Proposta para Implementação de Aprendizado em Redes de Comportamentos Utilizando Ferramentas do Mecanismo de Esquemas

Anderson Pontes Vieira
Instituto de Computação (IC) – Unicamp
andersonpvieira@gmail.com

Resumo

Este trabalho propõe uma alternativa de aprendizado para um agente baseado em Redes de Comportamentos. Uma Rede de Comportamentos é formada por diversos módulos interligados. Originalmente, as condições em que cada módulo pode ser executado, bem como seus resultados, são definidos pelo projetista. Utilizando ferramentas propostas por Drescher em seu Mecanismo de Esquemas, procuramos dotar o agente da capacidade de descobrir quando uma ação pode ser executada, e quais são suas possíveis consequências dentro daquele contexto. Detalhamos a Rede de Comportamentos modificada e discutimos suas possíveis vantagens e limitações.

1 Introdução

Um dos problemas cruciais no projeto de agentes de software autônomos é a seleção de ação. Um agente deve ser capaz de realizar uma sequência de ações de forma a atingir um objetivo. Deve também identificar certas condições no ambiente e agir de forma oportunista. Como balancear oportunismo e persistência na busca de um objetivo, agindo rapidamente em um ambiente ruidoso e dinâmico? Para tratar esta questão Pattie Maes (1989) propõe a idéia de Redes de Comportamentos.

Outro aspecto a se considerar na criação de um agente é se este será capaz de aprender, de forma a melhor se adaptar ao ambiente em que está inserido. Drescher (1986) apresenta o Mecanismo de Esquemas como um modelo da atividade e evolução de esquemas cognitivos. Uma das características deste modelo é criar previsões cada vez mais confiáveis das consequências de uma ação numa situação dada.

A idéia de unir a forma de seleção de ação da Rede de Comportamentos e a capacidade de aprendizado do Mecanismo de Esquemas surge graças às semelhanças entre os módulos de competência e os esquemas. O restante desta seção procura explicar resumidamente o funcionamento destas duas técnicas. Na seção 2 são detalhadas as modificações realizadas para adicionar aprendizado à Rede de Comportamentos. A seção 3 traz um experimento teórico realizado com o modelo proposto. A seção 4 faz uma breve comparação com trabalhos relacionados. E por fim, a seção 5 trata das limitações do modelo e possibilidades de trabalhos futuros.

1.1 Rede de Comportamentos

Uma Rede de Comportamentos é formada por diversos módulos de competência interligados. As conexões entre os módulos permitem que eles influenciem uns aos outros, e compitam pela possibilidade de serem ativados. Um módulo x é composto por:

- Lista de pré-condições (c_x): proposições que devem ser verdadeiras antes da execução do comportamento.
- Lista de adições (a_x): proposições que devem ser verdadeiras após a execução do comportamento.
- Lista de deleções (d_x): proposições que devem ser falsas após a execução do comportamento.
- Ação: ação ou ações executadas pelo módulo.

Existem três tipos de ligação entre os módulos definidas em função das listas que os compõem:

- Sucessor: uma ligação do módulo x ao módulo y para cada proposição $p \in a_x \cap c_y$.
- Predecessor: uma ligação do módulo x ao módulo y para cada proposição $p \in c_x \cap a_y$.
- Conflitante: uma ligação do módulo x ao módulo y para cada proposição $p \in c_x \cap d_y$.

Estas ligações são usadas para espalhar ativação pela rede. Os módulos se estimulam através das ligações do tipo sucessor e predecessor e se inibem através das ligações conflitantes. Três fontes são responsáveis por ceder e retirar ativação da rede:

- Estado: cada proposição do estado atual envia ativação para os módulos que os têm em suas listas de pré-condições.
- Objetivos: cada objetivo envia ativação para os módulos que os têm em suas listas de adição.
- Objetivos protegidos: cada objetivo protegido retira ativação dos módulos que os têm em suas listas de deleção.

Um módulo é dito executável quando todas as proposições de sua lista de pré-condições são satisfeitas. Para que um módulo seja ativado, ou seja, realize sua ação, três critérios devem ser satisfeitos:

i. o módulo deve ser executável,

ii. o nível de ativação do módulo deve ser maior que um limite θ,

iii. o módulo deve ter a maior ativação dentre todos os que satisfazem i e ii.

Quando um módulo é ativado, sua ação é executada, seu nível de ativação é reiniciado e θ volta ao valor original. Caso nenhum módulo se torne ativo na iteração atual, o valor de θ é decrementado e uma nova iteração é iniciada.

