Comparação entre Algoritmos Genéticos e Redes Neurais aplicados ao Problema da Cinemática Inversa

N. Lemke, A. Cechin, F. Osório, A. Wagner, A. Werhli and F. Heinen
Universidade do Vale do Rio dos Sinos – Unisinos, Centro de Ciências Exatas e Tecnológicas
Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
São Leopoldo – RS, Brasil, 93022-000
lemke,cechin,osorio,adilea,wehrli,farlei@exatas.unisinos.br

Abstract

This article proposes a comparison between Genetic Algorithms and Neural Networks on the Inverse Kinematics Problem. This problem consists on determining the set of joint angles of a robot arm in order to bring it to a certain position. We show that the Neural Network approach is adequate to solve this problem, if there is an onto function from the input dataset to the output dataset, while the Genetic Algorithm solve it accuratelly, but not in real time. We discuss which factors determine the Genetic Algorithm performance and the reason why the Neural Network fails and we conclude that in both cases the key factor is the existence of multiple solutions for this problem.

Keywords: Neural Networks, Genetic Algorithms, Inverse Kinematics, Robotic Arms

Resumo

Neste artigo, aplicamos duas técnicas clássicas da Inteligência Artificial - Redes Neurais e Algoritmos Genéticos - no Problema da Cinemática Inversa. Este problema consiste em determinar os ângulos das articulações de um braço robótico que levam o *end-effector* (garra) até uma posição desejada. Mostramos que a Rede Neural proposta é adequada para o problema em questão, caso haja uma função sobrejetora dos dados de entrada para os de saída, enquanto que o algoritmo genético o resolve acuradamente, porém não em tempo real. Finalmente discutimos os fatores que determinam o desempenho do Algoritmo Genético e o fracasso da Rede Neural. Com base em nossos resultados, concluímos que o fator determinante em ambos os casos é a multiplicidade das soluções encontradas.

Palavras Chaves: Redes Neurais, Algoritmos Genéticos, Cinemática Inversa, Braços Robóticos

1 Introdução

Os robôs são classificados como robôs industriais (manipuladores ou braços robóticos) e robôs autônomos [6]. Braços robóticos são dispositivos mecânicos equipados com atuadores e sensores sob o controle de computadores [3], que executam tarefas envolvendo movimentos no espaço físico, apesar de seus dados sensoriais serem ruidosos, incompletos e inexatos. Estes são geralmente projetados para executar tarefas simples e repetitivas em numerosas aplicações na área industrial. Contudo existem demandas por braços que executem tarefas não repetitivas, nestes casos o sistema de controle deve definir em tempo real como o braço irá se deslocar ao longo de uma trajetória a fim de atingir uma dada posição no espaço. Sendo assim, o robô deve ser capaz de transportar um objeto de um ponto a outro evitando possíveis colisões.

Para o posicionamento do braço é necessário determinar o ângulo de cada uma das suas articulações (graus de liberdade) em relação a posição do *end-effector*. Dependendo da construção do robô podem existir pontos que não sejam acessíveis ou pontos que podem ser atingidos com diferentes configurações de ângulos.

O problema da Cinemática Inversa (*"Inverse Kinematics Problem"*) [1] consiste em determinar o ângulo de cada uma das articulações para que o *end-effector* atinja uma determinada posição. Os atuadores deverão ser ajustados de forma a atingir estes ângulos. Este problema tem sido abordado por técnicas analíticas [1] e métodos baseados em Inteligência Artificial entre eles Redes Neurais, Algoritmos Genéticos e Sistemas Nebulosos.

Os métodos baseados em Redes Neurais partem de um conjunto de dados, obtidos através da simulação da Cinemática Direta do robô, que serão utilizados para treinar a Rede Neural. Oyama e Tachi [10] desenvolveram um novo modelo de Rede Neural para o tratamento das descontinuidades na relação entre posição e ângulos. Yang, Moghavvemi e Tolman comparam dois modelos neurais -*backpropagation* e RBF (Radial Basis Functions)- aplicados ao problema da Cinemática Inversa [17].

