DESEMPENHO DE ALGORITMOS DE APRENDIZAGEM POR REFORÇO SOB CONDIÇÕES DE AMBIGUIDADE SENSORIAL EM ROBÓTICA MÓVEL

Sildomar T. Monteiro*

Carlos H. C. Ribeiro*
carlos@comp.ita.br

*Divisão de Ciência da Computação, Instituto Tecnológico de Aeronáutica, Praça Mal. Eduardo Gomes, 50
CEP 12228-900 São José dos Campos - SP

ABSTRACT

We analyzed the performance variation of reinforcement learning algorithms in ambiguous state situations commonly caused by the low sensing capability of mobile robots. This variation is caused by violation of the Markov condition, which is important to guarantee convergence of these algorithms. Practical consequences of this violation in real systems are not firmly established in the literature. The algorithms assessed in this study were Q-learning, Sarsa and Q(λ), and the experiments were performed on a Magellan Pro^{TM}robot. A method to build variable resolution cognitive maps of the environment was implemented in order to provide realistic data for the experiments. The implemented learning algorithms presented satisfactory performance on real systems, with a graceful degradation of efficiency due to state ambiguity. The Q-learning algorithm accomplished the best performance, followed by the Sarsa algorithm. The Q(λ) algorithm had its performance restrained by experimental parameters. The cognitive map learning method revealed to be quite efficient, allowing adequate algorithms assessment.

KEYWORDS: Autonomous mobile robots, reinforcement learning, map learning, neural networks.

RESUMO

Analisamos a variação de desempenho de algoritmos de aprendizagem por reforço em situações de ambigüidade de estados comumente produzidas pela baixa capacidade sensorial de robôs móveis. Esta variação é produzida pela violação da condição de Markov, importante para garantir a convergência destes algoritmos. As conseqüências práticas desta violação em sistemas reais não estão avaliadas de maneira definitiva na literatura. São estudados neste artigo os algoritmos Q-learning, Sarsa e Q(λ), em experimentos realizados em um robô móvel Magellan Pro^{TM}. De modo a definir um verificador de desempenho para os algoritmos testados, foi implementado um método para criar mapas cognitivos de resolução variável. Os resultados mostram um desempenho satisfatório dos algoritmos, com uma degradação suave em função da ambigüidade sensorial. O algoritmo Q-learning teve o melhor desempenho, seguido do algoritmo Sarsa. O algoritmo Q(λ) teve seu desempenho limitado pelos parâmetros experimentais. O método de criação de mapas se mostrou bastante eficiente, permitindo uma análise adequada dos algoritmos.

PALAVRAS-CHAVE: Robôs móveis autônomos, aprendizagem por reforço, aprendizagem de mapas, redes neurais.

Artigo submetido em 11/02/03
1a. Revisão em 22/04/03; 2a. revisão em 02/09/03
Aceito sob recomendação do Ed. Assoc. Prof. Paulo E. Miyagi

1 INTRODUÇÃO

Métodos de aprendizagem por reforço (Sutton, R. S. e Barto, A. G. (1998)) tratam de situações onde um agente aprende por tentativa e erro ao atuar sobre um ambiente dinâmico. Desta maneira, não é necessária uma entidade externa que forneça exemplos ou um modelo a respeito da tarefa a ser executada: a única fonte de aprendizado é a própria experiência do agente, cujo objetivo formal é adquirir uma política de ações que maximize seu desempenho geral.

Aplicações reais envolvem sistemas observados por sensores que possuem limitações inerentes à sua construção mecânica, apresentando restrições de confiabilidade, ruído e não-uniformidade nas medidas. Isto faz com que o verdadeiro estado do processo dinâmico a ser controlado seja apenas indicado de forma grosseira. A ambigüidade decorrente da descrição incompleta dos estados anula a suposição de Markov, uma das bases teóricas mais importantes para a operação confiável de algoritmos de aprendizagem por reforço. Esta suposição estabelece que qualquer estado do sistema depende apenas do estado imediatamente anterior e da ação escolhida, e não da seqüência passada de estados e ações. A suposição de Markov pode ser teoricamente restabelecida por técnicas de aumento de estado (utilização de história de observações), mas na prática este aumento é freqüentemente inviável, por implicar aumento exponencial da cardinalidade do conjunto de estados discretizados.