Na concepção original, as listas de pré-condições, adições, deleções, bem como a ação de cada módulo devem ser definidas pelo projetista da rede. À medida que o agente e o ambiente se tornam complexos porém, pensar em todas as condições necessárias para se realizar uma ação, além de todos os possíveis resultados, se torna uma tarefa complicada. Seria interessante permitir que o próprio agente aprendesse, de forma incremental, quais os resultados de uma certa ação dentro de um contexto determinado. Para alcançar esta capacidade propomos a utilização de ferramentas do Mecanismo de Esquemas.

1.2 Mecanismo de Esquemas

O Mecanismo de Esquemas é um modelo computacional construtivista baseado nas teorias de Jean Piaget sobre o desenvolvimento cognitivo infantil. A unidade básica deste modelo é o esquema:

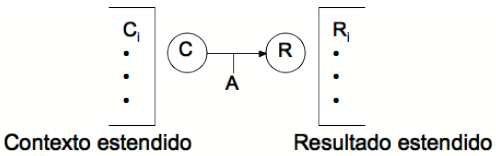

Figura 1: Representação de um esquema.

- Contexto: conjunto de proposições presentes antes da ação.
- Resultado: conjunto esperado de proposições após a ação.
- Ação: ação executada.
- Contexto estendido: informações estatísticas para cada proposição não contida no Contexto do esquema.
- Resultado estendido: informações estatísticas para cada proposição não contida no Resultado do esquema

Segundo o construtivismo, os elementos de representação mental não são inatos, mas construídos do zero por cada indivíduo. Os esquemas iniciais contêm ações simples como mover a mão ou olhar para o lado. Tanto o contexto quanto o resultado destes esquemas são vazios. A partir deles, são construídos todos os outros esquemas do agente. O mecanismo responsável por gerar esquemas novos a partir dos presentes é chamado Atribuição Marginal (Drescher, 1991).

Dizemos que um esquema obteve sucesso se todos os resultados previstos por ele foram obtidos após sua execução. Cada esquema faz um acompanhamento do seu grau de sucesso, que é chamado confiabilidade. Inicialmente o mecanismo de Atribuição Marginal checa o resultado estendido e identifica resultados relevantes, mas não confiáveis, da ativação de um esquema. Um novo esquema é gerado contendo estes resultados. Depois, o contexto estendido é examinado em busca de um item que aumente a confiabilidade do esquema, é gerado então um novo esquema contendo este item. Esquemas cada vez mais confiáveis são criados, aumentando o poder do agente de fazer previsões sobre sua interação com o ambiente.

2 Integração

Podemos estabelecer algumas ligações diretas entre a Rede de Comportamentos e o Mecanismo de Esquemas, mais especificamente, entre um módulo de competência e um esquema.

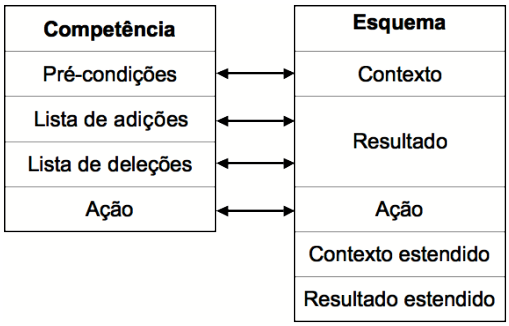

Figura 2: Competência x Esquema.

A competência modificada deverá contar agora tanto com um contexto estendido quanto com um resultado estendido, pois

ambos serão usados pela Atribuição Marginal para gerar novos módulos. Além disso, da mesma forma que um esquema, a competência deverá manter um registro de sua confiabilidade.

A nova rede será iniciada com um conjunto de módulos primitivos que possuem as listas de pré-condições, adições e deleções vazias. Não haverá, portanto, ligações entre os módulos, nem dos módulos com o estado e os objetivos do agente. Nesta situação, todos os módulos serão executáveis e terão o mesmo nível de ativação inicial. O limite θ será decrementado a cada iteração, fazendo com que, eventualmente, um módulo aleatório seja ativado. À medida que o agente realiza estas ações a Atribuição Marginal cria novas competências, cada vez mais confiáveis, contendo listas de pré-condições, adições e deleções mais detalhadas. Estas listas, por sua vez, implicam na criação de ligações entre os módulos, o estado geral e os objetivos, permitindo a atuação do mecanismo de seleção de ação da rede.

Competências confiáveis são mais desejáveis que as pouco confiáveis. O critério de ativação deve ser modificado para levar em conta a confiabilidade:

i. o módulo deve ser executável,

ii. $a \cdot \alpha + b \cdot \rho \geq \theta$,

iii. o módulo deve ter o maior valor $a \cdot \alpha + b \cdot \rho$ dentre todos os que satisfazem i e ii.