Rouvinen [13] demonstrou que Algoritmos Genéticos são capazes de resolver o problema da Cinemática Inversa para um manipulador com 3 graus de liberdade. Zakharov [18] mostrou que Algoritmos Genéticos são capazes de encontrar soluções para o problema da Cinemática Inversa com exatidão mesmo se o robô possuir codificadores de posição de baixa resolução. Chocron [2] propôs um algoritmo evolucionário que permitia determinar não apenas a posição final mas também trajetórias que evitam colisões.

A natureza deste problema é um excelente laboratório para comparar diferentes técnicas adaptativas da Inteligência Artificial, permitindo levantar suas qualidades e limitações. Optamos por comparar o desempenho de um Algoritmo Genético e de uma Rede Neural. Com este objetivo e baseados nos trabalhos anteriores, tratamos o problema da Cinemática Inversa usando uma Rede Neural do tipo MLP (Multi-Layer Perceptron) [4] e um Algoritmo Genético do tipo estado estacionário [7].

O modelo de robô utilizado neste trabalho é o Robix[1](RCS-6) [12] que combina diversas características de robótica para fins acadêmicos. O modelo Robix montado na UNISINOS é composto por 6 servos com 4 cabos de extensões de servo, base de pivô *link* básico e *link* diagonal com roda, *links* com 4 graus de liberdade e base de pivô externo. Baseado no braço robótico disponível, foi implementado um simulador. Na figura 1, mostramos o simulador apresentando o movimento do braço robótico. A partir do uso deste simulador podemos obter os dados necessários para a implementação e validação de nossos experimentos.

Figura 1: Visualização da movimentação do braço do Robix no Simulador.

Este artigo está estruturado da seguinte forma: o modelo de redes neurais, o Algoritmo Genético e os resultados obtidos são apresentados nas seções 2 e 3 respectivamente. Na seção 4 os resultados são

[1]Robix é uma marca registrada de Advanced Design, Inc., Tucson Arizona, USA.

comparados e apresentamos nossas conclusões acerca da adequabilidade destes dois métodos ao problema tratado.

2 Redes Neurais

Redes Neurais Artificiais são modelos utilizados para identificação e controle de processos [8, 9]. Neste trabalho, as redes são utilizadas para aproximar a relação existente entre a posição do *end-effector* e os ângulos das articulações. Com este objetivo foi empregada uma Rede Neural do tipo *feed-forward* MLP (*Multi-Layer Perceptron*) treinada através de aprendizado supervisionado.

Os dados utilizados no treinamento são compostos pela posição do *end-effector* (X, Y, Z) e pelos 4 ângulos das articulações do braço ($\theta_1, \theta_2, \theta_3, \theta_4$). No caso da Cinemática Inversa, são utilizados como entrada (X, Y, Z) e buscamos obter na saída os ângulos ($\theta_1, \theta_2, \theta_3, \theta_4$) correspondentes. Os dados foram gerados aleatoriamente, utilizando uma distribuição uniforme no espaço dos ângulos. Para fins de treinamento da Rede Neural, os dados foram divididos pelos respectivos intervalos de variação [15]. A base de dados de treinamento continha 700 pontos e a base de validação continha 300 pontos.

2.1 Descrição da rede

A ferramenta escolhida para o treinamento foi o programa SNNS (*"Stuttgart Neural Network Simulator"*) [14], um simulador de Redes Neurais desenvolvido na Universidade de Stuttgart. Esta ferramenta oferece um ambiente eficiente e flexível para auxiliar a construção e o treinamento de Redes Neurais, possuindo um grande número de algoritmos de aprendizado.

As Redes Neurais foram treinadas utilizando as seguintes opções: algoritmo de treinamento RProp (*"Resilient Propagation Learning"*)[11], função de inicialização de pesos *randomize-weight*, atualização dos pesos em ordem topológica, função de transferência dos neurônios tangente hiperbólica na camada oculta e função identidade ($f(x) = x$) na camada de saída. RProp é um algoritmo de aprendizagem de segunda ordem para redes *feedforward* de múltiplas camadas que possui a vantagem de adaptar a amplitude do passo em função da proximidade de um mínimo local e assim alcançar mais rapidamente o mínimo da função.

