PÊNDULO INVERTIDO - Uma Abordagem Evolucionista

Daniel J. Gonçalves, Luis T. Baptista, Vasco Manquinho
Licenciatura em Engenharia Informática e de Computadores
Instituto Superior Técnico
Universidade Técnica de Lisboa - Portugal
e-mail:djvg@camoes.rnl.ist.utl.pt

Abstract: The balancing of an inverted pendulum is a well known problem in the control area. The methods normally used to solve it make use of prior knowledge of it's dynamics. The method that will be presented will try to solve that problem without the use of that knowledge. Additionally, while using a neural network, the only feedback it will get from the system is a signal indicating the failure or success of a simulation. To achieve this, an evolutionary approach will be used. A genetic algorithm will be used to train the network. Besides the single-pole problem, commonly solved by usual methods, an attempt of solving the double-pole problem will be made. In addition, a way of deriving the best topology for the networks will be presented.

1 - Introdução

O problema do pêndulo invertido é bem conhecido no âmbito do controlo por computador. É um problema clássico de controlo de sistemas dinâmicos, não-lineares, inerentemente instáveis. Várias soluções foram já desenvolvidas para esse problema. Essas soluções, no entanto, requerem normalmente que os sistemas desenvolvidos devam ter algum conhecimento *a priori* do domínio a ser controlado. O problema torna-se muito mais difícil de resolver quando o controlador não dispuser de qualquer informação sobre o mesmo.

Para conseguir esse tipo de controlo, o sistema deverá ter como entradas os resultados de diversas simulações sucessivas, a partir dos quais poderá aprender e melhorar o seu desempenho. Um modo de implementar tal abordagem é treinar uma rede neuronal que indique, a cada instante, qual o comportamento a tomar.

As redes neuronais são, normalmente, treinadas através de métodos computacionalmente pesados, sendo o mais conhecido deles o *backpropagation*, uma variante do método do gradiente. No entanto, os problemas de convergência bem conhecidos de tais algoritmos, e a já referida complexidade computacional são agravados em aplicações nas quais se pretende o desenvolvimento de controladores para sistemas dinâmicos não-lineares e instáveis [4]. Para esses sistemas é difícil obter a modelação matemática muitas vezes necessária para o treino das redes. Logo, não é descabido considerar outros métodos de treino da rede para o problema em causa.

Assim, consideraremos no nosso trabalho uma abordagem evolucionista, recorrendo aos algoritmos genéticos. Tal abordagem foi já usada com sucesso no treino de redes neuronais. Nela, evoluiremos os pesos da rede de forma a que esta se comporte de forma cada vez melhor para o problema em causa.

Outro factor que normalmente é determinante para o bom funcionamento de uma rede neuronal quando aplicada a um dado problema é a sua topologia. Não existem métodos definidos para a criação da topologia de uma rede neuronal. No entanto, recentemente, a abordagem evolucionista tem sido usada com bons resultados também nesta área [7]. Logo, embora o objectivo principal do trabalho por nós desenvolvido seja o equilíbrio do pêndulo, consideraremos, também, o uso dessa abordagem para o treino das redes.

2 - O Pêndulo Invertido

No problema que iremos considerar, o pêndulo invertido, o objectivo a atingir é o de equilibrar um pêndulo com uma extremidade fixa sobre um pequeno carro, em redor da qual pode oscilar (**figura 1**). Para equilibrar esse pêndulo, o carro pode deslocar-se horizontalmente, sendo esse o único modo de influenciar a posição do pêndulo. O movimento do carro encontra-se,

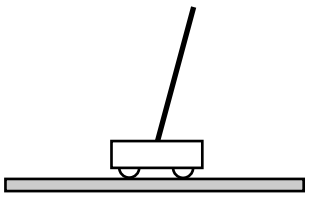

Figura 1: O Pêndulo Invertido

obviamente, constrangido pela largura do plano sobre o

qual se move, não lhe sendo permitido que exceda esses limites.

Em cada instante *t*, o sistema é caracterizado pelos seguintes valores: a posição e velocidade do carro (x_t e x_t') e a o ângulo entre o pêndulo e a vertical e a sua velocidade angular (θ_t e θ_t'). As unidades de x, θ e t serão, respectivamente, metros, radianos e segundos. Para além das grandezas acima referidas, também importa conhecer qual as massas do carro e do pêndulo (m_p), e o comprimento do pêndulo, em quilogramas e metros, respectivamente.

