `http://sites.poli.usp.br/p/eduardo.cabral/Cinem%C3%A1tica%20Direta.pdf`



O método de Denavit-Hartenberg utiliza quatro parâmetros para definição do movimento entre dois ligamentos consecutivos, os quais serão explicitados nos seguintes tópicos. Os parâmetros podem ser melhor explicitados em [16], [14] e [12].

- a_i: Módulo da distância entre os eixos z_{i-1} e z_i ao longo de x_i;
- α_i: Ângulo entre os eixos z_{i-1} e z_i ao longo de x_i, de acordo com a regra da mão direita;
- d_i: Distância entre os eixos x_{i-1} e x_i sobre z_{i-1} a partir da origem de $i-1$;
- θ_i: Ângulo entre os eixos x_{i-1} e x_i em torno de z_{i-1} também conforme a regra da mão direita.

Estes parâmetros são dispostos em forma de tabela com os valores destes para cada ligamento, ainda de acordo com Siciliano *et al.* [16]. No exemplo do braço plano com duas juntas de revolução temos a Tabela I onde são definidos os parâmetros.

Tabela I: Parâmetros de Denavit-Hartenberg para o manipulador plano de dois ligamentos.

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

Daí é possível definir as duas matrizes de transformação homogênea parciais (T_0^1 e T_1^2), conforme as equações 3 e 4.

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

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

Finalmente, para obter a matriz de transformação homogênea para todo o manipulador, basta multiplicar as matrizes parciais, como na Equação 5, onde c representa o cosseno do ângulo e s o seno.

$$T_0^2 = \begin{bmatrix} c(\theta_1+\theta_2) & -s(\theta_1+\theta_2) & 0 & a_1c(\theta_1)+a_2c(\theta_1+\theta_2) \\ s(\theta_1+\theta_2) & c(\theta_1+\theta_2) & 0 & a_1s(\theta_1)+a_2s(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Nota-se, portanto, que a orientação do efetuador (destacada em azul) é uma rotação de $\theta_1+\theta_2$ em torno do eixo z_0. A posição (destacada em vermelho) retorna x, y e z na primeira, segunda e terceira linha, respectivamente [12].

Assim, o algoritmo implementado calcula a matriz de transformação homogênea para o manipulador desejado e retorna a posição do efetuador para os parâmetros de Denavit-Hartenberg em cada iteração do algoritmo genético.

B. Algoritmo Genético

Algoritmo Genético é um método que se baseia na evolução genética dos seres vivos. Este algoritmo tem em sua teoria as definições para indivíduos, cromossomos, genes, mutação e crossover (interação de dois indivíduos para formar indivíduos para a nova geração). Inicialmente, uma população de indivíduos é gerada aleatoriamente. Em cada iteração, há o passo de seleção do mais apto, então ocorre cruzamento de cromossomos e ocorre a mutação para gerar uma nova população [19].

Segundo Fogel and Atmar [4], a princípio, a recombinação não realizava modificações nos genes, ocorria apenas uma troca entre eles. Um dos métodos pioneiros a implementar mutação (uma modificação nos genes) era o "*Davis averaging crossover*". Kelly Jr and Davis [7] propôs calcular a média entre alguns genes. A seguir, Radcliffe [13] surgiu com um método onde a nova população tem valores uniformes com relação a seus pais. O método de Radcliffe é um dos objetos de estudo deste artigo, e utiliza um número β randômico entre 0 e 1 para calcular os novos indivíduos conforme as equações descritas em Eq. (6) e Eq. (7). Nelas, p_1 e p_2 são os indivíduos selecionados para gerar novos indivíduos para a próxima geração e c_1 e c_2 são os indivíduos gerados por eles.

