SISTEMAS CLASSIFICADORES COM REDES NEURAIS (NNCS) : APLICAÇÃO AO CONTROLE DE UM VEÍCULO AUTÔNOMO SIMULADO COMPUTACIONALMENTE.

LUBNEN N. MOUSSI, RICARDO R. GUDWIN, FERNANDO J. VON ZUBEN, MARCONI K. MADRID

DSCE/DCA – FEEC - UNICAMP
Av. Albert Einstein 400 Bloco A – Barão Geraldo, Campinas - SP
E-mails: lubnen@dsce.fee.unicamp.br , gudwin@dca.fee.unicamp.br,
vonzuben@dca.fee.unicamp.br , madrid@dsce.fee.unicamp.br

Resumo: Neste trabalho desenvolvemos um experimento relacionado a um sistema classificador usando redes neurais. A principal pergunta que visamos responder é a seguinte: podem as redes neurais, na condição de aproximadoras universais, enriquecer o desempenho dos sistemas classificadores? Este trabalho representa uma primeira incursão na tentativa de se responder essa questão, por meio da aplicação de um sistema classificador deste tipo no controle de um veículo autônomo, em ambiente simulado computacionalmente. Os resultados obtidos constatam que existem grandes possibilidades de que esse hibridismo possa melhorar o desempenho dos sistemas classificadores.

Abstract: In this paper we develop an experiment related to a classifier system using neural networks. The main question we aim to answer is to determine whether neural networks, as universal approximators, can enrich classifier systems performance. This work presents a first incursion trying to solve that question, applying such classifier system to the control of an autonomous vehicle in a computational environment. The results we got showed that there exist great possibilities that this hybridism could enhance classifier systems performance.

Key Words: Classifier systems, evolutionary computation, neural networks, autonomous navigation.

1 Introdução

Sistemas classificadores (CS) [Booker et. al., 1989] são sistemas baseados em regras proposicionais. Diversos autores já demonstraram a ineficácia do uso de regras proposicionais no desenvolvimento de sistemas inteligentes [Simon 1996]. Apesar disso os sistemas classificadores, como será apresentado neste trabalho, possuem qualidades que poucos sistemas inteligentes demonstraram até agora.

Algumas perguntas se fazem prementes. Será possível evitar o uso de regras proposicionais e ainda assim aproveitar todas as vantagens que estes sistemas carregam consigo? Podem as redes neurais substituir as regras proposicionais dos sistemas classificadores? No ensejo de responder a essas perguntas de maneira pragmática, duas fases foram concebidas. Primeira, desenvolver um protótipo de um sistema classificador híbrido (NNCS), que substitua o uso de regras proposicionais por redes neurais. Esse protótipo deve ser testado em alguma aplicação prática que permita a comparação com um sistema classificador convencional. Em seguida, dependendo dos resultados, é possível proceder a uma análise mais generalizada desta nova versão de sistemas classificadores, procurando avaliar o seu escopo de aplicação e também aspectos computacionais envolvidos. O presente trabalho documenta a execução da primeira fase desta estratégia e levanta questões relevantes para o encaminhamento da segunda.

A aplicação prática concebida foi o controle de um veículo autônomo, em ambiente simulado computacionalmente. Os resultados obtidos foram promissores, o que nos encorajou a divulgá-los, antes mesmo que a segunda fase pudesse ser concluída. O tema é inédito na literatura, pelo menos até onde conhecemos. Existem, é claro, algumas aplicações que poderiam ser consideradas semelhantes, como por exemplo o ERL (Evolutionary Reinforcement Learning) [Ackley & Littman, 1991], embora essas semelhanças sejam de natureza local. Na verdade, estuda-se a possibilidade de virmos a aplicar algumas das idéias de Akcley e Littman em desenvolvimentos posteriores do NNCS.

Na seqüência deste artigo, apresentamos os conceitos básicos de um sistema classificador convencional para, a seguir, mostrar a implementação de sua versão com Redes Neurais (NNCS). Depois, discutimos os detalhes da implementação prática realizada, as simulações efetuadas, conclusões e considerações relevantes para a próxima fase.

2 Sistemas Classificadores

Existem poucas referências a sistemas classificadores na literatura. Algumas das referências mais importantes são o livro que lançou a proposta [Holland, 1975], o artigo de Booker [Booker et al., 1989], o livro de Goldberg [Goldberg, 1989] e a tese de Richards [Richards, 1995], que traz uma descrição bem detalhada sobre sistemas classificadores.