A modelação física das do sistema foi feita recorrendo às seguintes equações do movimento [5]:

$$\ddot{\theta}_t = \frac{mg\sin\theta_t - \cos\theta_t\left[F_t + m_p l\dot{\theta}_t^2\sin\theta_t\right]}{(4/3)ml - m_p l\cos^2\theta_t}$$

$$\ddot{x}_t = \left\{F_t + m_p l\left[\dot{\theta}_t^2\sin\theta_t - \ddot{\theta}_t\cos\theta_t\right]\right\}\Big/ m$$

Nas equações, *m* representa a massa total (carro + pêndulo), *g* a aceleração da gravidade ($9.8ms^{-2}$) e F_t a força exercida pelo carro num instante *t*. Como se pode depreender da análise das equações, o movimento do carro é influenciado pelo do pêndulo. A simulação do sistema físico foi feita recorrendo ao método de Euler, com um intervalo temporal $\tau = 0.02s$, sendo os valores actualizados ao longo do tempo com a seguinte relação: $\theta[t+1] = \theta[t] + \tau\dot{\theta}[t]$.

2.1 - O Problema dos Dois Pêndulos

Um problema análogo ao do pêndulo invertido mas de complexidade bastante superior é o do equilíbrio de dois pêndulos lado a lado sobre o mesmo carro (**figura 2**). Para além de dever ter em conta mais informação (sobre o ângulo e a velocidade angular do segundo pêndulo), o próprio equilíbrio é dificultado pelo facto de que ambos os pêndulos influenciam simultaneamente o comportamento do carro, podendo surgir situações em que, para equilibrar um dos pêndulos, se torna impossível equilibrar o outro.

3 - O Equilíbrio dos Pêndulos

3.1 - A População

Em concreto para o nosso problema, os indivíduos da população serão os pesos da rede neuronal que irá controlar o carro. Essa rede terá quatro ou seis neurónios de entrada consoante se esteja a considerar o equilíbrio de um único pêndulo ou de dois pêndulos (uma vez que os dados a considerar variam para cada um desses casos, como já foi descrito na secção 2). Para cada população, a

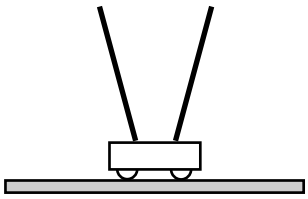

Figura 2: O problema dos dois pêndulos

topologia da rede manter-se-à constante. O valor obtido como saída da rede será, obviamente, a força que o carro irá exercer num dado instante.

Cada cromossoma será, portanto, constituído pelos pesos e bias da rede neuronal, sedo cada um desses valores considerado um gene.

3.2 - Os Operadores

Existem três operadores aplicáveis a estes indivíduos. Um é o cruzamento monoponto em que o ponto de corte é escolhido aleatoriamente entre dois genes. Desta forma os valores neles contidos são preservados para a geração seguinte, embora combinados de modos diferentes.

Outro é a mutação aleatória que consiste em somar a um gene um valor escolhido aleatoriamente com uma probabilidade uniforme num dado intervalo. O intervalo utilizado foi [-1, 1], para não provocar saltos muito grandes no espaço de estados..

Por fim, foi implementado um operador que consiste numa mutação probabilistica. Esta mutação soma também um valor ao gene. No entanto é mais elaborada. Assim para cada gene existe um parâmetro de estratégia (sigma) que determina o passo (soma) aplicado ao gene correspondente. Este parâmetro representa um desvio padrão que vai sendo adaptado à medida que o algoritmo genético avança. A forma como o gene e o sigma vão sendo adaptados é dada pelas formulas:

$$g'_i = g_i + \sigma_i \cdot N_i(0,1)$$
$$\sigma'_i = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1))$$

O índice i representa o gene i do cromossoma e o sigma correspondente. N(0,1) representa uma função aleatória com distribuição normal de média 0 e desvio padrão 1. Esta função foi descretizada entre -2,6 e 2,6. Obtemos, assim, valores aleatórios nesse intervalo com uma distribuição normal. O índice na normal significa que para cada mutação num gene existe um valor aleatório novo. Sem índice o valor é sempre o mesmo para todos os genes do cromossoma. Os factores τ' e τ são respectivamente iguais a $(\sqrt{(2.n)})^{-1}$ e a $(\sqrt{(2.\sqrt{n})})^{-1}$, em que o n representa o número de genes no cromossoma.