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

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

Outro método utilizado no presente artigo foi proposto por Wright [20], no qual os indivíduos são pontos no espaço euclidiano. Desta maneira, ele contorna os casos em que a mudança dos genes resulta em uma população com baixa aptidão. Considerando que p_1 e p_2 são os pais, o novo indivíduo gerado recebe o mais apto entre os três filhos gerados conforme as equações Eq. (8), Eq. (9) e Eq. (10), onde c_1, c_2 e c_3 são os indivíduos candidatos para a próxima geração.

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

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

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

Para garantir a aleatoriedade na evolução, o algoritmo gera um número aleatório r entre 0 e 1 e o compara à probabilidade de mutação (p_m). Se $r < p_m$ ocorre a mutação, que neste caso é simplesmente a substituição do indivíduo por outro gerado aleatoriamente.

O indivíduo é um vetor com os ângulos e deslocamentos de cada junta. A primeira função de aptidão é dada pela minimização da diferença entre a posição desejada e a posição dada pela matriz de transformação homogênea de Denavit-Hartenberg a partir do deslocamento do indivíduo. A outra função de aptidão é a minimização dos ângulos das juntas. As duas funções obedecem às restrições, que são: as limitações de cada junta fornecidas pelo usuário e o máximo valor aceitável para o erro de posição.

Como se trata de um problema de otimização multiobjetivo, será utilizado o algoritmo NSGA-II, que é um algoritmo genético adaptado para mais de um objetivo [3]. Este utilizará uma abordagem distinta do algoritmo genético puro apenas no método de seleção e de escolha do melhor indivíduo.

O algoritmo NSGA-II utiliza o critério de dominância, o qual afirma que tendo dois indivíduos aleatórios (x_1 e x_2), x_1 domina sobre x_2 se x_1 não é pior que x_2 para todos os objetivos e x_1 é melhor que x_2 em pelo menos um dos objetivos. Matematicamente pode-se afirmar considerando minimização dos objetivos que x_1 é dominante se ocorre a relação dada pela Equação (11).

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

A escolha do algoritmo NSGA-II é devida a três principais fatores: existência de elitismo (preservação do melhor indivíduo da população atual na nova população), a enfatização das soluções não-dominadas e por seu mecanismo de preservação de diversidade denominado distância da multidão.

O método de seleção inicia ordenando os indivíduos não-dominados, classificando-os em fronts por ordem ascendente de não-dominação. A partir desta classificação, os indivíduos podem ser selecionados para o processo de cruzamento.

O processo de seleção por torneio que foi implementado, seleciona n indivíduos aleatoriamente formando uma subpopulação temporária. Desta subpopulação será escolhido o indivíduo com menor valor de rank. Caso exista mais de um indivíduo com rank mínimo, a seleção se dá pelo método da distância de multidão, o qual calcula o valor do cuboide traçado entre os pontos vizinhos ao ponto i ($i-1$ e $i+1$) conforme a Equação (12) [5].

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

Na Equação (12), $\mathscr{F}_{dist}[i]$ é o valor da distância de multidão para o indivíduo i, $\mathscr{F}[i-1].m$ é o valor do m^{th} objetivo do i^{th} indivíduo no front $\mathscr{F}$, f_m^{max} é o máximo valor para o objetivo m e f_m^{min} é o mínimo valor para o objetivo m.

Os indivíduos escolhidos têm necessariamente que estar dentro do espaço viável das variáveis de decisão e do espaço viável dos objetivos. O primeiro é a região que possui todas as variáveis que obedecem às restrições impostas pelo problema de otimização, e o segundo é a região dos valores possíveis para os M objetivos, sendo a chamada curva de Pareto as soluções ótimas não dominadas dentre os pontos desta região.