Este artigo apresenta um estudo do desempenho de algoritmos de aprendizagem por reforço em um problema de navegação robótica, através de uma análise do seu comportamento sob condições realistas. Algoritmos de aprendizagem largamente estudados na literatura – a saber: Q-*learning*, Sarsa e Q(λ) – foram implementados em um robô Magellan Pro^{TM}. A tarefa de aprendizagem era sempre bastante simples: partindo de um ponto de referência inicial, o robô deveria aprender uma trajetória de navegação de modo a atingir um ponto alvo, e ao mesmo tempo, desviar dos obstáculos do ambiente.

Implementou-se, inicialmente, um algoritmo de aprendizagem de mapas cognitivos para adquirir modelos do ambiente, com o objetivo de prover os experimentos com dados realísticos. Além disso, como a construção de mapas é tarefa básica em robôs autônomos, julgamos que esta seria uma boa oportunidade para implementar um algoritmo eficiente de geração de mapas. Os mapas obtidos pelo robô real serviram então como base para estudos em simulação dos algoritmos de aprendizagem por reforço.

A seção 2 deste artigo contém a fundamentação teórica da pesquisa. A sub-seção 2.1 apresenta os Processos de Decisão Markovianos, base para o estudo dos algoritmos de aprendizagem por reforço. A sub-seção 2.2 apresenta a aprendizagem por reforço e os algoritmos testados. A sub-seção 2.3 apresenta o problema do *perceptual aliasing,* que ocorre quando algoritmos de aprendizagem por reforço são utilizados em situações de observação incompleta de estados, comum em domínios realistas. A sub-seção 2.4 apresenta brevemente o método para construção de mapas implementado. A seção 3 contém a descrição dos experimentos, mostrando os testes realizados e os resultados correspondentes. A sub-seção 3.1 apresenta o robô utilizado. A sub-seção 3.2 apresenta a primeira parte da pesquisa, que foi a obtenção dos modelos do mundo real através da geração de mapas cognitivos. A sub-seção 3.3 apresenta a implementação dos algoritmos de aprendizagem por reforço, e a seção 3.4 apresenta os resultados obtidos. Finalmente, a seção 4 apresenta uma discussão sobre os resultados alcançados na pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

A aprendizagem permite a um agente adquirir uma capacidade ou conhecimento que não estava disponível no estágio de projeto. Se uma estrutura (por exemplo, um controlador) para implementar uma capacidade não puder ser identificada ou projetada *a priori*, a aprendizagem é uma maneira de estabelecer esta competência.

A aprendizagem por reforço é um paradigma computacional de aprendizagem em que um agente aprendiz procura maximizar uma medida de desempenho baseada nos reforços (recompensas ou punições) que recebe ao interagir com um ambiente desconhecido (Ribeiro, C. H. C. (1999)). Mais especificamente, o agente atua em um ambiente formado por um conjunto de possíveis estados, e pode escolher ações dentro de um conjunto de ações possíveis. Ele recebe um valor de reforço cada vez que executa uma ação, indicando o valor imediato da transição de estado resultante. Ao longo do tempo, isto produz uma seqüência de pares estado-ação e respectivos valores de reforço. A tarefa do agente é aprender uma política de controle (seqüência de ações) que maximize a soma esperada destes reforços, descontando (usualmente de modo exponencial) as recompensas ou punições proporcionalmente ao seu atraso temporal.

2.1 Processos de Decisão Markovianos

A teoria de aprendizagem por reforço é teoricamente restrita a processos de decisão Markovianos, apesar de as idéias e métodos poderem ser aplicados de forma mais genérica.

Um ambiente satisfaz a propriedade de Markov se o seu estado resume o passado de forma compacta, sem perder a habilidade de prever o futuro. Isto é, pode-se predizer qual será o próximo estado e próxima recompensa esperada dado o estado e ação atuais (Sutton, R. S. e Barto, A. G. (1998)). Um processo de Markov é uma seqüência de estados, com a propriedade de que qualquer predição de valor de estado futuro dependerá apenas do estado e ação atuais, e não da seqüência de estados passados.

Um processo de aprendizagem por reforço que satisfaz a propriedade de Markov é chamado de processo de decisão Markoviano (MDP - *Markov Decision Process*). Se o espaço de estados e ações for finito, então ele é chamado de processo de decisão Markoviano finito, base para a teoria de aprendizagem por reforço, que assume o ambiente como sendo deste tipo.

Formalmente, um processo de decisão Markoviano é definido por um conjunto $\langle S, A, P, R\rangle$, onde temos: um conjunto finito de estados S do sistema, um conjunto finito de ações A, um modelo de transição de estados P, que mapeia os pares estado-ação em uma distribuição de probabilidades sobre o conjunto de estados, e, finalmente, uma função de recompensa R, que especifica o reforço que o agente recebe por escolher uma determinada ação $a \in A$ no estado $s \in S$. O estado s e a ação a atuais determinam a) o próximo estado s' de acordo com a probabilidade $P(s'|s,a)$, e b) a recompensa $r(s,a)$ associada.