Onde α é o nível de ativação e ρ é a confiabilidade de um módulo, e *a* e *b* são usados para ajustar o peso de cada parâmetro.

A condição iii, porém, por ser determinística, implica em um problema: O agente dificilmente voltará a executar os módulos primitivos uma vez que novos módulos mais confiáveis sejam criados. Isso diminuiria consideravelmente a chance de desenvolver novos módulos pouco relacionados aos módulos mais confiáveis criados até o momento. Para amenizar este problema podemos adicionar uma característica estocástica ao algoritmo, utilizando, por exemplo, a seleção por roleta, comum em algoritmos genéticos (Michalewicz & Fogel, 2000), com o valor $a \cdot \alpha + b \cdot \rho$ como parâmetro para construção da roleta.

Se mantivermos a condição ii intacta temos como consequência que módulos pouco confiáveis terão maior chance de serem executados quando o valor de θ for baixo. Ou seja, quando o agente está há muitas iterações sem realizar uma ação. Este efeito é interessante, pois implica na ativação de módulos pouco convencionais justamente quando o sistema encontra dificuldades em realizar uma ação mais confiável numa determinada situação.

3 Experimento teórico

Nesta seção propomos um experimento teórico para demonstrar o funcionamento da Rede de Comportamentos modificada. A situação escolhida envolve Condicionamento Clássico e Condicionamento Operante com reforço positivo e punição. O nosso agente é constituído por um olho e uma mão. O ambiente possui dois botões, um azul, que libera alimento, e um vermelho, que dá um choque no agente provocando desconforto.

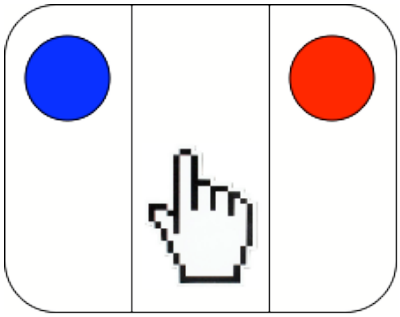

Figura 3: Ambiente observado pelo agente.

A situação inicial do sistema é a seguinte:
Estado: {*nada_à_esquerda, nada_à_direita, nada_à_frente, nada_atrás*}
Objetivos: {*comida*}
Objetivos protegidos: {*evitar_desconforto*}
Módulos primitivos:

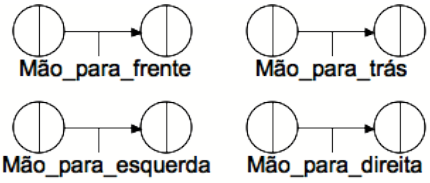

Figura 4: Módulos primitivos. Contexto e resultado são vazios.

Inicialmente todos os módulos são executáveis, possuem confiabilidade nula e a mesma ativação inicial baixa. O valor de θ é decrementado a cada iteração até que $a \cdot \alpha \geq \theta$. Um dos módulos é ativado aleatoriamente e as informações de seu contexto estendido e do seu resultado estendido são atualizadas. Com o tempo, o mecanismo de Atribuição Marginal monitora estas informações e, quando encontra um resultado significativo, cria novas competências, como por exemplo:

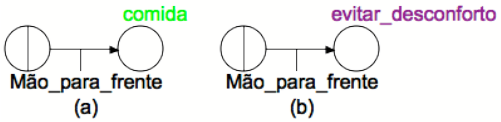

Figura 5: Módulos criados utilizando-se Atribuição Marginal. A cor verde indica que o item pertence a lista de adição. A cor roxa indica que o item pertence a lista de deleção.

Os módulos da Figura 5 indicam que há uma chance maior de obter comida ou de se levar um choque quando a mão é descolada para frente. Outra forma de obter comida não mostrada na figura seria descolando-se a mão para esquerda a partir da posição superior no meio. O módulo 5a recebe ativação do objetivo *comida* sempre que este estiver ativo. Já o módulo 5b é inibido pelo objetivo protegido *evitar_desconforto*, fazendo com que ele seja ativado com menor frequência.

O algoritmo continua executando, e agora a Atribuição Marginal procura por um item do contexto estendido que aumente a confiabilidade dos módulos. Quando estes itens são encontrados novos módulos são gerados:

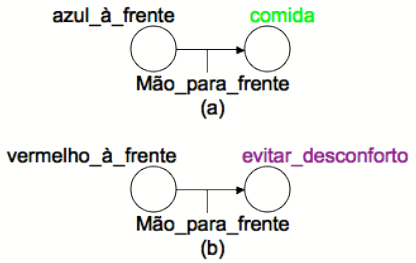

Figura 6: Módulos confiáveis criados utilizando-se Atribuição Marginal.

