NAVEGAÇÃO DE ROBÔS MÓVEIS UTILIZANDO APRENDIZADO POR REFORÇO E LÓGICA FUZZY

Gedson Faria*
gedson@icmc.usp.br

Roseli A. Francelin Romero*
rafrance@icmc.usp.br

*Departamento de Computação e Estatística, Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo – Campus de São Carlos
Av. Trabalhador Sãocarlense 400, Caixa Postal 668 13560-970, São Carlos, SP, Brasil

ABSTRACT

Reinforcement Learning can be seen as a way of programming agents by reward and punishment for solving specific tasks through repeated interactions with the environment. In this work, the performance of the most important reinforcement learning algorithms: Q-learning, Rlearning, H-learning is investigated in the context of a navigation task avoiding obstacles. Furthermore, this work proposes a sensor-based navigation method, named R'-learning, which incorporates fuzzy logic into the Rlearning algorithm for mobile robot navigation in uncertain environment. An application is realized consisting of teaching the robots to find small objects in a corridor. For this, a state set mapping has been proposed through force field concepts. R'learning algorithm has been used for this navigation task. The robot showed to have satisfactory behaviors in performing this task.

KEYWORDS: Mobile robot, navigation, reinforcement learning, fuzzy logic.

RESUMO

Aprendizado por Reforço pode ser visto como uma forma de programar agentes utilizando recompensas e punições para resolver tarefas específicas através de interações com o ambiente. Neste trabalho é investigado o desempenho dos mais importantes algoritmos de aprendizado por reforço: Q-learning, R-learning e H-learning, no contexto de tarefas de navegação evitando colisões. Além disso, é proposto neste trabalho um método de navegação por sonares, denominado R'-learning, que incorpora lógica *fuzzy* ao algoritmo R-learning para navegação de robôs móveis em ambientes incertos. Foi também realizada uma aplicação de navegação utilizando o algoritmo R'-learning que consiste em ensinar robôs a encontrar pequenos objetos em um corredor. Para isto, foi proposto um mapeamento de estados do ambiente a partir do conceito de força de repulsão. O robô mostrou ter comportamentos satisfatórios ao executar a tarefa proposta.

PALAVRAS-CHAVE: Robôs móveis, navegação, aprendizado por reforço, lógica fuzzy.

1 INTRODUÇÃO

O aprendizado em robôs consiste essencialmente em fazer com que o robô execute tarefas sem a necessidade de programá-los explicitamente. A programação de robôs é uma tarefa extremamente desafiadora, por muitas razões. O sensor de um robô, como por exemplo, o sonar, tem comportamentos imprevisíveis, algumas vezes variando conforme o ambiente. Sendo assim, não basta apenas conhecer o funcionamento dos sensores, também deverá ser fornecido um modelo do ambiente no qual

Artigo submetido em 17/04/01
1a. Revisão em 27/06/01; 2a. revisão em 03/12/01
Aceito sob recomendação do Ed. Assoc. Takashi Yoneyama

o robô deverá atuar. Para programar um robô, o problema deve ser decomposto em uma sucessão de tarefas até chegar em operações de baixo nível, tais como, andar, virar à esquerda, etc... Por estas razões, há um interesse considerável em que os robôs possam aprender a realizar tarefas automaticamente.

Nos últimos anos, a pesquisa em IA tem procurado substituir programação explícita pelo processo de ensinar uma tarefa. Pesquisas nesta área têm estudado várias formas de implementação de aprendizado (Mitchell, 1997).

Segundo Haykin (1994) há três paradigmas de aprendizado: supervisionado, não-supervisionado e aprendizado por reforço, sendo este último o foco de atenção do presente trabalho.

O aprendizado por reforço (AR) é um dos principais paradigmas de aprendizado de Robótica Móvel, utilizado para descrição de problemas no qual um agente (o robô) aprende estratégias interagindo com seu ambiente (Ribeiro at al, 2001).

O ponto crucial em um algortimo de AR é fazer com que o agente *aprenda* as condições (associações entre estados observados e escolha das ações) que conduzem a recompensas ou punições (Ribeiro, 1999).

O problema de aprendizado por reforço pode ser modelado como: um conjunto de estados do ambiente, um conjunto de ações e um conjunto de recompensas. Para cada ação realizada em um estado do ambiente o agente recebe uma recompensa. O agente não sabe qual é a melhor ação a tomar, como em muitas formas de aprendizado de máquina, por isso ele deve descobrir, através de tentativas, quais ações lhe rendem maior recompensa.

Outros problemas encontrados em aprendizado de robôs, tais como, ambientes estocásticos[1], resposta em tempo real e aprendizado *on-line*, também são resolvidos utilizando-se aprendizado por reforço.

Aplicações de AR podem ser encontradas nas áreas de robótica, produção industrial, otimização combinatória, como jogos de computador. Littman at al (1994) utilizou os algoritmos de AR para o Jogo de Damas e Thrun (1995) utilizou AR no jogo de Xadrez. Métodos de AR foram utilizados por vários pesquisadores para o controle e navegação de robôs, podendo ser encontrados em (Mahadevan and Connell, 1992; Schaal and Atkeson, 1994; Crites and Barto, 1996; Mataric, 1994; Bagnell et al., 1998).

Existem na literatura, experimentos de aprendizado sobre ambientes simulados e perfeitos, nos quais nunca ocorrem erros na execução das ações e nem na percepção dos estados do ambiente. Contudo, esses ambientes não fornecem um modelo consistente para aplicações de navegação em ambientes dinâmicos. Em outros trabalhos é considerado o ambiente real, mas um mapa do ambiente é fornecido previamente ou construído durante a exploração do robô. Esta abordagem limita a tarefa de evitar colisões ao mapa do ambiente, pois as regras de navegação para o robô serão criadas de acordo com este mapa.

Neste trabalho, o ambiente do robô foi considerado como desconhecido e não foi mantido nenhum tipo de mapa do ambiente para o aprendizado de evitar colisões. Desta forma, a política de controle do robô serve para qualquer ambiente interno não necessitando que o robô tenha um novo treinamento quando o ambiente for mudado. Além disso, alguns algoritmos de AR foram investigados para verificar seus desempenhos em aplicações de tempo real, para o robô Pioneer 1 Gripper[2], em ambientes estocásticos e desconhecidos.