Foram testadas as seguintes topologias para as redes: 3×4 (linear), 3×4×4 (3 neurônios na entrada, 4 neurônios na camada oculta e 3 neurônios na saída), 3×8×4 e 3×16×4.

A rede treinada para a Cinemática Inversa que apresentou o menor grau de erro foi a rede 3x16x4 com um erro padronizado (RMS - Raiz da Média do Erro Quadrático) de 23.7% do intervalo de variação, que corresponde aproximadamente a 184 mm [15].

Para fins de comparação com o Algoritmo Genético, foi utilizada a Rede Neural com topologia 3×16×4 treinada com os dados citados acima.

Para facilitar a análise dos resultados optamos por selecionar um conjunto de pontos de teste sobre um plano. Esta escolha não é uma limitação de nossa implementação mas apenas visou facilitar a análise dos dados e a comparação com os resultados do Algoritmo Genético. Um dos fatores que dificultam as nossas simulações é a existência de "pontos cegos"que não podem ser atingidos pelo braço do robô. Assim, a escolha de uma malha arbitrária é impossível. O conjunto de pontos acessíveis de nosso braço forma um conjunto bastante complexo, representado nas figuras 2 e 3.

A solução encontrada foi a escolha de uma malha cilíndrica onde se possa garantir a existência de uma solução. Tomamos os pontos ao longo de um plano arbitrário, traçamos 5 círculos centrados na origem com diferentes raios e tomamos 4 pontos em cada círculo, não representados nas figuras. A escolha dos raios está vinculada a acessibilidade destes pontos.

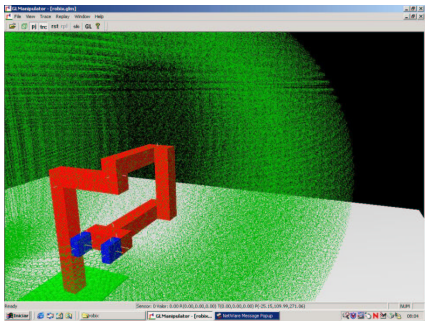

Figura 2: O simulador e os pontos acessíveis pelo braço. Os pontos foram obtidos pela discretização do espaço dos ângulos em intervalos de 0.01 graus.

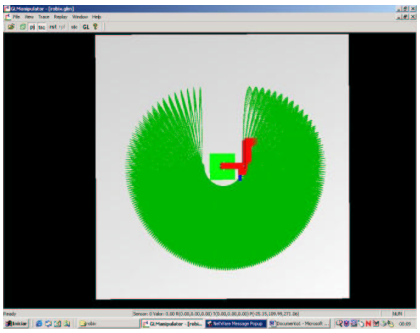

Figura 3: Os mesmos pontos da figura 2 vistos de cima. Pode-se notar que existe uma região próxima ao braço que não pode ser acessada.

2.2 Resultados

Os resultados obtidos para estes pontos são apresentados na tabela 1. x_o, y_o e z_0 representam a posição pretendida em milímetros e x', y' e z' a saída da Rede Neural, D é a distância euclidiana entres os dois pontos em milímetros.

x_o	y_o	z_o	x'	y'	z'	D
176	100	176	180.11	41.17	152.24	63.81
-176	100	176.7	-196.25	76.00	217.02	50.74
-190	100	190	-219.37	74.49	189.42	8.24
-212	100	212	-244.42	67.08	135.43	89.49
-250	100	0	-240.48	59.56	88.27	97.55
-270	100	0	-250.30	56.77	91.56	103.15
-300	100	0	-259.34	52.70	79.84	101.31
-49	100	49	-146.15	65.33	237.85	214.52
-70	100	0	-234.56	69.61	149.31	224.27
0	100	270	-48.55	104.20	105.51	171.55
0	100	300	-66.12	179.50	94.64	229.92
0.00	100	250.0	-61.18	58.40	134.59	137.08
0	100	70	-17.10	53.77	272.96	208.86
190	100	190	173.95	36.28	96.19	115.41
212	100	212	127.17	43.94	6.98	229.00
270	100	0	261.56	59.57	62.77	75.13
300	100	0	272.53	56.00	27.45	58.68
250.0	100	0	251.50	61.70	75.59	84.75
49	100	49	122.51	47.76	234.08	205.25
70	100	0	226.49	52.34	150.95	222.59

Tabela 1: Resultados obtidos com a Rede Neural. x_o, y_o e z_o são as coordenadas do ponto que o *end-effector* do robô deve atingir, x', y' e z' são a saída da Rede Neural e D é distância entre (x, y, z) e (x', y', z').