Dentre outras vantagens, os sistemas classificadores possuem as seguintes características: são indicados para operar em ambientes que tipicamente exibem eventos novos e sucessivos acompanhados de ruído e dados irrelevantes; são apropriados em situações em que haja a necessidade de agir de maneira contínua e freqüentemente em tempo real; possuem a habilidade de se guiar por metas implícitas ou inexatas, bem como promovem o aprendizado baseado em recompensas esparsas.

Essas características são possíveis porque os sistemas classificadores descobrem novas categorias

e conceitos por meio de regularidades encontradas no ambiente, uma vez que estas sejam relevantes para o atendimento das metas desejadas. Do mesmo modo, os sistemas classificadores utilizam o fluxo de informação encontrado no caminho até a meta para refinar seu modelo do ambiente e desta forma associar ações de controle apropriadas às situações encontradas na busca.

2.1 Arquitetura do Sistema Classificador

Os sistemas classificadores funcionam por meio de uma arquitetura hierárquica, onde o nível mais baixo é apresentado na Fig. 1.

Este nível da arquitetura funciona da seguinte forma. A interface de entrada coleta informações do ambiente através de sensores e as codifica em forma de mensagens que serão colocadas na Lista de Mensagens (LM). Observe que não somente a interface de entrada envia mensagens para a LM, mas também a Lista de Classificadores (LC). Uma mensagem na LM é um arranjo de caracteres composto de duas partes: os caracteres iniciais identificam quem enviou a mensagem e os seguintes identificam ocorrências no ambiente, detectadas pelos sensores. Os exemplos de mensagem a seguir correspondem a situações arbitrárias extraídas do ambiente da aplicação.

1	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**0**

Os dois primeiros bits **"10"** identificam a Interface de Entrada como a autora da mensagem. Os 9 bits seguintes descrevem a existência de obstáculos no campo de visão do Veículo, cada bit correspondendo a uma região. Bit 0 corresponde a não haver obstáculo, bit 1 indica a existência de obstáculo.

A LC é composta por Classificadores (regras), constituídos de um arranjo de caracteres dividido em 3 partes. Na primeira, temos os caracteres que identificam o tipo de *conseqüente* (será visto adiante) do Classificador. Na segunda, temos caracteres que constituem a sua *condição* (ou *antecedente*). Na terceira, estão os que correspondem à *ação* (ou *conseqüente*) do classificador, que é a mensagem que será enviada à LM caso o classificador seja ativado.

0	**1**	**#**	**1**	**#**	**0**	**#**	**1**	**0**	**#**	**#**	**0**	**1**

No exemplo acima, os dois primeiros bits, por uma convenção arbitrária, indicam que o conseqüente é uma mensagem para a interface de saída. Os 9 bits seguintes correspondem à condição, onde nota-se a presença do caracter "#", significando "*don't care*". Se a condição de um classificador for igual à mensagem da interface de entrada, ocorreu um "*matching*". Os símbolos # valem 0 ou 1, favorecendo o *matching*. Na terceira parte, o conseqüente do Classificador, pode ser de dois tipos: uma mensagem que será usada pelos classificadores no próximo ciclo, ou uma mensagem para a interface de saída providenciar uma ação determinada.

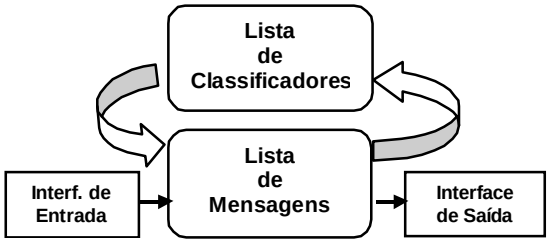

Figura 1: Sistema Classificador Básico

Os Classificadores que apresentarem *matching* com alguma(s) mensagem(ns) da LM, têm chance de postar novas mensagens. Seguindo o exemplo, temos a mensagem postada:

0	**1**	**0**	**1**

Essa mensagem é então enviada à LM. A Lista de Mensagens recebe mensagens da Interface de Entrada e da Lista de Classificadores, estas podendo ser de dois tipos: novas mensagens para os classificadores ou mensagens para a Interface de Saída. A interface de saída fica constantemente monitorando a LM, e uma vez que verifique a existência de mensagens para si, recolhe-as da lista, executando as instruções que nela estiverem determinadas. Além disso, a interface de saída pode resolver possíveis impasses oriundos de mensagens conflitantes.