Este trabalho está organizado como segue. Na Seção 2 são apresentados uma revisão sobre AR, a forma de modelagem do problema de aprendizado e o modelo de MDP (*Markov Decision Process*). Os algoritmos de aprendizado, mais conhecidos, tais como, Q-learning (Watkins, 1989), Rlearning (Schwartz, 1993) e Hlearning (Tadepalli and Ok, 1994) são apresentados na Seção 3. Na Seção 4, o robô Pioneer 1 Gripper e o seu software de controle são apresentados. A implementação do modelo MDP para a tarefa de navegação, assim como as comparações de desempenho entre os algoritmos investigados são detalhadas na Seção 5. Um novo algoritmo, R'-learning, que incorpora conceitos de lógica fuzzy, é proposto por nós, na Seção 5. Na Seção 6, é detalhada uma aplicação real para o robô Pioneer 1 Gripper, utilizando o algoritmo R'-learning. Na Seção 7, são apresentadas as conclusões e sugestões para trabalhos futuros.

2 APRENDIZADO POR REFORÇO

Problemas em Aprendizado por reforço (AR) são caracterizados por um agente que deve aprender comportamentos através de iterações de tentativa e erro em um ambiente dinâmico (Kaelbing and Littman, 1996). AR não é definido como um conjunto de algoritmos de

[1] Ambiente estocástico, também conhecido como ambiente não-determinístico, é um ambiente no qual uma mesma ação realizada em um mesmo estado do ambiente, em tempos distintos, pode retornar diferentes estados seguintes e/ou recompensas diferentes.

[2] Fabricado por ActivMedia, Inc.

aprendizagem, mas como uma classe de problemas de aprendizagem. Todo o algoritmo que resolver bem esse problema será considerado um algoritmo de aprendizado por reforço (Sutton and Barto, 1998).

Aprendizado por reforço é baseado na idéia que, se uma ação é seguida de estados satisfatórios, ou por uma melhoria no estado, então a tendência para produzir esta ação é aumentada, isto é, reforçada. Estendendo esta idéia, ações podem ser selecionadas em função da informação sobre os estados que elas podem produzir, o que introduz aspectos de controle com realimentação.

Quase todos métodos de AR em uso são baseados na técnica de Diferenças Temporais (TD) (Sutton, 1988). A idéia fundamental em TD é o aprendizado por predição: quando o agente recebe um reforço deve propagá-lo de alguma maneira no tempo. Desta forma, os estados que foram anteriormente visitados e que conduziram a esta condição, serão associados a uma predição. Isto está baseado em uma suposição importante em processos dinâmicos, chamado Processo de Decisão de Markov (MDP).

A aquisição de informações de aprendizado por experiência direta é uma prática que normalmente não está sob o total controle de um agente: ele pode escolher ações, mas não pode determinar as conseqüências destas ações com antecedência para todos os estados, pois normalmente não tem um modelo suficientemente preciso do processo no qual são baseados os julgamentos. Por isto, o agente deve estimar o custo esperado através de visitação direta aos estados; ele deve escolher uma ação, receber um resultado e propagá-lo aos estados anteriores, seguindo o procedimento de TD.

Uma das condições necessárias na qual algoritmos de AR podem encontrar uma política de ação é a exploração completa do espaço de estados, normalmente impossível em situações prática. Quando controle e aprendizado estão em jogo, o agente de aprendizado deve tentar encontrar um equilíbrio entre a exploração de alternativas de uma dada política e a exploitação desta política como um mecanismo para avaliar os custos associados. Em outras palavras, tentando-se alternativas desconhecidas pode-se correr riscos, mas mantendo-se sempre a mesma política não se consegue um aperfeiçoamento (Ribeiro, 1999).

Esta contradição entre exploração e exploitação, entre objetivo de controle e objetivo de estimação é um assunto bem conhecido na teoria de Controle Ótimo e é comumente referenciada como problema de controle dual (Bertsekas, 1995).

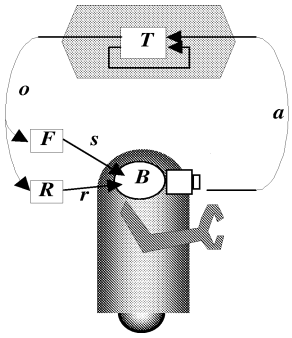

Figura 1: Modelo padrão de Aprendizado por Reforço.

2.1 Modelo Padrão de AR

No modelo básico de aprendizado por reforço, um agente é conectado a seu ambiente por percepção e ação, como descrito na Figura 1. Em cada passo de interação, o agente recebe como entrada o estado atual do ambiente, $\boldsymbol{s}$, proveniente da observação, $\boldsymbol{o}$, do ambiente; o agente escolhe então uma ação, $\boldsymbol{a}$, gerada como saída. A ação muda o estado do ambiente e é comunicado o valor desta transição de estado ao agente por um sinal de reforço escalar, $\boldsymbol{r}$. O comportamento do agente, $\boldsymbol{B}$, deve escolher ações que tendem a aumentar, ao longo do tempo, a soma dos valores do sinal de reforço. Ele pode aprender a fazer isto com o passar do tempo através da sistemática tentativa e erro, guiado por uma grande variedade de algoritmos.

Formalmente, o modelo consiste de: um conjunto discreto de estados de ambiente, S; um conjunto discreto de ações de agente, A; e um conjunto de sinais de reforço, tipicamente {0,1}, ou números reais.

A Figura 1 também inclui uma função de entrada F, que determina como o agente transforma as observações, $\boldsymbol{o}$, do ambiente em estado do ambiente, $\boldsymbol{s}$. Somente para estudos dos algoritmos, assume-se que F é a função identidade, ou seja, o agente percebe o estado exato do ambiente.

A maioria dos sensores de robô, incluindo dispositivos baratos como os sonares e dispositivos caros como os escâneres a laser, são incertos. Por este motivo, às vezes objetos não são vistos ou suas distâncias não são dadas corretamente. Para atenuar este problema propomos neste trabalho a utilização de lógica *fuzzy* na função F, para transformar os sinais recebidos pelo sonar, $\boldsymbol{o}$, em um estado atual, $\boldsymbol{s}$. Uma forma bastante conhecida utilizada atualmente para modelar problemas de AR é o Processo de Decisão de Markov (MDP).

2.2 Processo de Decisão de Markov

Na estrutura de aprendizagem por reforço, o agente faz suas decisões com base num sinal do ambiente, chamado de estado do ambiente. Caso um estado do ambiente contenha toda a informação relevante então ele é chamado de Markoviano ou que tem a propriedade de Markov. Pode-se observar esta propriedade tomando-se, por exemplo, o movimento de uma bola atirada por um canhão. Para determinar seu vôo futuro, não importa com que velocidade saiu e de que posição veio, basta conhecer a velocidade e a posição atual.