No inicio do algoritmo os pesos são inicializados com valores reais do intervalo [-0.5, 0.5], e os valores dos sigma com o valor de 5.0.

A motivação para a implementação de tal operador foi fazer com que, no início da evolução do sistema, dado

que os indivíduos da população se encontram potencialmente longe da solução, as variações de uma geração para a seguinte fossem grandes, de modo a que algum deles se aproxime da solução. No entanto, numa fase posterior da simulação, quando já se observa uma convergência apreciável, interessa evitar saltos grandes no espaço de estados. Logo, ao adaptar também os parâmetros σ, acima referidos, que controlam o desvio padrão da gaussiana, estamos a conseguir que o grau de mutação seja alterado ao longo da evolução do sistema na forma desejada.

3.3 - A Função de Mérito

Foram consideradas quatro funções de mérito diferentes. A primeira é a mais obvia para o problema em causa. Nela, apenas se considera como mérito dos elementos da população o número de iterações que decorreram até a simulação ter falhado (quer pelo carro ter excedido os limites de movimentação, quer devido à queda de um dos pêndulos).

Reconhecendo a excessiva simplicidade dessa função, uma outra foi implementada. Nessa, para além de considerar o número de iterações até à falha da simulação, beneficiam-se, também os indivíduos da população que se mantiverem mais perto do centro dos "carris". Com isto pretendem-se evitar situações em que o método utilizado para equilibrar os pêndulos seja, simplesmente, "perseguir" o pêndulo até exceder os limites dos "carris", o que pode fazer cair a população num máximo local, se muitos indivíduos adoptarem essa solução. O benefício decorrente da posição final do carro será escalado de forma a ser consistente com o número de iterações até à falha. Pretende-se, deste modo, descriminar correctamente entre os indivíduos no início da simulação, altura em que tal característica será determinante para guiar as gerações futuras em direcção à solução.

Finalmente, decidimos implementar duas funções que, embora análogas às anteriores, variam de forma exponencial, numa tentativa de mais facilmente discriminar entre elementos da população com méritos muito próximos. Essas funções serão da forma mérito = $\exp(\lambda x)$, em que λ é um parâmetro de controlo para influenciar o crescimento da exponencial e x representa o número de iterações até à falha ou esse número acrescido de um bónus proporcional à distância do centro.

4 - A Topologia das Redes

4.1 - Introdução

Como já se referiu anteriormente, embora não sendo esse o principal objectivo da investigação desenvolvida, tentou-se, também, implementar uma abordagem evolucionista para a criação das topologias das redes neuronais. Para tal, consideraram-se três tipos diferentes de indivíduos, representando abordagens diferentes ao problema, que serão descritos em seguida.

Para cada um dos tipos, a evolução processa-se de igual modo: após a criação, pelo algoritmo genético, de uma nova geração, o mérito de cada um dos seus elementos será calculado em função da facilidade com que essa rede conseguiu ser utilizada para achar a solução para o problema do pêndulo invertido (ou a distância a que ficou dessa solução).

Para tal, para cada topologia em causa, executa-se uma evolução do pêndulo invertido (como descrita na parte 3).

4.2 - Topologia Discreta

Neste tipo de representação de redes neuronais, existe o conceito de camada. Estas podem estar activas ou não. Por cada camada existe um número máximo de neurónios que estão activos. Diz-se que um neurónio está activo caso a camada a que pertence esteja activa e se existem ligações activas para ele de neurónios pertencentes à última camada activa.

Estabelecemos que as redes apenas terão, no máximo, três camadas escondidas. Estas deverão ser suficientes para aproximar a função de controlo e, ao mesmo tempo que manter o espaço de procura em dimensões aceitáveis. Temos também que definir qual o número máximo de neurónios por camada. Para o problema do pêndulo invertido, consideramos que seis é uma quantidade por demais suficiente.

As ligações entre os neurónios são representadas por um vector de bits. Para cada neurónio, um conjunto de bits (um bit por ligação) que define a que neurónios da camada seguinte é que está ligado. Este conjunto tem um tamanho fixo igual ao número máximo de neurónios por camada, excepto para os neurónios da última camada que apenas precisam de uma ligação para cada saída (no nosso caso, apenas uma).