Se o modelo de transição de estados for conhecido, técnicas de Controle Ótimo baseadas em Programação Dinâmica podem – ao menos em tese – ser utilizados para determinar uma política ótima de ações a ser seguida pelo agente (Bellman, R. (1957)). Alternativamente, aprendizagem por reforço é utilizada quando o modelo não está disponível (aprendizagem autônoma) ou quando existe apenas um modelo de simulação, que não permite a formulação analítica necessária para algoritmos de Programação Dinâmica. A relação teórica entre Controle Ótimo e aprendizagem por reforço pode ser encontrada em Singh, S. P. (1994).

2.2 Aprendizagem por Reforço

Aprendizagem por reforço preocupa-se com o problema de um agente aprender, por tentativa e erro, a atingir um objetivo interagindo com o seu ambiente. O domínio deve ser modelado como um MDP, e o agente e o ambiente interagem em uma seqüência discreta de passos no tempo, *t=0,1,2,....* O estado e a ação em dado instante ($s_t \in S$ e $a_t \in A$), determinam a distribuição de probabilidades para o estado s_{t+1} e o reforço r_t. O objetivo do agente normalmente é escolher ações de modo a maximizar uma soma descontada dos reforços subseqüentes:

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k} \tag{1}$$

onde a taxa de desconto $0 < \gamma \leq 1$ determina o peso temporal relativo dos reforços. Existem formulações alternativas para esta função de otimização (Sutton, R. S. e Barto, A. G. (1998)).

As escolhas das ações do agente são feitas a partir de uma função do estado, chamada política, $\pi : S \mapsto A$. O valor de utilidade de um estado, dada uma política, é o reforço esperado partindo do estado e seguindo a política:

$$V^{\pi}(s) = E_{\pi}\{R_t \,|s_t = s\} \tag{2}$$

e a política ótima de ações é aquela que maximiza o valor de estado:

$$V*(s) = \max_{\pi} V^{\pi}(s) \tag{3}$$

Existe sempre ao menos uma política ótima $\pi*$, que produz o valor de utilidade máximo em todos os estados $s \in S$.

Paralelamente a essas duas funções de valor de estado, existem duas funções de valor de ação,

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \,|s_t = s, a_t = a\} \tag{4}$$

e

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{5}$$

A partir de Q^*, pode-se determinar uma política ótima simplesmente como $\pi^*(s) = \arg\max_a Q(s,a)$.

2.2.1 Q-learning

O método de aprendizagem por reforço mais popular é o Q-*learning* (Watkins, C. J. C. H. e Dayan, P. (1992)). Trata-se de um algoritmo que permite estabelecer autonomamente uma política de ações de maneira interativa. Pode-se demonstrar que o algoritmo Q-*learning* converge para um procedimento de controle ótimo, quando a hipótese de aprendizagem de pares estado-ação Q for representada por uma tabela completa contendo a informação de valor de cada par. A convergência ocorre tanto em processos de decisão Markovianos determinísticos quanto não-determinísticos.

A idéia básica por trás do Q-*learning* é que o algoritmo de aprendizagem aprende uma função de avaliação ótima

sobre todo o espaço de pares estado-ação $S \times A$. A função Q fornece um mapeamento da forma $Q : S \times A \rightarrow V$, onde V é o valor de utilidade esperado ao se executar uma ação a no estado s. Desde que o particionamento do espaço de estados do robô e o particionamento do espaço de ações não omitam informações relevantes nem introduzam novas, uma vez que a função ótima Q seja aprendida o agente saberá precisamente que ação resultará na maior recompensa futura em uma situação particular s.

A função $Q(s, a)$, da recompensa futura esperada ao se escolher a ação a no estado s, é aprendida através de tentativa e erro segundo a equação a seguir:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_t + \gamma V_t(s_{t+1}) - Q_t(s_t, a_t)] \quad (6)$$

onde α é a taxa de aprendizagem, r é a recompensa, ou custo, resultante de tomar a ação a no estado s, γ é o fator de desconto e o termo $V_t(s_{t+1}) = \max_a Q(s_{t+1}, a)$ é a utilidade do estado s resultante da ação a, obtida utilizando a função Q que foi aprendida até o presente.

A função Q representa a recompensa descontada esperada ao se tomar uma ação a quando visitando o estado s, e seguindo-se uma política ótima desde então.