Uma tarefa de AR que satisfaça a propriedade de Markov é chamada de Processo de Decisão de Markov ou MDP. Se os estados e ações forem finitos, então será chamado de MDP finito. Formalmente, um MDP consiste de:

- um conjunto de estados do ambiente, $\boldsymbol{S}$;
- um conjunto de possíveis ações, $\boldsymbol{A(s)}$;
- uma função de probabilidade de transição para $\boldsymbol{s'}$ dado $\boldsymbol{s}$ e $\boldsymbol{a}$, $\boldsymbol{P(s'|s,a)}$;
- recompensas esperadas para a transição para $\boldsymbol{s'}$ dado $\boldsymbol{s}$ e $\boldsymbol{a}$, $\boldsymbol{R(s'|s,a)}$;

onde s', s $\in \boldsymbol{S}$ e a $\in \boldsymbol{A(s)}$.

Existem boas referências para MDPs, que podem ser encontradas em (Bellman, 1957; Bertsekas, 1987; Howard, 1960; Puterman, 1994). Maiores detalhes sobre AR podem ser encontrados em (Kaelbing and Littman, 1996; Ribeiro, 1999; Sutton and Barto, 1998).

3 ALGORITMOS DE APRENDIZADO POR REFORÇO

Os métodos de aprendizado por reforço estão divididos em: *métodos independentes de modelo* e *métodos baseados em modelo.*

Um *modelo* consiste em se ter o conhecimento prévio da função de probabilidade de transição *P(s'|s,a)* e da função de reforço *R(s'|s,a).*

Os métodos independentes de modelo (*model-free*) aprendem um controlador sem ter um modelo explícito dos efeitos das ações, já os métodos baseados em modelo (*model-based*) aprendem e usam um modelo de ações enquanto simultaneamente aprendem um controle ótimo (Barto et al., 1993). A preocupação principal em aprendizado por reforço é obter uma política ótima quando o modelo for desconhecido.

Os algoritmos Q-learning (Watkins, 1989) e Rlearning (Schwartz, 1993) são exemplos de métodos independentes de modelo, enquanto Hlearning (Tadepalli and Ok, 1994) é um exemplo de método baseado em modelo.

3.1 Algoritmos Q-learning e SARSA

O algoritmo Q-learning, uma técnica proposta por Watkins (1989), é um método iterativo para o aprendizado de uma política de ação em agentes autônomos. Este método é baseado na medida de custo de ações, $Q(s,a)$, o qual representa o custo descontado esperado por executar uma ação a no estado s, seguindo-se uma política ótima. Em cada interação com o ambiente $Q(s,a)$ é atualizado pela regra:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

onde, γ é o fator de desconto e α é a taxa de aprendizado. O fator γ pode ser interpretado de vários modos. Pode ser visto como uma taxa de gratificação, como uma probabilidade de ir para o próximo estado ou como um artifício matemático para evitar a soma infinita (Kaelbing and Littman, 1996).

Após executar uma ação a, o agente deixa o estado s e vai para s', recebendo por esta transição uma recompensa imediata r. $\max_{a'} Q(s',a')$ é uma previsão da recompensa do próximo estado.

Uma variação interessante para o Q-learning é o algoritmo SARSA (Sutton, 1996). A atualização dos custos das ações é realizada em cada passo de acordo com a equação abaixo:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma Q(s',a') - Q(s,a)] \quad (2)$$

Naturalmente, caso a ação escolhida a' seja $\max_{a'} Q(s',a')$, este algoritmo será equivalente ao Qlearning padrão. Entretanto, o algoritmo SARSA admite que a' seja escolhido aleatoriamente com uma probabilidade predefinida. Por eliminar o uso do operador **max** sobre as ações, este método é mais rápido que o Q-learning para situações onde o conjunto de ações tenha cardinalidade alta.

3.2 Algoritmo R-learning

R-learning é uma técnica proposta por Schwartz (1993) que maximiza a recompensa média a cada passo. R-learning é um método de controle independente de política para versões avançadas de aprendizado por reforço nas quais não se utilizam descontos e nem dividem as experiências em episódios distintos com retornos finitos.

O algoritmo Q-learning não maximiza a recompensa média, mas descontos acumulados de recompensa, por isso R-learning pode fornecer de fato resultados melhores que o Q-learning. No R-learning utiliza-se o *average-reward model*, ou seja, a função de custo para uma política π é definida em relação à média das recompensas esperadas em cada passo de tempo, como:

$$\rho^{\pi} = \lim_{h \to \infty} E\left(\frac{1}{h}\sum_{t=0}^{h} r_t\right) \quad (3)$$

É assumido que a probabilidade de se sair de qualquer estado e ir para outro sob uma política deve ser diferente de zero e deste modo ρ^{π} é independente do estado inicial.

Os valores de $R(s,a)$ são ajustados a cada ação baseados na seguinte regra:

$$R(s,a) \leftarrow R(s,a) + \alpha[r - \rho + \max_{a'} R(s',a') - R(s,a)], \quad (4)$$

que difere da regra do Q-learning, simplesmente por subtrair a recompensa média ρ do reforço imediato $\boldsymbol{r}$ e por não ter desconto γ para o próximo estado. A recompensa média é calculada como:

$$\rho \leftarrow \rho + \beta[r + \max_{a} R(s',a) - \max_{a} R(s,a) - \rho] \quad (5)$$

O ponto chave é que ρ somente é atualizado quando $R(s,a)$=max$_a R(s,a)$. A recompensa média ρ não depende de algum estado particular, ela é uma constante para todo o conjunto de estados.

3.3 Algoritmo H-learning

O algoritmo H-learning (Tadepalli and Ok, 1994) é um algoritmo que utiliza métodos baseados em modelo e assim como o R-learning foi introduzido para otimizar a recompensa média sem utilizar desconto.

O algoritmo H-learning estima as probabilidades *P(s'|s,a)* e os reforços $R(s,a)$ por contagem direta e atualiza os valores da recompensa esperada $\boldsymbol{h}$ utilizando a Equação (6).

$$h(s) = \max_{a \in A(s)} \left\{ r(s,a) + \sum_{s'=1}^{n} P(s'|s,a)h(s') \right\} - \rho \quad (6)$$

O H-learning pode fazer escolhas aleatórias de ações com uma probabilidade fixa. O método utilizado para atualizar a recompensa média ρ segue a idéia de Schwartz (1993).