A média dos erros obtidos foi de 136.07 milímetros, sendo o menor erro 8.24 mm e o maior erro 229.92 mm. Os dados indicam que a Rede Neural não foi capaz de aproximar de modo satisfatório a relação entre a posição do *end-effector* e os ângulos das articulações. Estes mesmos dados foram usados junto ao Algoritmo Genético.

3 Algoritmo Genético

Em ciência e tecnologia nos deparamos muitas vezes com problemas complexos de otimização. O planejamento e a construção de qualquer artefato tecnológico implica na escolha de diversas variáveis com o objetivo de garantir o desempenho do artefato ou minimizar seu custo. A complexidade do espaço de busca destas soluções que incluem muitas variáveis sujeitas a vínculos inviabilizam o uso de técnicas de otimização tradicionais. A Cinemática Inversa é um exemplo típico desta classe de problemas.

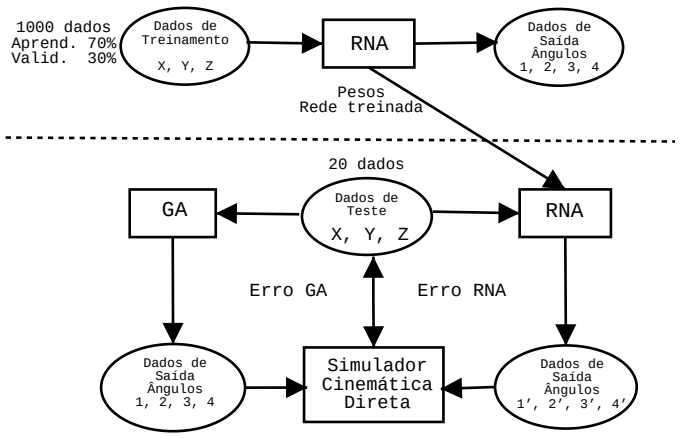

Figura 4: Cálculo do erro Euclidiano entre posições X, Y, Z e X', Y', Z' do *end-effector*.

Uma inspiração importante para o desenvolvimento de heurísticas é a seleção natural que tem se mostrado capaz de gerar soluções intrincadas e eficazes em uma infinidade de situações. Tradicionalmente, as heurísticas inspiradas na seleção natural tem sido nomeadas ¨Algoritmos Genéticos¨ desde o trabalho seminal de J. H. Holland [5].

Os Algoritmos Genéticos (AG) combinam a sobrevivência dos mais aptos com uma troca aleatória de informação entre membros de uma população composta por cadeias de bits. A cada geração um novo conjunto de criaturas, (cadeias de bits) é criado usando características dos elementos mais adaptados da anterior, isto permite um uso eficiente das informações contidas na população para especular novas possíveis soluções.

O poder destas heurísticas decorre diretamente da composição de dois processos: a mutação e o *crossover*. O primeiro é equivalente a um processo de busca localizada e permite que uma vez que as seqüências estejam próximas a um máximo local da função f, que queremos maximizar, elas convirjam para ele, já o *crossover* permite uma busca muito mais completa no espaço de parâmetros permitindo que máximos globais sejam encontrados.

Suponhamos que queremos maximizar a função f que depende de vários parâmetros x_i que podem ser tanto números inteiros, reais ou variáveis binárias. Para simplificar a discussão supomos que f seja sempre positiva e denominamos f de *fitness*.

A evolução da heurística pode ser decomposta em diversos operadores: seleção, mutação, *crossover*, reescalonamento entre outros. Cada um dos operadores é praticamente independente dos demais o que garante uma boa modularidade para o problema, ou seja, sempre podemos adequar a heurística para o problema que queremos resolver através de uma escolha judiciosa dos operadores que iremos aplicar.