Esse loop inicial em nada difere de um mecanismo de base de regras proposicionais. Ele pressupõe que a LC represente um conjunto de regras que possua o conhecimento necessário para processar as informações vindas do ambiente. Entretanto, ao contrário de uma base de regras usual, onde o conhecimento é adquirido de um especialista, nesta técnica as regras (*classificadores*) na LC serão introduzidas por meio de um mecanismo evolutivo.

Os classificadores são criados inicialmente de forma aleatória. Então, para que possam ter um comportamento adequado, será necessário que o CS descubra quais classificadores geram comportamentos apropriados, crie novos classificadores e descarte aqueles que sejam ou tornem-se inapropriados, considerando a variabilidade do ambiente. Esse mecanismo de aprendizagem de regras é implementado pelo algoritmo de *Atribuição de Créditos* e por um *Algoritmo Genético* (*AG*), que atuam nos níveis superiores da hierarquia em que se estrutura o sistema de classificadores.

2.2 Atribuição de Créditos

Para cada classificador é estipulado um valor de "*strength*", que é a medida da confiança em um classificador perante uma situação. Os classificadores que fazem "*matching*" com alguma mensagem na lista de mensagens terão que fazer uma *aposta* ou "*bid*" para obtê-la e aquele que oferecer maior lance terá maior probabilidade de postar sua mensagem. O quadro a seguir ilustra como isso é implementado.

Aposta do classificador C:
B(C,t) = b.R(C).s(C,t)
R(C) - especificidade da condição do classificador (tamanho da condição menos nr. de símbolos #, dividido pelo tamanho da condição).
b - constante menor que 1 (por exemplo, 1/8 ou 1/16)
s(C,t) - *strength* do classificador C no instante t.
B - valor do *bid*, aposta, determina a probabilidade de que o classificador poste sua mensagem
Strength do classificador C:
s(C,t) = s(C,t) - B(C,t)
Recompensa: Se a ação gerada pelo classificador for apropriada, ele receberá a sua aposta de volta, adicionada de uma recompensa R:
s(C,t+1) = s(C,t) + B(C,t) + R

Um dos algoritmos mais conceituados para a implementação da Atribuição de Créditos é o chamado algoritmo "*Bucket Brigade*" [Booker et. al., 1989]. Utiliza a idéia de que os classificadores podem postar mensagens que irão gerar novas mensagens, formando um encadeamento até gerar uma ação e, por isso, todas as mensagens envolvidas na ação devem ter seu crédito de volta mais uma recompensa se a ação foi bem sucedida, ou nada em caso contrário. Neste trabalho, não usaremos o *Bucket Brigade*, mas um algoritmo alternativo baseado nos mesmos princípios.

2.3 Algoritmo Genético (AG)

O processo de descoberta de regras em sistemas classificadores utiliza um algoritmo genético.

Basicamente, o AG seleciona os classificadores com os maiores *strengths* como pais, gerando novos indivíduos por meio da recombinação e mutação destes. Os novos classificadores gerados tomam o lugar dos mais fracos, modificando o conjunto de classificadores do sistema [Holland, 1975].

3 A Aplicação e seu Ambiente

O Veículo Móvel, ver Fig. 2, tem forma circular, com diâmetro de 50 pixels e campo de visão dividido em 9 regiões. Seu objetivo é não colidir com nenhum obstáculo. O Ambiente é um reticulado de 700x700 pixels, onde as marcas circulares representam pilastras adimensionais (diâmetro zero), ver Fig. 3.

4 NNCS – O Sistema Classificador com Redes Neurais.

4.1 Por que usar Redes Neurais

A inclusão de redes neurais nos CS, para substituir algumas de suas funções, tem implicações referentes às suas características de aproximadoras universais e do uso, quase que conseqüente, de números reais. Ou