O agente aprende que a consequência de apertar o botão azul é conseguir comida, e que apertar o botão vermelho provoca desconforto. O módulo 6a recebe ativação do objetivo *comida* e o módulo 6b é inibido pelo objetivo protegido *evitar_desconforto*. Dessa forma, há um aumento do comportamento "Apertar o botão azul" e uma diminuição do comportamento "Apertar o botão vermelho", mostrando que o algoritmo é capaz de aprendizado por Condicionamento Operante.

Em relação ao Condicionamento Clássico, de forma semelhante ao que ocorre no experimento feito por Krichmar e Edelman (2002), o nosso agente associa uma característica visual a um efeito bom ou ruim, dando preferência ao módulo 6a que cumpre um objetivo, e evitando o módulo 6b que conflita com um objetivo protegido.

4 Trabalhos relacionados

Maes e Brooks (1990) descrevem um algoritmo para aprendizado em Redes de Comportamento. A principal diferença entre a abordagem deles e a apresentada neste trabalho é o tipo de aprendizado realizado. No caso do algoritmo proposto por Maes e Brooks, não são geradas novas competências a partir de módulos primitivos. O agente aprende o melhor momento para se ativar uma competência, de modo a maximizar o *feedback* positivo e minimizar o *feedback* negativo. Já a Atribuição Marginal, aqui utilizada, cria uma representação do mundo, onde cada competência é uma previsão, com um certo grau de confiabilidade, do que aconteceria caso uma determinada ação fosse tomada dentro de um contexto dado. Podemos esperar, no entanto, que a influência dos objetivos na ativação dos módulos acabe funcionando também como uma espécie de *feedback*. Como competências que realizam os objetivos do agente são ativadas com mais frequência, a geração de novos módulos tende a ocorrer mais a partir delas. Já os módulos que conflitam com objetivos protegidos são pouco utilizados. Assim, o aprendizado obtido também é orientado aos objetivos do agente.

Uma vantagem do algoritmo de Maes e Brooks é a menor complexidade computacional, já que o sistema monitora apenas o tipo de *feedback*, sem necessidade de manter dados estatísticos atualizados para todos os itens percebidos pelo agente.

5 Limitações e trabalhos futuros

A primeira limitação que surge no algoritmo proposto é a complexidade computacional. Devido a necessidade de cada módulo manter um contexto e um resultado estendidos com informações sobre o comportamento de todas as proposições, o sistema tende a ficar mais lento com o aumento da quantidade de módulos. Esta característica é antinatural, pois o esperado seria que o agente se tornasse mais rápido à medida que adquirisse mais conhecimento.

A implementação de foco de atenção, como descrito em Foner e Maes (1994), poderia diminuir a quantidade de itens que precisam ser acompanhados ao mesmo tempo, aumentando assim a velocidade do algoritmo. Outra alternativa seria a implementação de alguma forma de esquecimento. Módulos pouco usados poderiam ser apagados, começando a partir dos mais complexos, e mantendo intactos somente os módulos primitivos.

Outras ferramentas presentes no Mecanismo de Esquemas, como Ações Compostas e Itens Sintéticos (Drescher, 1986), poderiam ser adicionadas ao algoritmo possibilitando a criação de construções mais sofisticadas. Uma Ação Composta utiliza um

conjunto de esquemas para atingir um resultado. Um Item Sintético representa algo independente de existir uma informação sensorial acerca daquilo naquele instante.

Para a apresentação de dados mais concretos é ainda necessário realizar uma implementação completa e testar o algoritmo em um ambiente simulado.

6 Bibliografia

Drescher, G. L. (1986) Genetic AI - Translating Piaget into LISP

Drescher, G. L. (1991) Made-Up Minds: A Constructivist Approach to Artificial Intelligence

Foner, L. N., Maes, P. (1994) Paying Attention to What's Important: Using Focus of Attention to Improve Unsupervised Learning.

Krichmar, J.L., Edelman, G.M. (2002) Machine Psychology: Autonomous Behavior, Perceptual Categorization and Conditioning in a Brain-Based Device

Maes, P. (1989) How To Do the Right Thing

Maes, P., Brooks R.A. (1990) Learning to Coordinate Behaviors

Michalewicz Z., Fogel D.B. (2000) How to Solve It - Modern Heuristics