O funcionamento de um algoritmo genético pode ser resumido como:

1. Inicie com uma população de N_{pop} indivíduos descritos por l bits escolhidos aleatoriamente.
2. Calcule a função de *fitness* para cada um dos indivíduos.
3. Repita cada um dos passos até que N_{pop} indivíduos tenham sido escolhidos
   (a) Selecione um par de indivíduos da população.
   (b) Aplique o operador de *crossover*.
   (c) Aplique o operador de mutação.
   (d) Coloque o indivíduo na nova população.
4. Substitua a população atual pela nova população.
5. Repita o passo 2 até que o critério de parada tenha sido atingido.

3.1 Descrição do Algoritmo Genético

Nesta seção descrevemos o Algoritmo Genético utilizado em nossas simulações. O primeiro passo é descrever a estrutura dos cromossos, como temos quatro juntas e elas definem a posição da *end-effector* e sua orientação

é conveniente considerar um cromossomo com a estrutura $\{\theta_1, \theta_2, \theta_3, \theta_4\}$. Onde cada um dos ângulos é uma variável real que pode assumir valores dentro dos limites físicos das juntas do braço.

A função de *fitness* é escolhida para permitir que o *end-effector* atinja uma determinada posição independente de sua orientação, com este objetivo em mente a função de *fitness* é definida como:

$$f(\{\theta_i\}) = \sqrt{(x - x_o)^2 + (y - y_o)^2 + (z - z_o)^2} \tag{1}$$

onde x_o, y_o e z_o é o ponto que queremos atingir e x, y e z é a posição da garra determinada pelos ângulos θ. Neste trabalho a orientação do *end-effector* não é considerada, mas em aplicações em que isto seja relevante poderíamos facilmente considerar este fator.

Como a nossa codificação dos ângulos é baseada em números reais, temos que introduzir operadores de mutação e *crossover* adequados. O operador de mutação toma um novo valor de acordo com uma distribuição gaussiana centrada no ponto inicial com largura unitária. O operador de *crossover* toma a média dos dois ângulos θ_i. A taxa de mutação é de 10% e a taxa de *crossover* é de 90% .

Além destes operadores a seleção foi do tipo roda de roleta. A população considerada continha 50 indivíduos e consideramos um Algoritmo Genético do tipo *Steady State* que mantém 10% dos indivíduos mais adaptados para a próxima geração.

A implementação do Algoritmo Genético foi realizada utilizando a biblioteca GAlib que pode ser obtida em `http://lancet.mit.edu/ga/`.

3.2 Resultados

A convergência do algoritmo é lenta e depende do ponto que queremos atingir. Visando quantificar esta dependência analisamos a evolução temporal do melhor indivíduo da população para 10 diferentes inicializações do algoritmo. Resultados típicos são apresentados na figura 5. Onde apresentamos a dependência temporal de $F(t)$, que representa o *fitness* do indivíduo melhor adaptado na geração t.

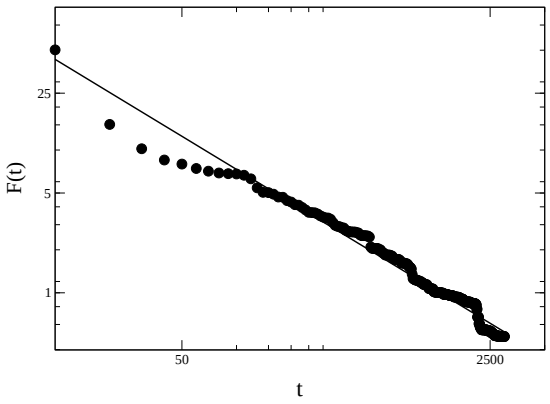

Figura 5: Evolução temporal do *fitness* do melhor indivíduo da população, $F(t)$, quando x_o=-190, y_o=100 e z_o=190.