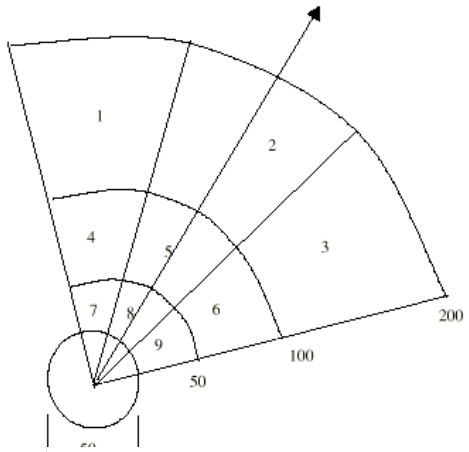

Figura 2: Veículo e Campo Visual

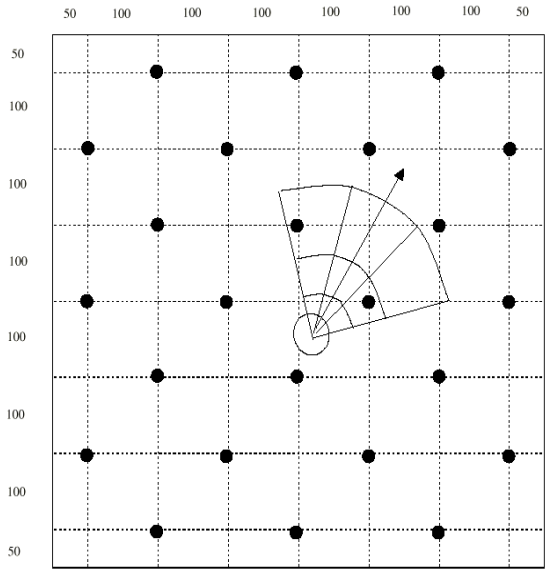

Figura 3: Ambiente de Simulacão

seja, não é uma mera substituição de métodos numéricos e é natural supor que Redes Neurais têm chance de enriquecer o desempenho dos CS. Espera-se que, com sua inclusão, seja possível observarmos comportamentos mais sofisticados do que os normalmente conseguidos sem elas.

Com relação ao tipo de rede neural, utilizamos redes Perceptron [Haykin 1999], sem um critério específico para sua escolha

4.2 O que as Redes Neurais irão substituir

Cada classificador é um arranjo que codifica os pesos de duas redes neurais: uma para avaliação e outra para ação (conforme a figura 4). A rede neural correspondente à a*valiação* substitui o *matching* no CS convencional. Seu papel será determinar o grau de interesse do classificador em uma mensagem qualquer colocada na LM. A rede neural correspondente à ação gerará uma nova mensagem que será postada na LM. Ambas as redes processam a mesma entrada, vinda da interface de entrada. O restante do mecanismo é exatamente o mesmo que no CS normal.

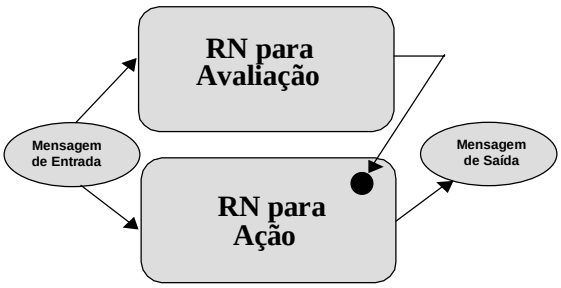

Figura 4: RN para Avaliação e para Ação.

4.3 Configuração do NNCS

Utilizamos Redes Neurais do tipo Perceptron com uma camada intermediária de dois neurônios. A camada de saída da rede de *Avaliação* tem apenas um neurônio, que fornece o nível de especificidade. A camada de saída da rede de *Ação* tem um ou dois neurônios, dependendo de considerar-se apenas a orientação do veículo como variável, ou também a velocidade. A função de ativação da camada intermediária é a tangente hiperbólica, e a camada de saída tem ativação linear.

A entrada da rede é a mensagem da Interface de Entrada mais polarização. Cada mensagem é um *arranjo* com 9 caracteres dos tipos 0 e 1. Assim, os classificadores têm o antecedente com 23 parâmetros reais (pesos da Rede Neural de Avaliação), e têm o conseqüente composto de 23 ou 26 parâmetros reais (pesos de uma Rede Neural de Ação). Serão também, em alguns testes, considerados como entrada os valores de velocidade e orientação do veículo, acrescentando mais duas posições reais ao arranjo de entrada.

O AG utiliza crossover de dois pontos. Como são números reais, para conseguir uma maior faixa de valores, é feita a soma, diferença ou média, dos valores entre os pontos. É usada mutação inversiva para os testes iniciais, porém a mutação indutiva deve permitir maior diversidade para os valores.