Todos os métodos em Aprendizado por Reforço, exceto o H-learning, tem um ou mais parâmetros, como por exemplo, o Q-learning tem α e γ e o R-learning tem α e β. A performance de todos estes algoritmos é sensível a estes parâmetros, e conseqüentemente fica necessário ajustá-los para obter um melhor desempenho.

A descrição completa destes três algoritmos pode ser encontrada em (Faria and Romero, 2000a).

Esses três algoritmos foram implementados e testados para uma tarefa de aprendizado para o robô Pioneer 1 Gripper. Desta forma, o robô, seus recursos e seu software de controle são apresentados na próxima seção e a modelagem do problema, assim como os resultados dos testes com os algoritmos são apresentados na Seção 5.

4 ROBÔ E SOFTWARE DE CONTROLE

Nesta seção são apresentados os sensores, atuadores e algumas limitações do robô Pioneer 1 Gripper, seguido do seu software de controle Saphira. O robô Pioneer 1 Gripper (Figura 2) é montado sobre um eixo de duas rodas que lhe permite fazer rotações e movimentos para frente ou para trás. Este robô possui sete sonares ultra-

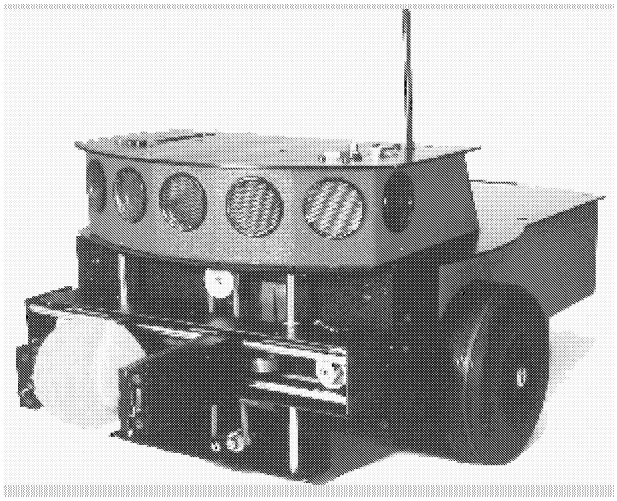

Figura 2: Pioneer 1 Gripper^{ActivMedia,Robotics}

sônicos, sendo cinco frontais, um na lateral direita e um na lateral esquerda. Os valores de leitura dos sonares variam de 200mm à 5000mm. Com freqüência, os sonares utilizados não conseguiram detectar objetos mais próximos que 200mm e superfícies que estavam a 45^o da normal do sonar, devido ao ângulo de reflexão dos raios ultra-sônicos.

O software utilizado para controle do robô Pioneer 1 é o Saphira da ActivMedia, Inc., v.6.1 (Konolige, 1997). Este software vem acompanhado de simulador do robô

e uma biblioteca de funções em linguagem C que permite incluir ao controlador Saphira novos procedimentos. Na 5(a) pode ser observado um ambiente criado no Simulador do robô e na 5(b) é mostrado como Controlador Saphira representa as leituras dos sonares realizadas neste ambiente.

Os sensores do robô servem para lhe dar uma noção do mundo real. É através das informações recebidas pelos sonares que o ambiente é mapeado em um conjunto de estados necessários aos algoritmos de aprendizado. Na próxima seção, será apresentado o mapeamento dos estados do ambiente através dos sonares, assim como a comparação entre os algoritmos Qlearning, Rlearning e Hlearning.

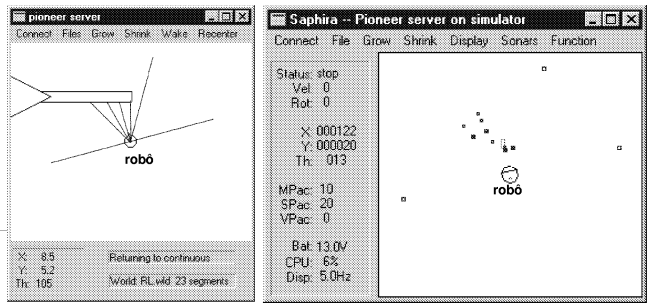

Figura 3: (a) Simulador. Para melhor visualização foram incluídas linhas partindo do centro do robô representando os raios ultra-sônicos dos sonares. (b) Controlador Saphira: Os pontos são representações gráficas das leituras dos sonares, ou seja, como o robô "vê" o mundo ao seu redor.

5 DESCRIÇÃO DOS EXPERIMENTOS

Nesta seção, serão detalhados os passos seguidos na modelagem de uma tarefa de aprendizado em um modelo MDP finito. O mesmo modelo MDP será usado nos três algoritmos e então será apresentada uma comparação de desempenho dos algoritmos.

Posteriormente, será proposto um novo algoritmo de aprendizado que incorpora ao R-learning, conceitos de lógica *fuzzy*. Este algoritmo foi denominado por nós de R'-learning.

A tarefa escolhida para ser modelada como um problema de aprendizado por reforço foi fazer o robô aprender a navegar em um ambiente desconhecido evitando obstáculos.

5.1 Modelagem da Tarefa de Navegação

Para criar um modelo MDP deve-se definir: o conjunto de estados do ambiente, S; o conjunto de ações, $A(s)$; as probabilidades de transições entre os estados, $P(s'|s,a)$; e as recompensas para essas transições, $R(s'|s,a)$. Os conjuntos P e R não são conhecidos, portanto os algoritmos de aprendizado utilizarão somente os conjuntos S, $A(s)$ e r' para definir uma política de controle ótima, onde r' é o conjunto de recompensas imediatas.

Neste problema de navegação espera-se que o robô sempre ande para frente e só gire para evitar colisões com obstáculos, portanto todas as vezes que escolher ir para frente o robô deve ser recompensado. Não se espera que o robô ande para trás ou que ele colida com objetos, desta forma estas são situações que o robô deve ser punido.

Com isso, define-se o conjunto de ações como:

$$A = \{\textit{avançar, girar à direita, girar à esquerda, recuar}\}.$$

Com base nas recompensas imediatas utilizadas por Bagnell et al. (1998) com o robô "Charm", foi criado um conjunto de recompensas que pode ser observado na Tabela 1. Note que a recompensa imediata pode ser dada tanto sobre as ações como sobre o estados. No caso da tarefa de navegação, se for detectado uma colisão, o robô recebe a punição pela colisão, não importando com qual ação o robô colidiu. Para mapear

Tabela 1: Reforço imediato para a tarefa de navegação

colisão	**avançar**	**girar à direita**	**girar à esquerda**	**recuar**
-1000	100	50	50	-50

os estados do ambiente, deve-se considerar uma função $f(d_i)$ que classifica a distância, d, retornada pelo sonar, i, em noções de distância. Estas noções de distância são representadas por números binários, sendo estes diretamente proporcional à distância do objeto detectado pelo sonar. O estado do ambiente, s, será obtido através da concatenação dos valores binários de $f(d_i)$, ou seja, $s = f(d_0)f(d_1)f(d_2)...f(d_6)$. Um estado deve ser acrescentado para representar o estado de colisão, pois este estado não é detectado através dos sonares.