Como a curva possui um comportamento linear quando t é grande em um gráfico log-log, podemos deduzir que $F(t)$ pode ser descrita por uma equação do tipo:

$$F(t) = At^b \tag{2}$$

onde A e b são parâmetros de ajuste. É mais conveniente caracterizar o dempenho do algoritmo utilizando o parâmetro b do que simplesmente o valor de $F(t)$ após um número fixo de gerações por que caracterizamos toda a evolução temporal do algoritmo para valores de t muito grandes. Além disto, com os parâmetros A e b conhecidos é fácil estimar quantas gerações serão necessárias para atingirmos um ponto com uma determinada precisão.

Os resultados obtidos após o ajuste das curvas que descrevem a evolução estão representados na tabela 2. Onde podemos comprovar que tanto o *fitness* do melhor indivíduo após 5000 gerações como o parâmetro b variam fortemente com (x_o, y_o, z_o).

x_o	y_o	z_o	A	b	D
176	100	176	116.78	-0.7	0.34
-176	100	176	142.94	-0.69	0.45
-190	100	190	214.0	-0.73	0.49
-212	100	212	25.84	-.35	1.21
-250	100	0	115.5	-0.71	0.49
-270	100	0	12.13	-0.29	1.05
-300	100	0	55.29	-0.4	2.40
-49	100	49	136.39	-0.81	0.18
-70	100	0	151.23	-0.77	0.26
0	100	270	51.36	-0.56	0.49
0	100	300	36.42	-0.36	1.98
0	100	250	55.33	-0.64	0.34
0	100	70	272.50	-0.91	0.40
190	100	190	480.74	-0.89	0.40
210	100	212	13.74	-0.27	1.36
270	100	0	107.92	-0.64	0.59
300	100	0	106.4	-0.52	1.42
250	100	0	209.59	-0.81	0.24
49	100	49	52.31	-0.70	0.18
70	100	0	131.67	-0.81	0.20

Tabela 2: Resultados obtidos com o Algoritmo Genético. x_o, y_o e z_o são as coordenadas do ponto que queremos atingir, A, b são parâmetros de ajuste e D é distância do melhor resultado obtido com o Algoritmo Genético no ponto (x, y, z).

Uma hipótese que nos parece razoável é que os pontos mais afastados sejam os mais difíceis de atingir. Para investigar esta hipótese comparamos o expoente b com a distância a origem do sistema de coordenadas. O resultado é apresentado na figura 6. O gráfico indica, apesar das barras de erro, que existe uma correlação entre as duas quantidades.

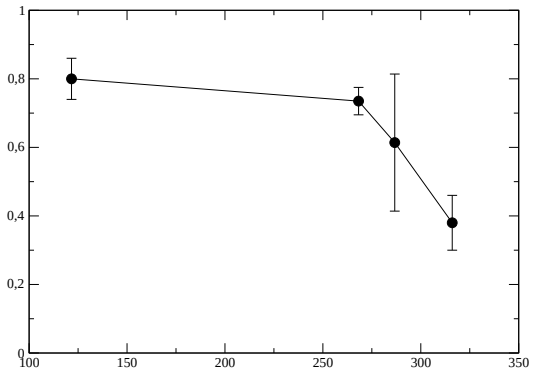

Figura 6: Expoente b em função da distância a origem.

Outra questão interessante é investigar se as soluções encontradas pelo AG dependem da inicialização. Na figura 7 mostramos diferentes soluções para cada valor de x_o, y_o e z_o, para permitir a visualização escolhemos um conjunto de 3 ângulos entre os quatro possíveis. Esta escolha particular não muda qualitativamente os resultados. De fato em inúmeros casos os Algoritmos Genéticos encontram diferentes soluções, como pode ser depreendido da figura, enquanto que em outros casos as soluções encontradas estão muito próximas. É interessante observar que para as soluções representadas por pontos em cinza claro os valores de b são altos (em módulo) indicando uma convergência rápida, enquanto que no caso das soluções representadas por pontos cinza escuro, temos valores de b pequenos (em módulo) ou seja uma convergência lenta. Isto pode ser facilmente compreendido pois no primeiro caso o espaço de soluções admissíveis é maior do que no segundo

caso. Ou seja a redundância permite que o AG convirja mais rapidamente.