O AG seleciona os ‘n‘ melhores classificadores e aplica o algoritmo *Roullete Wheel* [Goldberg, 1989] para formar os pares. São utilizados dois critérios para disparar o AG: número de iterações, ou número de colisões, aquele que ocorrer primeiro:

- **Número de colisões:** muitas colisões podem indicar insuficiência de classificadores adequados, o que seria resolvido acionando-se o AG.
- **Número de iterações:** se não há colisões não significa que o comportamento não possa ser melhorado.

É importante observar que a configuração escolhida para o primeiro protótipo do NNCS é a mais simples possível, desenvolvida somente com o propósito de verificarmos a exeqüibilidade do sistema, sem a consideração de nenhum requisito ligado a desempenho.

5 Desenvolvimento

Para verificarmos a funcionalidade do NNCS consideramos que o veículo terá o objetivo de não colidir. Desta forma, o objetivo será atendido se, após uma fase de aprendizado, o veículo conseguir move-se sem colidir por um número grande de iterações. Esse comportamento normalmente será caracterizado pelo movimento do veículo descrevendo órbitas de tipos bastante variados, envolvendo uma ou mais pilastras ou mesmo no interior de quatro delas, Ver fig 5, onde as linhas retas mostram o veículo sendo colocado de volta ao centro após uma colisão.

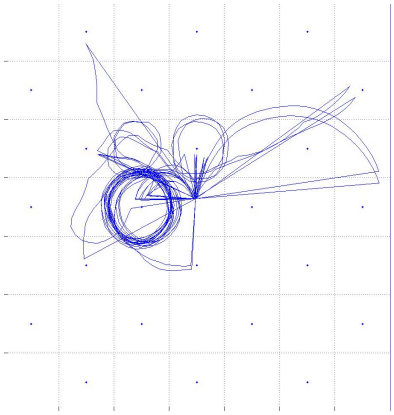
Figura 5: Trajetória do Veículo

O trabalho de verificação da funcionalidade do NNCS foi dividido em sete etapas.

5.1 Etapa Inicial: Ajuste de parâmetros

Aqui o objetivo principal é conseguir o ajuste de parâmetros para se obter velocidades e desvios na orientação do veículo em ordens de grandeza apropriadas. Nesta etapa, permitimos que somente 1 classificador postasse mensagens, ou seja, aquele com maior valor de avaliação. Observamos a ocorrência de trajetórias elípticas com o centro móvel. No CS convencional, em situações semelhantes, somente órbitas circulares foram observadas. Esta diferença pode ser fruto da utilização de números reais e também da substituição dos classificadores por redes neurais. Notamos dificuldade no aprendizado, fato que atribuímos à utilização de apenas um ganhador para fazer a aposta.

5.2 Etapa 2: 5 ganhadores

Optamos por selecionar para apostar os 5 classificadores com melhores *matching*s. A fórmula da etapa anterior para a aposta:

B = 0.1**strength*

foi substituída por:

B = *matching***strength*

onde *matching* corresponde à saida da rede neural de avaliação, e será usada para medir a especificidade do classificador. Quanto maior, mais específico. Os *bid*s e *strength*s podem ter valores positivos ou negativos. A Figura 6 apresenta os resultados de um

dos testes efetuados. Notam-se velocidades positivas e negativas, inclusive de valores elevados, o que não nos preocupou a esta altura. A partir da iteração 1100 o veículo entra em órbita central, andando de ré. O veículo não tem sensores de ré, mas consegue aprender, mesmo assim, pela simetria do ambiente.