Existe, ainda outro vector de bits para representar quais as camadas activas da rede. Caso a camada *n* não esteja activa, as ligações dos neurónios da camada *n-1* referem-se aos neurónios da camada *n+1*.

Para esta representação definimos os operadores de cruzamento e mutação. O primeiro operador efectua cruzamento monoponto para os vectores das ligações e das camadas. O operador de mutação modifica no máximo uma ligação e se uma camada está ou não activa. Esses operadores têm efeitos locais, isto é, não provocarão, em principio grandes saltos no espaço de procura.

4.3 - Topologia Probabilística

A abordagem probabilistica para a optimização da topologia da rede neuronal utiliza uma codificação em termos de probabilidades para representar as ligações entre neurónios de diferentes camadas. Para cada neurónio, esse valore representa a probabilidade de esse neurónio estar ligado a cada um dos neurónios da camada seguinte.

O código genético dos elementos desta população é constituído por uma sequência de bits com a seguinte interpretação: os primeiros **p** bits representam a

probabilidade de ligação dos neurónios de entrada aos da primeira camada. Em seguida para cada camada existem **c** bits para o número de neurónios nessa camada e **p** bits para a probabilidade de ligação dessa camada para a seguinte. Cada conjunto de bits representa um gene. Utilizamos 8 bits para a probabilidade e 3 bits para o número de neurónios, com o que conseguimos obter, no máximo, 7 neurónios por camada. O número de camadas reflecte o tamanho do cromossoma. Neste caso utilizamos 3 camadas escondidas. Se uma camada tiver 0 neurónios essa camada fica desactivada.

Criámos para estes indivíduos da população um operador de cruzamento e outro de mutação. O primeiro implementa um cruzamento uniforme em que os pontos de corte são entre os genes, não destruindo assim os genes. O segundo soma um determinado valor inteiro aos genes. Para os genes da probabilidade de ligação os valores possíveis são retirados aleatoriamente do intervalo [-10, 10] e para os genes do número de neurónios do intervalo [-2, 2].

Com este tipo de população tentamos analisar o efeito de não constringir em demasia a evolução da topologia das redes. Em vez de evoluir a topologia de forma concreta, tentaremos evoluir um conjunto de topologias com potencial para aprender.

4.4 - Sistemas-L

Os sistemas-L, originalmente utilizados na simulação do crescimento biológico de plantas, são baseados em regras sequencialmente aplicadas a um axioma inicial. O resultado da aplicação das regras de produção pode ser interpretado de acordo com a semântica dos símbolos utilizados. Com o uso dos sistemas-L para a geração de arquitecturas de redes neuronais espera-se a obtenção de módulos na arquitectura da rede, o que será útil, tendo em conta os princípios de processamento neuronal.

O alfabeto por nós escolhido foi o seguinte conjunto de símbolos: { A-G, 0-2, [,] }. Cada letra do alfabeto representa um nó da rede e pode ser vista como sendo o menor modulo possível. Se as letras estiverem agrupadas dentro de parêntesis definem um modulo que pode, por sua vez, estar embebido dentro de outros módulos. Quando um digito está a seguir a um modulo, indica a que modulo é que está ligado, ou seja, quando é encontrado um digito *x*, o modulo que o precede está ligado ao modulo a que se pode chegar ao saltar *x* módulos. O alcance do salto ser exterior ao modulo onde o digito está inserido.

Neste tipo de codificação algumas restrições foram impostas. Por exemplo, o maior salto que uma ligação pode indicar é de 2. Estamos, assim, a restringir as redes que é possível gerar. Como o problema que temos em mãos não exige redes muitíssimo elaboradas, esta solução é perfeitamente aceitável, diminuindo o espaço de procura sem afectar a qualidade da solução final.

Os sistemas utilizados são na realidade sistemas-2L, dado que cada regra de produção possui um contexto anterior e posterior. Assim, as regras utilizadas têm o seguinte formato: $L < P > R \rightarrow S$.

Nos sistemas-2L convencionais, os contextos são emparelhados directamente na *string*. No nosso caso, a regra só dispara se houver um modulo que emparelhe com o contexto L e que esteja ligado ao predecessor P. É ainda necessário que o predecessor P esteja ligado ao contexto R.