Este fato também explica os resultados apresentados na figura 6, pois a medida que nos afastamos da origem o conjunto de ângulos para atingir uma determinada posição diminui seu volume de forma substancial.

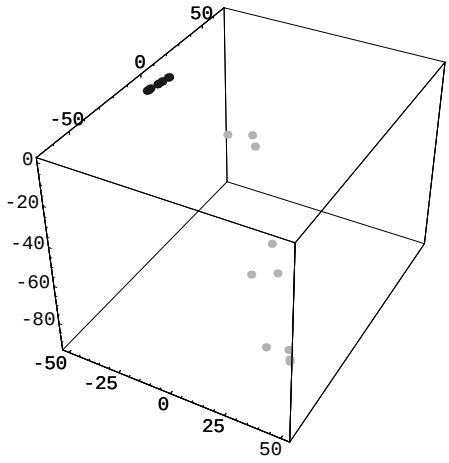

Figura 7: Comparação entre as soluções encontradas para duas posições objetivo diferentes: Em cinza escuro $\{-49, 100, 49\}$, em cinza claro $\{0, 100, 270\}$. Observe que no primeiro caso temos b=-0.56 e no segundo b=-0.81. Na figura representamos apenas o valor de 3 dos quatro ângulos das juntas, se projetamos sobre um outro conjunto de ângulos, qualitativamente o gráfico mantém o mesmo aspecto.

4 Conclusões

Comprovamos que o uso de Redes Neurais *Feedforward* MLP é inadequado para o tratamento deste problema, pois a relação que existe entre a posição e os ângulos θ_i não é uma função, ou seja, para um dado ponto (x, y, z) podem existir mais de um conjunto de ângulos θ_i adequado. As Redes Neurais acabam por "interpretar" esta relação como sendo uma função descontínua devido a forma aleatória como os pontos de treinamento foram escolhidos, levando a resultados com erros substanciais. Ironicamente esta mesma caracteritica é um facilitador para os Algoritmos Genéticos que conseguem convergir mais rapidamente nos casos onde temos um espaço de soluções com maior volume e converge mais lentamente quando este conjunto possui um menor volume.

Contudo o desempenho dos Algoritmos Genéticos é ainda insuficiente para seu uso em aplicações em tempo real. As simulações realizadas consomem da ordem de minutos para serem executadas (o valor depende dos parâmetros do Algoritmo Genético e do ponto que queremos atingir) o que para muitas aplicações práticas seria um sério impedimento. Este problema poderia ser mitigado com uma escolha mais criteriosa dos parâmetros do Algoritmo Genético, mas acreditamos que isto não implicaria em um ganho substancial de performance. Uma solução interessante poderia ser o uso de técnicas hibridas como por exemplo Raciocínio Baseado em Casos [16], neste caso a população seria inicializada com indivíduos próximos de uma solução. Por exemplo se queremos determinar uma trajetória para o braço robótico, formada por uma seqüência de pontos, poderíamos utilizar indivíduos do ponto anterior para auxiliar a convergência do algoritmo. Outra técnica poderia ser a combinação de Redes Neurais com Algoritmos de Otimização. Sabendo-se que a Rede Neural resulta em uma única solução aproximada entre as várias possíveis para esse caso, a mesma poderia ser utilizada como sugestão inicial para que o método de otimização possa tentar atingir o objetivo (erro $\cong$ 0). Essa abordagem poderia resolver o problema de velocidade de processamento do método de otimização, decorrente da varredura na superfície de solução, pois esta seria acelerada pela sugestão inicial da Rede Neural.

Acreditamos que estes resultados possam ser aplicados em outros contextos, onde é relativamente simples transformar um problema de aproximação de funções em um problema de minimização. Nos casos em que existirem múltiplas soluções adequadas, ou seja, múltiplos mínimos globais, o uso de Algoritmos Genéticos deverá atingir resultados satisfatórios. Quando a solução for única neste caso a Rede Neural deverá produzir bons resultados com um custo computacional sensivelmente menor.

Agradecimentos

Este trabalho foi parcialmente financiado pelo CNPq e pela Fapergs.