A forma procedimental do algoritmo Q-*learning* é:

Para cada *s,a* inicialize *Q(s,a)=0*

Observe *s*

Repita

- Selecione ação *a* usando a política de ações atual
- Execute a ação *a*
- Receba a recompensa imediata *r(s,a)*
- Observe o novo estado *s'*
- Atualize o item *Q(s,a)* de acordo com a equação (6).
- $s \leftarrow s'$

Até que critério de parada seja satisfeito

Uma vez que todos os pares estado-ação tenham sido visitados um número infinito de vezes, garante-se que o método gerará estimativas Q_t que convergem para o valor de Q(Watkins, C. J. C. H. e Dayan, P. (1992)). Na prática, a política de ações converge para a política ótima em tempo finito, embora de forma lenta.

2.2.2 Sarsa

O algoritmo Sarsa é uma modificação do algoritmo Q-*learning* que utiliza um mecanismo de iteração de política (Sutton, R. S. e Barto, A. G. (1998)). A função de atualização do algoritmo Sarsa obedece à equação a seguir:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)] \quad (7)$$

A forma procedimental do algoritmo Sarsa é similar à do algoritmo Q-learning. Idealmente, o algoritmo Sarsa converge para uma política e valor de função de ação ótimos assim que todos os pares estado-ação tenham sido visitados um número infinito de vezes e a política de escolha da próxima ação convirja, no limite, para uma política que utilize a melhor ação (ou seja, aquela que maximize a recompensa futura esperada).

2.2.2 Q(λ)

O algoritmo Q(λ), proposto em Peng, J. e Williams, R. J. (1996) é caracterizado por ser uma adaptação de uso de traços de elegibilidade para o algoritmo Q-*learning*.

Traços de elegibilidade são registros temporários da ocorrência de um evento, como por exemplo, a visita a um estado ou a execução de uma ação. O traço marca os parâmetros de memória associados com o evento como estados elegíveis para passar por mudanças no aprendizado. Quando um passo de aprendizado ocorre, apenas os estados ou ações elegíveis recebem o crédito pela recompensa ou a culpa pelo erro.

De um ponto de vista teórico, traços de elegibilidade são como uma ponte entre os métodos de Monte Carlo e de Diferenças Temporais, onde se enquadram o Q-*learning* e o Sarsa. Quando métodos de diferenças temporais são incrementados com traços de elegibilidade, eles produzem uma família de métodos atravessando um espectro que tem métodos de Monte Carlo em uma ponta e métodos de Diferenças Temporais na outra (Ribeiro, C. H. C. (1999)). Neste intervalo estão métodos que herdam vantagens de ambos os extremos, freqüentemente apresentando melhor desempenho.

Métodos de Monte Carlo podem apresentar vantagens para lidar com processos não-Markovianos, porque não atualizam estimativas baseados em outras estimativas (*bootstrapping*). A principal desvantagem destes métodos é a sua pesada carga computacional. Métodos usando traços de elegibilidade buscam portanto combinar a vantagem da rapidez relativa de aprendizado dos

métodos de Diferenças Temporais e a capacidade de lidar com reforços atrasados ou observabilidade parcial dos métodos Monte Carlo. O algoritmo Q(λ) é o seguinte:

Para cada *s,a*: $Q(s,a) \leftarrow 0$e $Tr(s,a) \leftarrow 0$

Repita

Inicialize *s*

Repita

- $s \leftarrow$ estado atual
- Selecione a ação a^*
- Execute a ação a*
- Receba a recompensa imediata *r*
- Observe o novo estado *s'*
- $e' \leftarrow r + \gamma \max_{a'} Q(s',a') - Q(s,a^*)$
- $e \leftarrow r + \gamma \max_{a'} Q(s',a') - \max_a Q(s,a^*)$
- Para todo par *(s,a)* faça:

$$Tr(s,a) \leftarrow \gamma\lambda Tr(s,a)$$
$$Q(s,a) \leftarrow Q(s,a) + \alpha Tr(s,a)e$$

- $Q(s,a^*) \leftarrow Q(s,a^*) + \alpha e'$
- $Tr(s,a) \leftarrow Tr(s,a) + 1$

Até que critério de parada seja satisfeito

O fator *Tr(s,a)* é o traço de elegibilidade do par estado-ação (s,a). Observe que o algoritmo realiza atualizações sobre todos os pares estado-ação, sendo que para o par (s,a) visitado é feita uma atualização Q-learning usual. Para este algoritmo, diferentemente do que ocorre para Q-*learning* e Sarsa, não há prova de convergência para o valor correto de Q^* sob uma política arbitrária que tente todas as ações em todos os estados. Entretanto, a estratégia óbvia de reduzir gradualmente o valor de λ ou de baixar gradualmente o grau de aleatoriedade da escolha da ação a^* no decorrer do aprendizado freqüentemente permite convergência para políticas sub-ótimas de boa qualidade (Peng, J. e Williams, R. J. (1996)).