Suponha-se, por exemplo, que o sistema possui as regras da **tabela 1** e que o axioma inicial é A:

1:		<	A	>		→	B0[CD]0F
2:		<	B	>	D	→	C
3:		<	C	>	C	→	E
4:	C	<	C	>		→	C1
5:	E	<	D	>	F	→	B0C

Tabela 1

Aplicando as regras temos assim que a string final será E0[CB0C]0F, o que, após tradução, dá origem à rede da **figura 3**.

Ao utilizar este mecanismo é necessário ter alguns cuidados, como por exemplo, evitar ciclos infinitos causados por regras do tipo $A \rightarrow AA$. Este problema é evitado pela restrição que fazemos de que o número máximo de regras a aplicar é 8. No entanto, se não puséssemos outras restrições, num sistema com a regra referida, esta seria aplicada o numero máximo de vezes. Assim, para evitar que seja sempre a mesma regra a ser disparada, introduzimos a restrição de que a mesma regra não pode ser disparada mais do que duas vezes consecutivas.

Dado que não é possível estabelecer à partida o número de entradas ou saídas da rede retornada pelo sistema-L, consideramos esta rede representa apenas as camadas intermédias. Assim, assumimos que todas as entradas ligam às entradas da rede do sistema-L e que

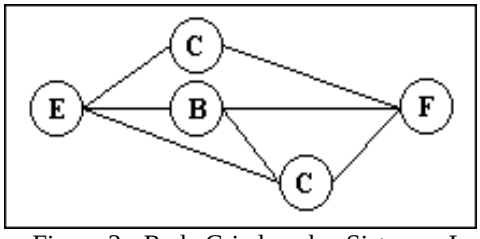

Figura 3 - Rede Criada pelos Sistemas-L

todos os nós de saída desta rede estão ligados ao nó de saída do nosso problema.

Os símbolos do alfabeto são codificados num vector de bits onde cada símbolo ocupa 6 posições. Após o vector ser descodificado, é criada uma string com os símbolos do nosso alfabeto e com o símbolo * que delimita os componentes de uma regra. A sequência de símbolos é de seguida interpretada, dando origem às regras do sistema-L. Os componentes das regras são lidos sequencialmente da string.

Por exemplo, a partir da string B*A**C*, retiramos a regra B < A → C, onde se pode notar que o contexto da direita é, neste caso, vazio.

Para obter muitas regras sem utilizar um vector de grandes dimensões, esse vector de é lido diversas vezes, primeiramente a partir da posição 0, de seguida a partir da posição 1, etc. O vector é também lido da direita para a esquerda. Assim, tendo um vector com, por exemplo 60 bits, conseguimos obter em cada leitura dez símbolos do alfabeto. Como fazemos seis leituras a partir da esquerda e seis leituras a partir da direita, temos que um vector com 60 bits é mapeado para uma string com 120 símbolos.

O operador de cruzamento é monoponto. Este operador escolhe, segundo a probabilidade do operador, o ponto de cruzamento dos cromossomas. O operador de mutação apenas modifica, no máximo, um bit no cromossoma. Estes operadores são bastante simples, mas se pensarmos bem, dificilmente outros mais complexos poderiam ser adequados. Relativamente ao operador de mutação, não podemos modificar mais do que um bit, dado que potencialmente iríamos modificar um número elevado de símbolos do alfabeto ao efectuarmos diversas leituras a partir do mesmo vector de bits. Caso a mutação seja mais abrangente poderá fazer com que o novo elemento da população não tenha muitas semelhanças com o original. O mesmo se passa em relação ao operador de cruzamento que, se for uniforme, pode dar como resultado novos indivíduos sem qualquer relação com os pais. Ao ter apenas cruzamento monoponto, estamos a indicar que parte das regras dos dois pais são passadas para os filhos. Isto muito dificilmente poderia acontecer em outros tipos de cruzamento.