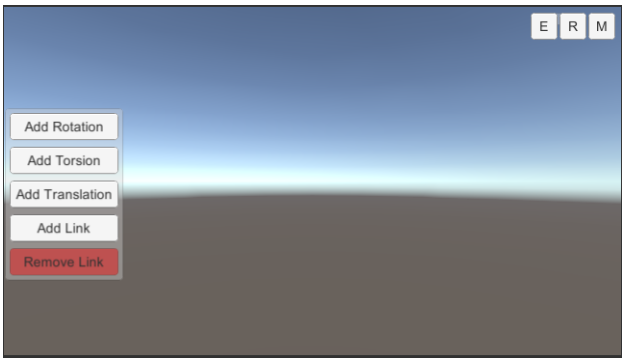
C. Implementação do Programa

A parte gráfica do programa foi implementada utilizando o software Unity 3D e o script em linguagem C#. Ele possui três telas: uma para montagem do robô com inserção das restrições para cada junta, outra para rodar o programa e a última onde o usuário movimenta o robô manualmente para setar seus pontos inicial e final.

Na tela de montagem, retratada na Figura 3, é possível inserir conforme a geometria desejada pelo usuário juntas de torção, de rotação, deslocamentos e elos. A cada junta ou elo

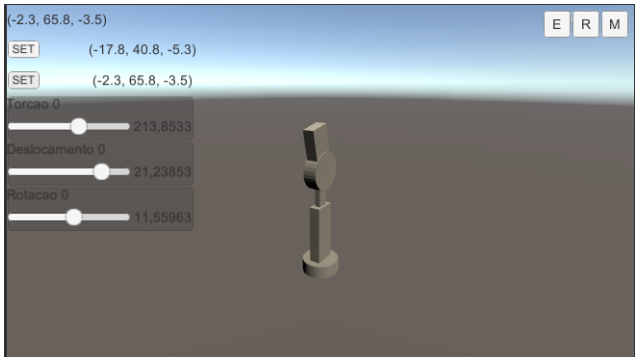
inseridos, o usuário seta o valor máximo e o mínimo possível para cada junta e o tamanho dos elos até formar a geometria completa do robô que deseja simular.

Figura 3: Tela para criação da geometria do robô.



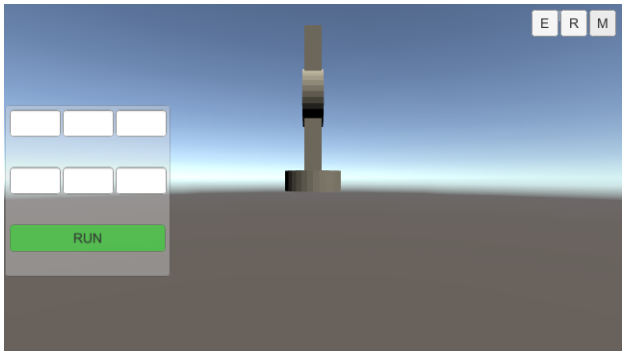
Após inseridos os elos e juntas com suas respectivas restrições, o usuário pode optar por realizar testes manuais em cada junta, podendo visualizar a cada movimento a posição real do robô no espaço. Isto pode ser realizado na tela de controle manual, ilustrada na Figura 4, onde o usuário clicando no botão "set" pode determinar a posição atual no manipulador como a inicial ou final para o algoritmo.

Figura 4: Tela para controle manual das juntas.



A última tela (Figura 5) permite iniciar o algoritmo, tendo como base os valores definidos na tela de criação. Iniciado o algoritmo, o usuário poderá realizar novas alterações apenas quando a busca desse chegar ao fim.

Figura 5: Tela para iniciar o algoritmo.



D. Parâmetros de Otimização

O problema de otimização abordado neste artigo é dado pelas funções f_1 e f_2 da Equação (13) e Equação (14) respectivamente. Na Equação (13), $pos_{desejada}$ é a posição que foi inserida pelo usuário como final para o manipulador e pos_{AG} é a posição obtida através da matriz de transformação homogênea para o indivíduo gerado pelo algoritmo.

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

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

Na Equação (14), j é o index do indivíduo atual, num é a quantidade de juntas inseridas e $\mathscr{J}$ é o valor da j^{th} junta. Portanto, f_1 e f_2 são os objetivos a serem otimizados.