Ao mapear todos os sonares através da função $f(d)$, não se conseguiu convergência de nenhum algoritmo para uma política de comportamento satisfatória. Neste trabalho, considera-se que uma política é satisfatória

quando o robô aprende a navegar, ou seja, consegue mover-se pelo ambiente evitando obstáculos.

Desta forma, foi por nós proposto um modelo que classificasse os sinais recebidos pelos sete sonares como uma distância à esquerda, uma distância à direita e uma distância à frente, para reduzir o número de estado do ambiente. A distância dos objetos à frente foi adotada como sendo a distância mínima recebida pelos cinco sonares frontais, Figura 4. A classificação dos sinais foi feita

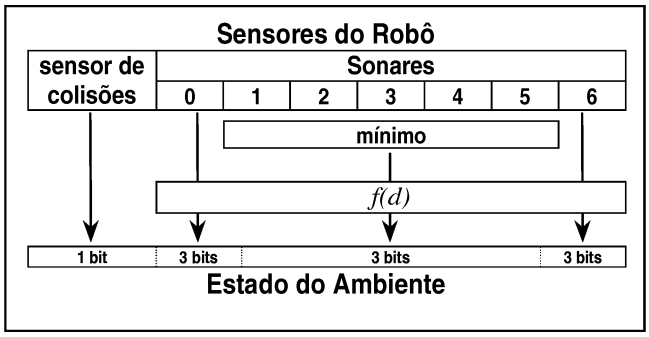

Figura 4: Estado do Ambiente mapeado através da simplificação das leituras dos setes sonares em três sinais de entrada, diminuindo sensivelmente o número de estados.

através de 5 classes, cuja função de distância adotada é apresentada em (7). Desta forma, tem-se um total 5^3 estados do ambiente.

$$f(d) = \begin{cases} 000_b & se \quad 0 \leq d \leq 400 \\ 001_b & se \quad 400 < d \leq 600 \\ 010_b & se \quad 600 < d \leq 800 \\ 011_b & se \quad 800 < d \leq 1000 \\ 100_b & se \quad 1000 < d \leq 5000 \end{cases} \tag{7}$$

Com o modelo, representado esquematicamente na Figura 4, foi alcançada uma convergência para uma política de controle, para os algoritmos Q-learning com $\gamma = 0.75$, R-learning com $\beta = 0.001$ e H-learning. Em todos os algoritmos foi utilizada uma taxa de aprendizado $\alpha = 0.2$. A política encontrada fez com que o robô navegasse pelo ambiente evitando colisões. Todavia, os movimentos do robô não pareciam tão inteligentes, desviando bruscamente dos obstáculos. Isto aconteceu devido, talvez, à própria simplificação realizada nos cinco sonares frontais.

Através de observações realizadas durante o aprendizado pode-se notar as seguintes características no comportamento dos algoritmos testados:

O algoritmo Q-learning é muito sensível a variável γ e simples variações interferem diretamente na convergência para uma política de comportamento satisfatória.

Para este problema de aprendizado, o H-learning foi o algoritmo que levou mais tempo para atingir uma política de comportamento satisfatória. Isto contrariou o esperado, pois, de acordo com Tadepalli and Ok (1994) este algoritmo converge mais rápido que os demais algoritmos.

Os algoritmos Q-learning e R-learning apresentaram tempos de convergência muito próximos para atingir uma política de comportamento satisfatória. Portanto, para verificar quais destes algoritmos apresentaram melhor desempenho foi feita uma comparação com o número de colisões durante o aprendizado. Percebeu-se que o algoritmo Rlearning diminuiu o número de colisões bem antes do Qlearning.

Buscando uma melhor forma de classificar as leituras dos sonares em estados do ambiente, foi por nós proposto uma classificação de leituras dos sonares através do uso de funções *fuzzy* (Shaw and Simões, 1999). O modelo utilizando lógica *fuzzy* foi incorporado ao algoritmo R-learning, gerando o novo algoritmo, denominado por nós de R'-learning.

5.2 Modelo Incorporando Conceitos de Lógica Fuzzy

Notou-se que a classificação da leitura do sonar poderia ser mais bem representada se fosse considerada a chance da distância lida pelo sonar pertencer às outras classes. As incertezas geradas seriam, desta forma, utilizadas pelo o algoritmo de aprendizado para ponderar a recompensa imediata.

Assim sendo, para resolver este problema foi por nós proposto o algoritmo R'-learning (Faria and Romero, 2000b). Neste algoritmo, as leituras dos sonares são classificadas através de funções *fuzzy* para que valores muito próximos de outras classes não sejam considerados tão precisos quanto os demais.

Para classificar a distância recebida por um sonar, foram definidas quatro funções *fuzzy*, cada uma representando uma noção de distância, como pode ser observado na Figura 5. Além de classificar os três sinais de leitura dos sonares, as funções *fuzzy* também retornam o grau de pertinência para cada classe. Com estes três novos valores foi criada uma nova variável, φ (*grau de certeza*), proposta pela necessidade de considerar o fato que o robô possa estar na fronteira de dois conjuntos *fuzzy* vizinhos. A variável φ foi definida como sendo a média aritmética dos graus de pertinência recebidos das funções *fuzzy* que codificam os sinais de entrada.

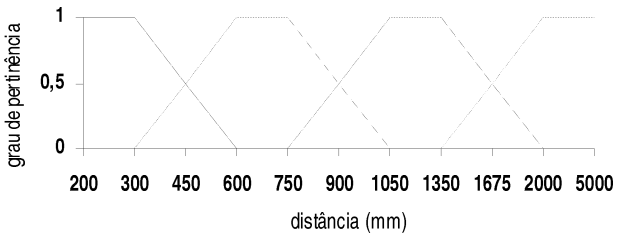

Figura 5: Classificação da leitura dos sonares em quatro funções *fuzzy*.