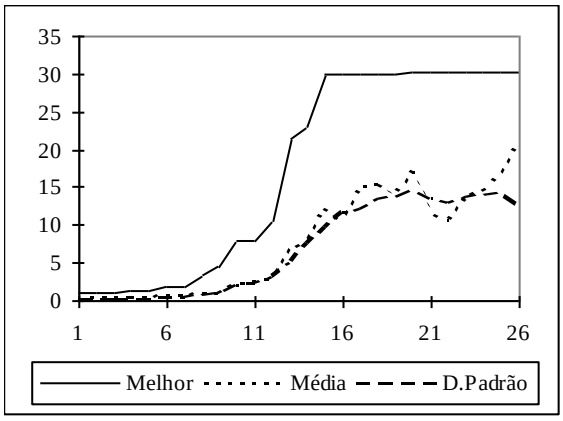

Gráfico 1 - Evolução da População

5 - Resultados Obtidos

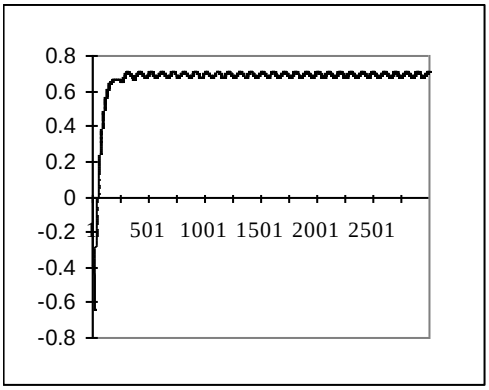

Gráfico 3 - Posição do Carro

5.1 - Equilíbrio dos Pêndulos

5.1.1 - Um Pêndulo

O problema de equilíbrio de um pêndulo revelou-se de fácil resolução. Foi usada uma rede com uma camada escondida com seis neurónios, *fully-connected*. De modo a obter alguns resultados significativos, foram feitos testes para os quais se considerou se a massa do pêndulo como 1,6 Kg e o seu comprimento como 2 metros. O ângulo inicial do pêndulo é aleatório para cada simulação (de modo a manter a generalidade da solução obtida) e nunca se permite que a força produzida pelo carro seja zero, obrigando-o a evitar situações nas quais se limita a ficar parado. Para essas condições, uma evolução típica da população pode ser observada no **gráfico 1**. Como se pode constatar, após um período inicial de lenta convergência, segue-se um acréscimo bastante significativo do mérito do melhor elemento da população (recorrendo ao elitismo), que quase imediatamente atinge a solução do problema. Segue-se-lhe, em seguida, a média para toda a população. O desvio padrão mantém-se inicialmente elevado, indiciando a existência de alguns elementos da população ainda com valores bastante baixos, diminuindo à medida que a média sobe. Acima de um dado valor da média (que continua a subir para além do número de gerações mostradas no gráfico), o desvio padrão começa, mesmo, a descer de forma acentuada.

Em média, o primeiro indivíduo que resolve o problema surge por volta da vigésima-quinta geração.

Constatamos que o problema é bastante dependente das condições iniciais para o achar da solução do problema. Essa dependência foi minimizada escolhendo criteriosamente a função de mérito mais adequada para o problema. Na maior parte dos casos, uma convergência lenta correspondia a uma população em que a maioria dos indivíduos pouco se preocupava com a posição do carro, acabando por exceder os limites a ela impostos. De facto, a primeira função de mérito usada, linear, apenas tinha em conta o número de iterações até à falha. Esse valor era quase idêntico para todos os elementos da população, não facilitando a discriminação entre eles. Quando substituímos essa função pela que beneficia os indivíduos que, no final da sua simulação, mais se aproximem do centro dos "carris", os resultados melhoraram imediatamente, aumentando a taxa de sucesso de 55% para 70%.

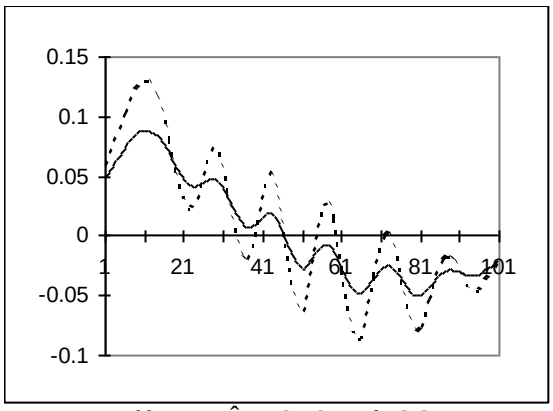

Gráfico 2 - Ângulo dos Pêndulos

Quando foi efectuada a verificação das funções de mérito de natureza exponencial, os valores acima referidos melhoraram ainda mais. Para o caso simples, ignorando a posição do carro, a função exponencial mostrou-se pouco melhor que a linear. Quando aliada à posição do carro, no entanto, conseguiram-se resolver sem dificuldades 80% dos problemas. Para além disso, a convergência deu-se de forma mais rápida e eficiente, dado que a média da população subia mais acentuadamente.