Para cada junta inserida pelo usuário são geradas duas equações de restrição para que o valor da junta permaneça entre os valores solicitados pelo usuário. Estas restrições são calculadas conforme as equações dadas em Eq. (15) e Eq. (16), sendo lb o limite mínimo, ub o limite máximo e $\mathscr{J}$ o valor gerado para a junta, que pode ser α, θ ou d, se a junta for de rotação, torção ou deslocamento, respectivamente.

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

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

Outra restrição que o indivíduo deve obedecer é o valor de acurácia do erro de posição, que neste trabalho foi utilizado como 1.0. Portanto, o indivíduo é penalizado se o erro de posição ($\mathscr{E}_p$) for maior que o valor da acurácia ($\mathscr{A}$), conforme a Equação (17).

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

Desta maneira, existirão $2j+1$ equações de restrição. A penalidade $\mathscr{P}$ é dada pela Equação (18), onde r_p é a constante de penalidade que neste trabalho possui valor de 10^{10}.

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

O valor final da função para o objetivo m (F_m) é o valor da minimização somada com a penalidade, de acordo com a Equação (19).

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

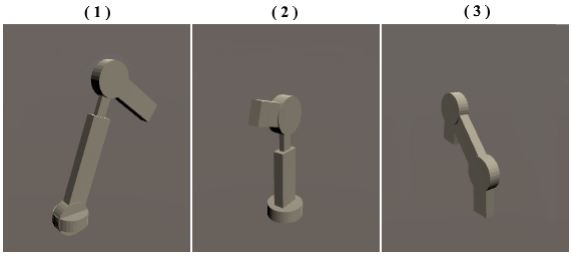
A definição dos melhores parâmetros de otimização foi realizada a partir da análise de sensibilidade envolvendo o número de gerações, a quantidade de indivíduos em cada população, a probabilidade de cruzamento e a probabilidade de mutação. Os valores utilizados na análise de sensibilidade encontram-se descritos na Tabela II e foram testados para cada um dos dois tipos de crossover.

Tabela II: Valores utilizados na análise de sensibilidade para cada parâmetro de otimização.

Parâmetro	Valores para teste de sensibilidade
Número de gerações	1000, 3000, 5000, 7000, 9000, 11000
Número de indivíduos	10, 20, 30, 40
Probabilidade de cruzamento (p_c)	0.6, 0.7, 0.8
Probabilidade de mutação (p_m)	0.05, 0.15, 0.25, 0.35, 0.45, 0.55

Foram utilizados três modelos distintos de manipulador para a análise de sensibilidade, totalizando 3240 testes com a semente randômica igual a 42. Os manipuladores para a análise têm sua geometria ilustrada na Figura 6, sendo (1) o primeiro teste, (2) o segundo teste e (3) o terceiro teste.

Figura 6: Manipuladores utilizados para análise de sensibilidade.



As posições iniciais e finais escolhidas para cada um dos testes da análise de sensibilidade se encontram na tabela III. Os valores dos elos e restrições para as juntas de cada um dos manipuladores são listados na tabela IV.

Tabela III: Posição desejada para cada manipulador.

	Posição Inicial	Posição Final
Primeiro Manipulador	$(2.0, 91.0, -4.8)$	$(-3.2, 90.9, -4.1)$
Segundo Manipulador	$(-14.9, 29.1, -13.0)$	$(6.8, 63.8, -10.7)$
Terceiro Manipulador	$(0.0, 74.4, 10.8)$	$(0.0, 71.7, 21.0)$

Tabela IV: Valores utilizados para a análise de sensibilidade dos manipuladores.