O mapeamento dos estados do ambiente utilizando funções *fuzzy* pode ser observado na Figura 6, onde é mostrada a classificação dos três sinais de distância e inserção da nova variável φ. A diferença entre os algoritmos

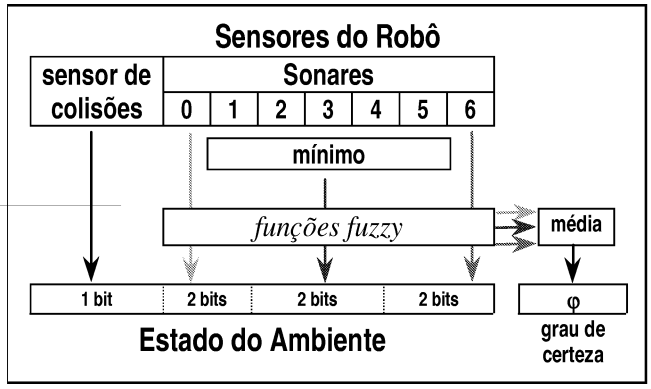

Figura 6: Definição do estado do ambiente incorporando lógica fuzzy.

R'-learning e Rlearning, está no fato do primeiro considerar a variável φ, para ponderar o reforço imediato, r. Desta forma, o novo valor da recompensa imediata, r', foi definido como segue:

$$r' = \varphi r \tag{8}$$

Observe que $r' = r$somente quando todas as classificação de leitura do sonar tiverem grau de pertinência 1. Desta forma, fornece-se ao algoritmo um r' sempre menor que r quando alguma incerteza for detectada.

Ambos algoritmos, R-learning e R'-learning foram testados usando o simulador do robô Pioneer 1, adotando α=0.2 e β=0.001, sendo que o robô conseguiu navegar de forma satisfatória utilizando ambos algoritmos.

A comparação dos algoritmos Rlearning e R'learning não é algo trivial, uma vez que o mapeamento do ambiente é baseado nas leituras dos sonares, sendo desta forma, um mapeamento não determinístico. Assim sendo, não se pode garantir que uma mesma ação em um mesmo estado, em tempos distintos, leve o agente para um mesmo estado seguinte. Portanto, não é possível ter uma comparação eficiente das performances de um robô em experimentos distintos.

O aprendizado foi acompanhado em gráficos alterações x tempo, no qual uma alteração é adicionada cada vez que um estado muda sua melhor ação. Desta forma, quando não se computar mais alterações é porque uma política ótima foi alcançada. Isto pode ser observado nos gráficos da Figura 11 que mostram as curvas de aprendizado para os algoritmos Rlearning e R'-learning. Embora sejam gráficos relativamente parecidos, pode-se observar que o algoritmo R'-learning converge para uma política de comportamento satisfatório antes do R-learning.

Desta forma, a nossa proposta de incorporação de lógica fuzzy no algoritmo R-learning, para mapear os estados do ambiente e calibrar o valor da recompensa imediata, propiciou melhorias no aprendizado da tarefa de navegação.

6 APLICAÇÃO UTILIZANDO R'-LEARNING

Para escolher uma aplicação real para o robô, considerou-se o fato do robô não depender de um mapa do ambiente. Desta forma, a tarefa implementada consiste em fazer com que o robô navegue por um ambiente desconhecido, procurando por pequenos objetos que possam ser recolhidos e levados a um local específico (lixeira).

Para implementar esta tarefa, foi escolhido o algoritmo R'learning, pois foi este que apresentou os melhores resultados no aprendizado da tarefa de navegação e por possuir um conjunto de variáveis de mais fácil configuração que os outros algoritmos implementados.

Esta aplicação foi desenvolvida em módulos, como é apresentado na Figura 7.

O módulo cálculo da força de repulsão é uma implementação parcial da proposta de Borenstein and Koren (1989). Neste módulo é calculada a força de repulsão como uma fusão das leituras dos sonares, também é calculada a força de atração entre o robô e a meta (utilizado somente no módulo de *planning*). A direção da força de repulsão foi utilizada por nós para representar os estados do ambiente que representam os obstáculos. Pode-se observar este mapeamento através da Figura 8. Esta forma de mapeamento reduz sensivelmente o número de estados do ambiente possibilitando que os estados sejam

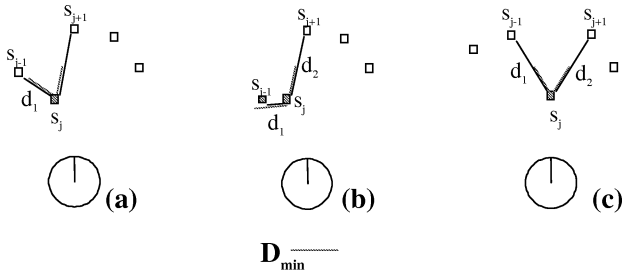

Figura 7: Reconhecimento de objetos: (a) objeto estreito, (b) objeto extenso e (c) objeto estreito pronto para ser coletado. Os quadrados em frente aos robôs representam as distâncias recebidas pelos 5 sonares frontais.

explorados em menos tempo. Apesar dos sonares não serem os melhores sensores para reconhecimento de objetos. Um objeto é reconhecido como "pequeno" sempre que somente um sonar detectá-lo (Figura 9). Um objeto só estará pronto para ser pego quando estiver exatamente na frente do robô. Desta forma, o robô deve fazer os giros necessários para colocar os objetos reconhecidos como estreitos exatamente à sua frente. Para levar o objeto coletado até a lixeira o robô utiliza a rotina de planejamento reativo, utilizando para isso as forças de repulsão, a força de atração com a meta e a rotina de seguir paredes usada para tirar o robô de "armadilhas" do ambiente (Borenstein and Koren, 1989).

Tabela 2: Recompensas imediatas para o robô

Estados	**Recompensas**
Passagem livre	10
Colisão	-100
risco de colisão	-100
levar para lixeira	100
Obstáculo	-50

As recompensas imediatas são atribuídas conforme o estado que o robô atinge, como é mostrado na tabela 2. Com o conjunto de ações proposto, conseguiu-se atingir os resultados esperados, ou seja, que os movimentos do robô se tornassem suaves ao desviar de obstáculos. Além disto, o robô conseguiu executar a tarefa proposta, isto é, pegar objetos pequenos e levá-los para uma lixeira. Para verificar o desempenho do robô nesta tarefa foram colocados 2 corpos de prova em um ambiente (Figura 10 (a)), onde o robô aprendeu a desviar de obstáculos. Após o aprendizado mais dois ambientes foram testados no simulador e o robô navegou neles normalmente sem precisar de um novo treinamento (Figura 10 (b) e Figura 10 (c)).