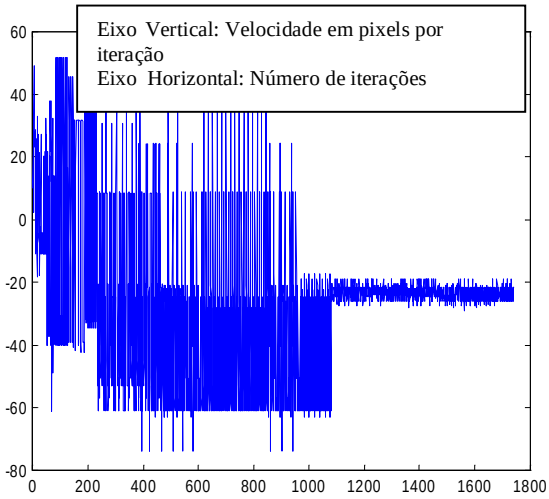

Figura 6: Velocidade x iterações

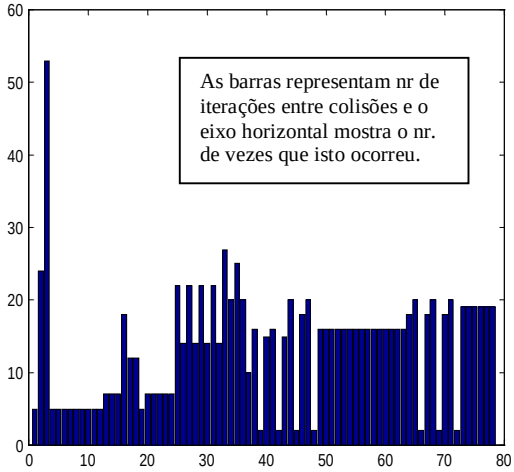

Figura 7: Iterações entre colisões. Depois deste período de aprendizagem, não foram observadas colisões.

A Figura 7 apresenta um gráfico de número de iterações entre colisões. Nota-se bem no início um intervalo de mais de 50 iterações sem colidir. É óbvio que este comportamento não foi aprendido, ilustrando a possibilidade de ocorrência de conhecimento inato, o que de qualquer forma é casual.

Observamos este teste até a iteração 6000 e ele manteve uma órbita central, às vezes com algumas modificações na trajetória descrevendo órbitas mais complexas, combinando elipses com helicoidais.

5.3 Etapa 3: Velocidade e orientação como entradas

Com velocidade variável é razoável considerá-la também como um dado de entrada. No caso da orientação do veículo, como está sendo dado um deslocamento na orientação, não há necessidade. Pelo menos a velocidade como entrada, num estudo futuro, mais aprofundado, terá que ser considerada.

5.4 Etapa 4: Velocidade (v) e deslocamento da orientação (dteta) dentro de uma faixa de valores.

Será possível ajustar-se os parâmetros para a obtenção de valores absolutos de *v* e *dteta* dentro de uma faixa, sem a imposição de restrições? Procuramos obtê-los com o máximo em torno de 25. Conseguimos, sem grande dificuldade, *v* entre -30 e 40, mas com a maior parte entre –10 e 20. Não parece fácil obter resultados melhores desse modo. E como limitar *v* para não assumir valores negativos e assim evitar o uso de sensores traseiros? Uma idéia é utilizar aceleração ao invés de velocidade, com aceleração negativa correspondendo a frear o veículo, o que é sofisticado para este estudo introdutório.

5.5 Etapa 5: Só dteta variável

Para conseguir situações mais simples de serem observadas e podermos aprofundar o entendimento do que é fundamental no aprendizado eliminamos a variação da velocidade. Com isso o atuador passará a ter apenas um neurônio na camada de saída. O aprendizado fica mais nítido, mas o seu desempenho varia muito durante o treinamento. Notamos dois pontos a serem considerados.

- É preciso melhorar o aprendizado, selecionando melhor os classificadores apropriados. A regra sendo utilizada de aposta (penalização) e recompensa parece não estar sendo suficiente.
- O AG atua como uma faca de dois gumes: o conhecimento gera novos comportamentos que terão que ser avaliados e classificados.

A seguir introduziremos um reforço no aprendizado, procurando solucionar o primeiro ponto. Para o segundo, pode-se buscar por períodos mais adequados para atuação do AG.

5.6 Etapa 6: Reforço adicional para o aprendizado

O ideal seria implantar agora o *Bucket Brigade*, mas optamos por um algoritmo estruturalmente mais simplificado, o que permitirá inclusive apontar a viabilidade de sua implantação no futuro. O princípio é penalizar uma seqüência de ações que leve a uma colisão.

Em média, o veículo a 15 pixels por iteração, irá precisar de 5 iterações até atingir uma pilastra. Consideraremos que as três últimas ações foram causadoras da colisão, pois, neste tempo, teria sido possível tomar alguma outra medida para evitá-la. Isto feito houve melhora no aprendizado.

5.7 Etapa 7: Movimento das pilastras

Movimentando as pilastras, tem-se um fluxo de informações mais variado vindo do ambiente. Isto

resultou na necessidade de um número maior de iterações para se adquirir o aprendizado.