	Juntas	Valores setados
Primeiro Manipulador	T_0	$-60^\circ \leq \alpha \leq 60^\circ$
	R_0	$0^\circ \leq \theta \leq 45^\circ$
	a_0	$50cm$
	d	$0 \leq d \leq 46cm$
	R_1	$-90^\circ \leq \theta \leq 45^\circ$
	a_1	$33cm$
Segundo Manipulador	T_0	$0^\circ \leq \alpha \leq 360^\circ$
	a_0	$30cm$
	d	$0 \leq d \leq 25cm$
	R_1	$-120^\circ \leq \theta \leq 120^\circ$
	a_1	$20cm$
Terceiro Manipulador	a_0	$21cm$
	R_0	$0^\circ \leq \theta \leq 30^\circ$
	a_1	$37cm$
	R_1	$-10^\circ \leq \theta \leq 20^\circ$
	a_2	$18cm$

Os dados obtidos na análise de sensibilidade são salvos em arquivos de extensão json contendo todos os parâmetros de otimização e os valores dos objetivos. Foi implementado um script em linguagem Python que lê este arquivo de dados e realiza o cálculo estatístico da moda entre os parâmetros do melhor indivíduo de cada manipulador. Os resultados obtidos por ele como os melhores parâmetros de otimização, são utilizados para análise dos resultados.

O número de indivíduos e gerações diferiram na análise de sensibilidade, então nos testes para resultados foram utilizados os maiores valores para estes parâmetros. Assim, foram utilizados os seguintes parâmetros de otimização: 9000 gerações, 50 indivíduos, probabilidade de cruzamento de 0.6 e probabilidade de mutação de 0.55.

III. RESULTADOS

De posse dos parâmetros obtidos como frequentemente melhores pela análise de sensibilidade, foi realizado um novo teste com um robô possuindo geometria distinta, ilustrada pela Figura 7. Esta geometria foi escolhida devido à quantidade de graus de liberdade ser maior que as utilizadas para a análise de sensibilidade.

Figura 7: Manipulador usado para teste dos parâmetros da análise de sensibilidade.

Os valores dos elos e restrições para este manipulador encontram-se na Tabela V. A posição inicial deste é $(-22.1, 65.2, -16.3)$ e é desejável que ele chegue à posição $(7.7, 54.5, 32.0)$.

Tabela V: Valores usados no manipulador para validação dos parâmetros.

Juntas	Valores setados
T_0	$0^\circ \leq \alpha \leq 360^\circ$
R_0	$-30^\circ \leq \theta \leq 30^\circ$
a_0	$24cm$
T_1	$-90^\circ \leq \alpha \leq 90^\circ$
R_1	$0^\circ \leq \theta \leq 60^\circ$
a_1	$32cm$
T_2	$0^\circ \leq \alpha \leq 180^\circ$
R_2	$0^\circ \leq \theta \leq 45^\circ$
a_2	$15cm$

O manipulador foi treinado com cada tipo de crossover para sementes randômicas de 40 a 51. Os valores das funções f_1 e f_2 para cada treino encontram-se na Tabela VI como um ponto (f_1, f_2) no espaço viável de objetivos.

Tabela VI: Resultados obtidos nos testes finais.

Semente	Método de Radcliffe	Método de Wright
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)

O método proposto por Radcliffe [13] possui um erro médio de posição de 0.41557, enquanto o método proposto por Wright [20] tem como erro médio 0.36327. Assim, o segundo método é mais indicado caso o objetivo principal seja a posição mais próxima possível da desejada. Apesar do método de Radcliffe apresentar maiores erros de posição, ele prioriza a minimização das juntas, sendo seus valores de junta aproximadamente 15% menores que utilizando o método de Wright.

Portanto, tratando-se de robôs utilizados para atividades como pick and place em indústrias ou mesmo que realizam exames ultrassonográficos ou outras atividades que não necessitem alta precisão, o método de Radcliffe atende relativamente bem, já que exige menos deslocamento da junta enquanto mantém o erro de posição dentro de uma faixa aceitável. Porém, se utilizado para atividades de alta precisão como inserção de parafusos e rebites ou robôs utilizados na realização de cirurgias, é aconselhável a utilização do método Wright que apesar de usar maior deslocamento das juntas, garante uma melhor precisão da posição do efetuador.