Pode, pois, concluir-se, que a função de mérito exponencial, aliada ao conhecimento da posição do carro, se revelou superior às restantes, discriminando melhor entre indivíduos similares, principalmente no início do programa, em que tal distinção é bastante importante.

Finalmente, no respeitante aos operadores, concluímos que, de entre os dois tipos de mutação, a probabilística se mostrou superior à aleatória. De facto, a evolução do mérito usando a mutação probabilística é feita de forma bastante mais suave do que usando a mutação aleatória. Não se verificam, por exemplo, oscilações de grande amplitude nos méritos que ocorrem frequentemente usando a mutação aleatória. Além disso, a solução foi encontrada mais vezes recorrendo à mutação aleatória.

Verificou-se, também, que a probabilidade de mutação deve rondar os 40%, pelo menos no início do programa. Taxas maiores levam a um dispersar dos elementos da população, e taxas menores não são suficientes para favorecer uma rápida evolução da mesma.

Quanto ao cruzamento, este demonstrou ter uma importância mais reduzida do que a mutação, para o encontrar da solução. Embora, normalmente, o cruzamento torne a convergência mais simples, não devem, no entanto, ser usadas probabilidades de cruzamento muito altas dado que nesse caso a convergência começa a ser mais difícil (talvez devido a não fazer muito sentido trocar arbitrariamente os pesos de duas redes neuronais, que se encontram, de alguma forma, relacionados com os restantes, definindo uma dada função implicitamente, que pode, assim, perder todo o seu significado). Valores perto de 80% parecem adequados.

5.1.2 - Dois Pêndulos

Para dois pêndulos o problema foi, como já se previa, bastante mais difícil de resolver. A convergência dá-se de forma muito mais lenta do que para um único pêndulo. Para pêndulos com massas e comprimentos diferentes, as forças a aplicar pelo carro devem ser tais que embora suficientemente grandes para não deixar um dos pêndulos cair numa dada direcção, não sejam suficientemente grandes para tombar o outro pêndulo na direcção oposta.

Para testar este problema, foi usada uma rede com uma camada escondida com oito neurónios, *fully-connected* e partiu-se de um estado inicial em que o primeiro pêndulo tem massa de 0,1 kg e comprimento de 0,5 m. As grandezas que definem o segundo pêndulo são dez vezes menores que as suas análogas para o primeiro. Após obter convergência para essa situação, o tamanho do pêndulo menor é aumentado 0,01 e a sua massa 0,05, e assim sucessivamente. Garante-se, também, no início, que

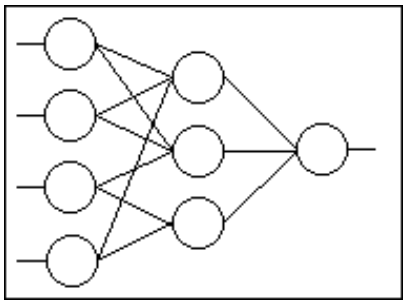

Figura 4 - Topologia Típica

os ângulos iniciais dos dois pêndulos fazem com que estes estejam inclinados para o mesmo lado, de modo a evitar que a diferença entre os ângulos seja muito grande, o que poderia tornar a solução inalcansavel.

Desse modo, a convergência para a situação inicial (e usando os operadores e funções de mérito já optimizados anteriormente, como foi explicado na sub-secção anterior), obteve-se, em média, o equilíbrio de ambos os pêndulos por volta da 700ª geração. Após esse primeiro equilíbrio, cada equilíbrio adicional torna-se bastante mais simples, bastando para tal, em média, 10 gerações.

Nos gráficos 2 e 3 podem ser vistas as evoluções dos ângulos dos pêndulos e da posição do carro, respectivamente.

5.2 - Criação das Topologias da Rede

Os testes para criar as topologias da rede foram efectuados realizando simulações da evolução dos pesos da rede durante 30 gerações.

No respeitante à evolução das topologias da rede, todos os métodos foram unânimes: as redes mais simples são as melhores. Após vários testes com os diversos métodos, todos eles convergiram, passadas algumas gerações, para redes com uma ou duas camadas escondidas, com poucos neurónios, como a representada na **figura 4**.

A escolha de esse tipo de redes pode dever-se ao facto de que, nelas, o espaço de estados é de dimensões

bastante mais reduzidas e de menor complexidade. Isto faz que existam, potencialmente, menos máximos locais onde a população pode ficar atolada. Mostrou-se, também, crucial, que todos os neurónios de entrada influenciem, de algum modo, a saída.