6 NNCS – Conclusões e Consderações Finais

Os resultados deste trabalho, pela sua natureza, são notadamente qualitativos. Conseguimos atingir o objetivo de implementar todas as funções do CS convencional usando NNCS e como era de se esperar, em todos os testes, foi consumido um tempo significativo para se ajustar os parâmetros.

Os resultados são surpreendentes, pois foram usadas técnicas bem simples para todas as suas partes. É claro que, com isso, a solução ficou mais dependente das condições iniciais, às vezes sem conseguir solução. Nos casos em que o ambiente foi mantido estacionário, só o veículo se movimentando, ou quando foi dado movimento periódico para a movimentação das pilastras, observamos, em boa parte dos experimentos, a procura por um estado de equilíbrio.

No caso do CS, o equilíbrio mencionado se manifesta em movimentos periódicos bem simples. No caso do NNCS, o equilíbrio é muito mais sofisticado, pois além de utilizar valores reais para as saídas do atuador, as redes neurais devem estar produzindo mapeamentos não-lineares suaves. Assim, ocorrem órbitas de excentricidade variável, helicoidais, e aparentemente periódicas, com períodos bem longos.

7 NNCS – Próximos Passos

A seguir, enumeramos pontos importantes como próximos passos com o propósito de se entender melhor os resultados da utilização de NNCS:

- Verificar uma maneira de não se alterar bruscamente a velocidade e orientação do veículo.
- Como pode ocorrer *matching* negativo, estudar uma maneira mais adequada para fazer a aposta e deduzi-la do *strength*, pois pode ocorrer aposta negativa.
- Pode ser o caso de se considerar que não ocorreu *matching* se a avaliação for negativa.
- Implementar variações baseadas no *Bucket Brigade*.
- Estudar como utilizar *Evolutionary Reinforcement Learning* nos NNCS.
- Desenvolver operadores genéticos específicos, propiciando melhores resultados quando da atuação do AG?
- Melhores operadores genéticos irão facilitar a definição de critérios para a freqüência de ativação do AG.
- Verificar o que há de comum com # (*don't care)* do CFS com a sempre ocorrência de *matching* no NNCS.
- O que ocorre com os *building blocks* no NNCS?
- Considerar a dinâmica do movimento do veículo.

8 Referências bibliográficas

[Ackley & Littman 1991] Interactions Between Learning and Evolution, David Ackley and Michael Littman. Artificial Life II, SFI Studies in the Sciences of Complexity, vol. X, edited by C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen, Addison-Wesley, 1991.

[Belew et. al. 1991] "Evolving Networks: Using the Genetic Algorithm with Connectionist Learning", Richard K. Belew, John McInerney, and Nicol N. Schraudolph. Artificial Life II, SFI Studies in the Sciences of Complexity, vol. X, edited by C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen, Addison-Wesley, 1991.

[Booker et. Al. 1989] "Classifier Systems and Genetic Algorithms" - L. B. Booker, D. E. Goldberg and J. H. Holland - Artificial Intelligence 40 (1989) pp. 235-282.

[Goldberg 1989] "Introduction to Genetics-Based Machine Learning" - Chapter 6 do livro "Genetic Algorithms in Search, Optimization and Machine Learning" - David. E. Goldberg - Addison-Wesley 1989.

[Haykin 1999] Neural Networks – A Comprehensive Foundation. Symon Haykin, 2nd. edition, Prentice-Hall, 1999.

[Holland 1975] "Adaption in Natural and Artificial Systems", University of Michigan Press, An Arbor, MI, 1975.

[Michalewicz 1996] "Genetic Algorithms + Data Structures = Evolution Programs", Zbigniew Michalewicz, 3rd, Revised and Extend Edition, Springer. 12.1 The Michigan approach.

[Poli 1996] "Introduction to Evolutionary Computation", Riccardo Poli - Web Page at the School of Computer Science - The University of Birmingham.

[Richards 1995] "Classifier Systems & Genetic Algorithms" Robert A. Richards - Chapter 3 of Richards, Robert A.; 1995; Zeroth-order Shape Optimization Utilizing a Learning Classifier System Ph.D. Dissertation, Mechanical Engineering Department, Stanford University.

[Simon 1996] The Sciences of the Artificial. H. A. Simon, MIT Press, 1996.