IV. CONCLUSÃO

O presente artigo visa apresentar uma proposta de minimização dos ângulos das juntas de um robô, mantendo um erro de posição menor que 1.0 a partir dos valores de juntas de um robô qualquer cuja geometria é escolhida pelo usuário. Para isto foi utilizado um método baseado em algoritmo genético para problemas de otimização multiobjetivo (NSGA-II) para encontrar os valores de juntas que minimizassem as duas funções de aptidão do problema. Uma das funções de aptidão (f_1) é baseada no algoritmo de Denavit-Hartenberg da Cinemática Direta e a outra (f_2) em uma média dos valores dos deslocamentos das juntas.

Uma comparação entre dois métodos de cruzamento dos indivíduos no algorimo genético foi realizada (Radcliffe e Wright), sendo que o método de Wright provou ser mais eficaz em termos de erro de posição, enquanto o método de Radcliffe possui melhor minimização dos deslocamentos das juntas, possuindo um erro de posição aproximadamente 12% maior.

REFERÊNCIAS

[1] VK Banga, Y Singh, and R Kumar. Simulation of robotic arm using genetic algorithm & ahp. *World Academy of Science, Engineering and Technology*, 25(1):95–101, 2007.

[2] Carl D Crane III and Joseph Duffy. *Kinematic analysis of robot manipulators*. Cambridge University Press, 2008.

[3] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[4] David B Fogel and J Wirt Atmar. Comparing genetic operators with gaussian mutations in simulated evolutionary processes using linear systems. *Biological Cybernetics*, 63(2):111–114, 1990.

[5] Félix-Antoine Fortin and Marc Parizeau. Revisiting the nsga-ii crowding-distance computation. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 623–630. ACM, 2013.

[6] Bahaa Kazem, Ali Talib, Bahaa Ibraheem Kazem, Ali Ibrahim Mahdi, and Ali Talib Oudah. Motion Planning for a Robot Arm by Using Genetic Algorithm. Technical Report 3, 2008.

[7] James D Kelly Jr and Lawrence Davis. A hybrid genetic algorithm for classification. In *IJCAI*, volume 91, pages 645–650, 1991.

[8] Tung-Kuan Liu, Chiu-Hung Chen, and Shang-En Tsai. Optimization on robot arm machining by using genetic algorithms. 2007.

[9] Project Manager and AssistProfSırma Ç Yavuz. KINEMATIC ANALYSIS FOR ROBOT ARM. Technical report, 2009.

[10] Amin A. Mohammed and M. Sunar. Kinematics modeling of a 4-DOF robotic arm. In *Proceedings - 2015 International Conference on Control, Automation and Robotics, ICCAR 2015*, 2015.

[11] Fernando Pazos. *Automação de sistema 6 robótica*. Axel Books, 2002.

[12] Luiz Radavelli, Roberto Simoni, E De Pieri, and Daniel Martins. A comparative study of the kinematics of robots manipulators by denavit-hartenberg and dual quaternion. *Mecánica Computacional, Multi-Body Systems*, 31(15):2833–2848, 2012.

[13] Nicholas J Radcliffe. Non-linear genetic representations. In *PPSN*, pages 261–270, 1992.

[14] Paul G Ranky, Chung You Ho, and Paul G Ranky. *Robot modelling: control and applications with software*. IFS (Publications), 1985.

[15] Gouri Shankar Sharma and Amanpreet kaur. Optimization of Energy in Robotic arm using Genetic Algorithm. *International Journal of Computer Science and Technology*, 2(2), 2011.

[16] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[17] Stanislav Števo, Ivan Sekaj, and Martin Dekan. Optimization of robotic arm trajectory using genetic algorithm. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2014.

[18] Lung-Wen Tsai. *Robot analysis: the mechanics of serial and parallel manipulators*. John Wiley & Sons, 1999.

[19] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[20] Alden H Wright. Genetic algorithms for real parameter optimization. In *Foundations of genetic algorithms*, volume 1, pages 205–218. Elsevier, 1991.