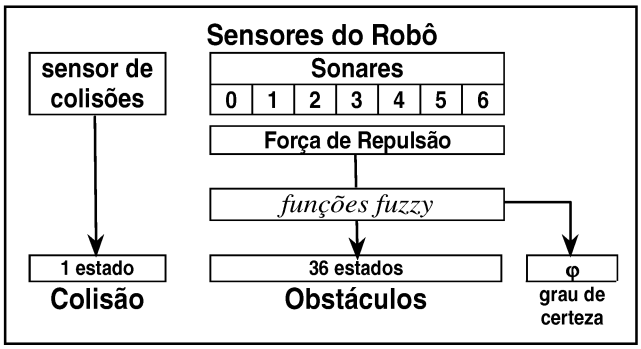

Figura 8: Mapeamento dos sensores do robô em estados do ambiente.

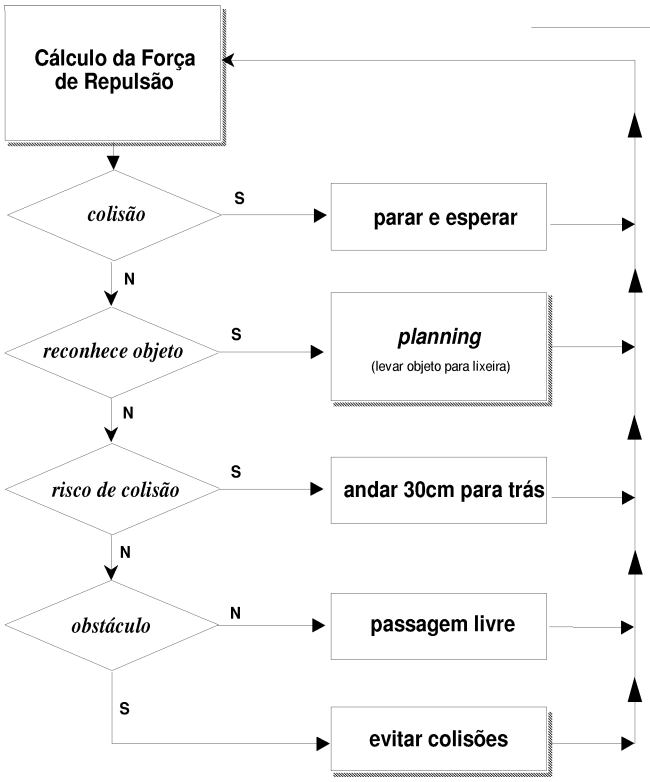

Figura 9: Representação dos módulos da aplicação utilizando aprendizado por reforço

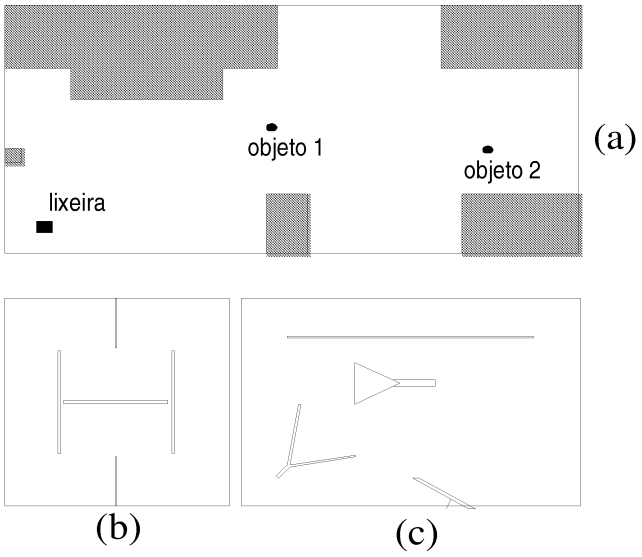

Figura 10: (a) Planta do ambiente onde o robô recolheu objetos e aprendeu a navegar; (b) e (c) ambientes do simulador.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve como meta a investigação de algoritmos de aprendizado por reforço, visando a navegação, em tempo-real, de robôs móveis em ambientes desconhecidos.

A abordagem de navegação proposta neste trabalho não utiliza um mapa do ambiente, fazendo com que o robô possa mudar de ambiente e continuar usando a mesma política de navegação, sem a necessidade de um novo treinamento. Dentro desta abordagem foram estudados três métodos de aprendizado por reforço no domínio de robôs móveis. Estes métodos foram comparados entre si, considerando a tarefa de navegação em um ambiente real.

Tendo como proposta a tarefa de navegação de robôs móveis, foi comparado neste trabalho, o desempenho dos algoritmos Qlearning, R-learning e Hlearning. Para esta tarefa, chegou-se à conclusão que o algoritmo R-learning apresentou melhores resultados, aprendendo a tarefa de navegação com um número menor de colisões. Observou-se que o Rlearning também possui um conjunto de variáveis de mais fácil manipulação que o Q-learning. Com algoritmo Q-learning também foram obtidos bons resultados, mas o ajuste da variável γ, até encontrar uma política ótima, é uma tarefa bastante exaustiva. Em outros trabalhos (Tadepalli and Ok, 1994), o algoritmo H-learning mostrou ter conseguido convergir para a política ótima com bem menos passos que os demais algoritmos. Neste trabalho, porém, não foi conseguida uma política de ações satisfatória para o desvio de obstáculos, provavelmente isto se deve ao próprio modelo MDP utilizado.

Além das implementações dos três algoritmos citados, foi proposta neste trabalho uma modificação no algoritmo Rlearning com a finalidade de classificar melhor os sinais de entrada. Esta modificação foi realizada através da incorporação de lógica *fuzzy* ao algoritmo R-learning, gerando um novo algoritmo denominado R'-learning. Este algoritmo apresentou melhores resultados que o R-learning, no domínio de navegação de robôs móveis, por calibrar o reforço imediato.

Além disto, uma aplicação para o robô Pioneer 1 foi realizada utilizando o algoritmo R'-learning. A aplicação consistiu em fazer com que o robô navegasse em um ambiente desconhecido, evitando colisões e procurando por pequenos objetos para serem colocados em uma lixeira. Para tanto, o mapeamento dos estados foi feito utilizando força de repulsão entre os objetos e o robô. Este mapeamento propiciou um número menor de estados do ambiente, por fundir as leituras dos sonares em uma única informação, permitindo que o robô aprendesse a navegar em menos tempo. Por outro lado, foi proposto um conjunto de ações que fez com que o robô tivesse movimentos mais suaves.