De entre os três métodos, a evolução discreta obteve os melhores resultados. Nela, a evolução dá-se de forma mais ou menos monótona, encontrando-se redes que equilibram facilmente o pêndulo logo nas primeiras gerações.

Em seguida, o melhor comportamento foi revelado pelos Sistemas-L. Embora a diversidade das redes obtidas não fosse tão grande, as redes por ele geradas são, de um modo geral, pequenas, e *fully-connected*, ou perto disso. Como já se referiu, redes em que todas as entradas influenciem a saída são as melhores para o controlo do pêndulo invertido. Logo, as redes *fully-connected* são candidatos ideais (o que confirmou as nossas suspeitas iniciais). Aliás, para todos os métodos a convergência começa a dar-se a partir da altura em que vários elementos da população se aproximam desse estado.

Com o pior comportamento revelou-se a abordagem probabilística. Nessa abordagem, embora, por vezes, as redes obtidas fossem de boa qualidade, ao não estarem fixadas as ligações um indivíduo que deu origem a uma rede boa pode não o voltar a fazer. A convergência torna-se, portanto, bastante mais lenta.

No respeitante à função de mérito mais adequada para este problema, a sua escolha não foi trivial. De facto, não é invulgar (principalmente nos Sistemas-L, em que a diversidade é menor), encontrar na primeira ou segunda gerações redes que resolvam o problema. No entanto, isto deve-se apenas ao facto do elitismo favorecer essa situação. A média da população dos carros, para essas redes, raramente atinge valores elevados. Daí se pode concluir que um valor mais indicado para determinar o mérito de uma dada topologia é o mérito médio da população dos carros, e não o seu melhor valor.

Consideramos, então, como função de mérito, foi da forma: ln (média da população). O logaritmo foi usado de modo a facilitar a convergência do algoritmo, distinguindo-se de forma mais fidedigna os vários elementos da população. De entre as várias funções experimentadas, essa foi a que nos forneceu os melhores resultados.

6 - Conclusões

Do estudo efectuado podemos concluir que os algoritmos genéticos são uma ferramenta poderosa para a resolução de problemas de difícil resolução usando as usuais ferramentas analíticas e numéricas. Ao tratar-se de um método de aprendizagem não simbólico, não necessita ter uma representação explicita do problema a resolver. Conseguimos, desse modo, resolver um problema, o dos dois pêndulos, que não é facilmente resolúvel de outro modo. A maior dificuldade encontrada foi a escolha de operadores adequados, dada a falta de sensibilidade sobre o efeito de alterações na rede. No entanto, os operadores por nós considerados obtiveram bons resultados.

O treino das redes neuronais decorreu de forma correcta, verificando-se a adequação dos algoritmos genéticos para esse fim. Também as topologias para as redes neuronais encontradas recorrendo à abordagem evolucionista foram de boa qualidade. Esse é um dos principais problemas encontrados em aplicações de redes neuronais, pelo que essa abordagem parece promissora para futura investigação.

7 - Bibliografia

[1] Hiddleton, Richard e Goodwin, Graham; *Digital Control and Estimation, an unified approach*; Prentice-Hall International Editions, 1990 USA. pp. 436 - 437.

[2] Elgard, Olle; *Control Systems Theory*; McGraw-Hill, 1967 Tokyo - Japão. pp. 9 - 21.

[3] Ogata, Katsuhiko; *Modern Control Engeneering*; Prentice-Hall, 1970 USA. pp. 278 - 279.

[4] Saravanan, H. e Fogel, David. *Evolving Neural Control Systems*. *In* IEEE Expert - Intelligent Systems & theyr Applications; vol.10, num. 3. pp. 23 - 27.

[5] Anderson, Charles. *Learning to Control an Inverted Pendulum Using Neural Networks*. *In* IEEE Control Systems Magazine, vol. 9, num 3. pp. 31-33.

[6] Boers, Egbert, *et al*. *Designing Modular Artificial Neural Networks*. Departamento de Informatica da Universidade de Leiden - Holanda, Setembro de 1993.

[7] Sims, Karl. *Evolving 3D Morphology and Behaviour by Competition*, *In* Rodney Brooks e Pattie Maes (eds.), Artificial Life IV; MIT Press 1994 , pp. 28 - 39.