Referências

[1] C. D. Crane and Joseph Duffy, ***Kinematic Analisys of Robot Manipulators***, Cambridge University Press, 1998.

[2] O. Chocron and P. Bidaud, "**Evolutionary Algorithms in Kinamatic Design of Robotic Systems**," in *International Conference on Intelligent Robots and Systems*, vol. 2, pp. 1111–1117, Grenoble, France, September, 1997, citeseer.nj.nec.com/chocron97evolutionary.html.

[3] D. Halperin, L. Kavraki, and J. C. Latombe, "**Robot Algorithms**," in *CRC Handbook of Algorithms and Theory of Computation* (M. Atallah, ed.), ch. 21, Boca Raton, CRC Press, 1998, citeseer.nj.nec.com/halperin98robot.html.

[4] S. Haykin, ***Neural Networks: a compreensive foundation***. Inc., New Jersey: Prentice-Hall, 1999.

[5] J. H. Holland, ***Adaptation in natural and artificial systems***, The University of Michigan Press, Ann Arbor, 1975.

[6] A. A. D. Medeiros, ***Introdução à Robótica***, in *ENA-98 XVII Encontro Nacional de Automática*, pp. 56–65, Natal, RN, 1998.

[7] M. Mitchell, ***An introduction to genetic algorithms***, MIT Press, Cambridge, 1998.

[8] K. S. Narendra and K. Parthasarathy, "**Indentification and control of dynamical systems using neural networks**," *IEEE Transactions on Neural Networks*, vol. 1, 1990.

[9] K. Narendra and K. Parthasarathy, "**Gradient methods for the optimization on dynamical systems containing neural networks**," *IEEE Transactions on Neural Networks*, vol. 2, pp. 252, 1991.

[10] E. Oyama and S. Tachi, "**Modular neural net system for inverse kinematics learning**," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, p. 3239, San Francisco, CA, 2000.

[11] M. Riedmiller and H. Braun, "**A Direct Adaptive Method for Faster Backpro pagation Learning: The RPROP algorithm**," in *IEEE International Conference on Neural Networks*, pp. 586–591, San Francisco, CA, 1993, citeseer.nj.nec.com/riedmiller93direct.html.

[12] *http://www.robix.com*, "**Robix is a trademark of Advanced Design**," Inc., Tucson, Arizona, USA, last modified 2001-12-13.

[13] A. Rouvinen and H. Handroos, "**Robot Positioning of a Flexible Hydraulic Manipulator Utilizing Genetic Algorithm and Neural Networks**," in *Proceedings of the Fourth Annual Conference on Mechatronics and Machine Vision in Practice*, pp. 182–187, Toowoomba, Australia, 1997.

[14] A. Zell, M. Mamier, N. Mache, R. Hübner, S. Döring, K.-U. Herrmann, T. Soyez, M. Schmalzl, T. Sommer, A. Hatzigeorgiou, D. Posselt, T. Schreiner, B. Kett, and G. Wieland, "**Snns (stuttgart neural network simulator) user manual**," *http://www-ra.informatik.uni-tuebingen.de/SNNS/UserManual/UserManual.html*, Visitado em abril de 2002.

[15] A. Wagner, A. L. Cechin, F. J. Heinen, and A. C. S. Souto, "**Simulação de Manipuladores e Modelagem através de Redes Neurais**," in *X SEMINCO - FURB. Anais do X Seminário de Computação*, Blumenau, SC, 2001.

[16] Ian D. Watson, ***Applying Case Based Reasoning***, Morgan Kauffman, Tockholes, 1994.

[17] S. S. Yang, M. Moghavvemi, and J. D. Tolman, "**Modeling of robot inverse kinematics using two ann paradigms**," in *TENCON 2000, Proceedings*, vol. 2, p. 173, Kuala Lumpur, Malasya, 2000.

[18] A. Zakharov and S. Halász. "**Genetic Algorithms based Identification Method for a Robot Arm**". In *ISIE'99 - The 1999 IEEE International Symposium on Industrial Electronics*. Bled, Slovenia, July, 1999.