Assim sendo, a experiência obtida com a realização deste trabalho nos permite concluir que devido à incerteza no ambiente, pois estes algoritmos atuam em ambientes desconhecidos, existe necessidade de muitos ajustes tanto no modelo MDP quanto nas variáveis internas aos algoritmos.

Como continuidade deste trabalho pretende-se: Realizar experimentos de navegação de robôs em outros algoritmos de aprendizado por reforço, como o ARTDP, para que se possa determinar qual algoritmo é melhor para realizar uma tarefa específica. Comparar os resultados já obtidos utilizando uma política que comece com *exploração* dos estados e siga para uma política com 100% de *exploitação* das melhores ações, pois neste trabalho foi utilizada uma política constante durante todo o aprendizado e acreditamos que esta política variável possa melhorar o aprendizado de escolha de ações.

REFERÊNCIAS

Bagnell, J., Doty, K. and Arroyo, A. (1998). Comparison of Reinforcement Learning Techniques for Automatic Behaviour Programming, *Proceedings of the*

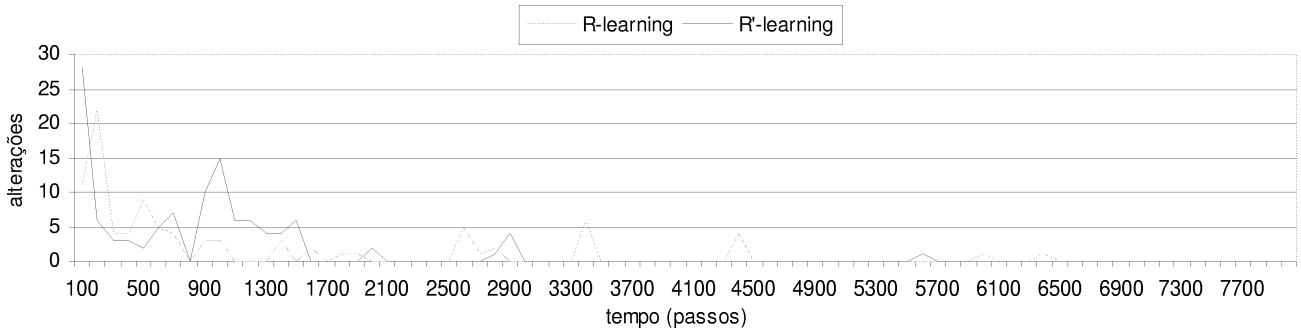

Figura 11: Número de alterações da política de ações.

CONALD, CMU-USA.

Barto, A. G., Bradtke, S. J. and Singh, S. P. (1993). Learning to act using real-time dynamic programming, *Artificial Intelligence*.

Bellman, R. (1957). *Applied Dynamic Programming*, Princeton University Press, Princeton, N.J.

Bertsekas, D. P. (1987). *Dynamic Programming: Deterministic and Stochastic Models*, Pentice-Hall, Englewood Cliffs, NJ.

Bertsekas, D. P. (1995). A counter example to temporal differences learning, *Neural Computation* **7**: 270–279.

Borenstein, J. and Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots, *IEEE Transactions on Systems, Man, and Cybernetics* **19**: 1179–1187.

Crites, R. H. and Barto, A. G. (1996). Improving elevator performance using reinforcement learning, *in* D. Touretzky, M. Mozer and M. Hasselmo (eds), *Neural Information Processing Systems 8*.

Faria, G. and Romero, R. A. F. (2000a). *Explorando o potencial de algoritmos de aprendizado com reforço em robôs móveis*, Master's thesis, Universidade de São Paulo.

Faria, G. and Romero, R. A. F. (2000b). Incorporating fuzzy logic to reinforcement learning, *Proceedings of the 9th IEEE International Conference on Fuzzy Systems*.

Haykin, S. (1994). *Neural Networks – A comprehensive Foundation*, IEEE Press and IEEE Computer Society Press, McMaster University, Hamilton, Ontario, Canada.

Howard, R. A. (1960). *Dynamic Programming and Markov Process*, The MIT Press, Cambridge, MA.

Kaelbing, L. and Littman, M. (1996). Reinforcement learning: A survey, *Journal of Artificial Intelligence Research* **4**: 237–285.

Konolige, K. (1997). *Saphira Software Manual*, ActivMedia.

Littman, M. L., Dean, T. and Kaelbling, L. P. (1994). Markov games as a framework for multi-agent reinforcement learning, *Proceedings of the Eleventh International. Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, pp. 157–163.

Mahadevan, S. and Connell, J. (1992). Automatic programming of behaviour-based robots using reinforcement learning, *Artificial Intelligence* (55): 311–365.

Mataric, M. J. (1994). Reward functions for accelerated learning, *in* W. W. Cohen and H. Hirsh (eds), *Proceedings of the Eleventh International Conference on Machine Learning*, Morgan Kaufman.

Mitchell, T. (1997). *Machine Learning*, McGraw Hill.

Puterman, M. L. (1994). *Markov Decision Process–Discrete Stochastic Dynamic Programming*, Inc. John Wiley & Sons, New York, NY.

Ribeiro, C. H. C. (1999). Aprendizado por reforço, Tutorial. CDROM do IV Congresso Brasileiro de Redes Neurais IV CBRN e V Escola de Redes Neurais V ERN.

Ribeiro, C., Reali, A. and Romero, R. (2001). Robôs móveis inteligentes: Princípios e técnicas, Capítulo de livro da I Jornada de Atualização em Inteligência Artificial - JAIA'2001, Anais do XXI Congresso da SBC.

Schaal, S. and Atkeson, C. (1994). Robot juggling: An implementation of memory based learning, *Control Systems Magazine 14* **.**

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards, *Machine Learning: Proceedings of the Tenth International Conference*, Morgan Kaufmann, San Mateo, CA.

Shaw, I. S. and Simões, M. G. (1999). *Controle e Modelagem Fuzzy*, 1st edn, Editora Edgar Blücher LTDA.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences, *Machine Learning* **3**: 9–44.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding, *in* D. S. Touretzky, C. M. MJozer and M. E. Hasselmo (eds), *Advances in Neural Information Processing Systems 8*, MIT Press, pp. 1038–1044.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA.

Tadepalli, P. and Ok, D. (1994). A reinforcement learning method for optimising undiscounted average reward, *Technical Report 94-30-01*, Department of Computer Science, Oregon State University.

Thrun, S. (1995). Learning to play the game of chess, *in* G. Tesauro, D. S. Touretzky and T. K. Leen (eds), *Advances in Neural Information Processing Systems 7*, The MIT Press.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*, PhD thesis, University of Cambridge.