2.2.3 Dyna

O termo *Dyna* foi introduzido em Singh, S. P. e Sutton, R. S. (1996), e define uma técnica simples para integrar funções de aprendizado, planejamento e atuação. O agente interage com o ambiente, gerando experiências reais. Estas experiências são utilizadas para melhorar diretamente as funções de valor e política de ações (através de algum método de aprendizagem por reforço) e aperfeiçoar um modelo do ambiente, que o agente pode usar para prever como o ambiente responderá a suas ações. As experiências originárias de simulação sobre este modelo são então utilizadas para melhorar as funções de valor e política de ações (planejamento sobre o modelo).

O método de aprendizagem por reforço utilizado em associação com o *Dyna* pode ser qualquer um dos discutidos nas seções anteriores. Nos experimentos descritos neste artigo, *Dyna* foi utilizado com a intenção de aumentar a velocidade de aprendizado dos algoritmos de aprendizagem por reforço.

Após cada transição $s_t, a_t \rightarrow s_{t+1}, r_t$, o algoritmo Dyna armazena em uma tabela, para o valor de (s_t, a_t), a transição observada $(s_{t+1,}, r_t)$. Durante o planejamento, o algoritmo escolhe amostras aleatórias de pares estado-ação que foram experimentados anteriormente, ou seja, contidos no modelo. A seguir, realiza experiências simuladas nestes pares estado-ação selecionados. Finalmente, é aplicada uma atualização baseada em um método de aprendizagem por reforço sobre essas experiências simuladas, como se elas tivessem realmente ocorrido. Tipicamente, o mesmo método de aprendizagem por reforço é utilizado tanto para o aprendizado a partir da experiência real quanto para o planejamento das experiências simuladas.

2.3 Ambigüidade Sensorial

Um problema comum em aplicações de robótica é que, freqüentemente, o estado não é totalmente observável, ou seja, não é sempre conhecido qual o estado atual em que o sistema está. Nas situações práticas, os sensores do agente fornecem apenas informações parciais (e com freqüência ruidosas), que não conseguem perceber o estado real do ambiente. Este fato implica a quebra da condição de Markov.

A observabilidade parcial resulta no que é denominado *perceptual aliasing* (Chrisman, L. (1992)), situação na qual dois ou mais estados são perceptualmente idênticos, mas necessitam de respostas (ações) diferentes. Um agente que aprende o seu comportamento como uma função das percepções imediatas dos sensores para escolher uma ação, estará sujeito aos efeitos de *perceptual aliasing*.

Problemas envolvendo *perceptual aliasing* são conhe-

cidos como Processos de Decisão Markovianos Parcialmente Observáveis (POMDP – *Partially Observable Markov Decision Process*). Várias pesquisas vêm sendo realizadas para resolver esta classe de problemas e existem vários métodos sub-ótimos, geralmente utilizando memória ou atenção para este fim (Lovejoy, W. S. (1991), Chrisman, L. (1992)).

No contexto de aprendizagem por reforço, o processo de aquisição de valores de utilidade (V ou Q) através do recebimento de reforços pelo agente gera uma complicação produzida pela observabilidade parcial: mesmo que uma dada observação defina perfeitamente um estado, se a próxima observação de estado for equivocada, o erro no reforço da ação poderá ser propagado para estados ou pares estado-ação correspondentes a visitações anteriores. Este processo de propagação de erro corresponde a um efeito típico do problema de *perceptual aliasing*. Por exemplo, um robô móvel normalmente precisa percorrer longas distâncias dentro de ambientes como salas e escritórios para executar uma tarefa, mas devido a erros na sua percepção de localização, regiões diferentes do ambiente podem parecer iguais. Se a ação esperada do robô nesses locais for a mesma, não há problema, mas existe o caso em que a ação é particular a cada local. Neste caso, haverá um erro no valor de utilidade atualizado, propagado para estados vizinhos e prejudicando o aprendizado.

2.4 Construção de Mapas

Esta seção trata de um dos maiores objetivos da pesquisa de robôs móveis, a criação de mapas a partir de dados sensoriais locais coletados à medida que o robô se move por um ambiente desconhecido. Utilizando algoritmos para aprendizagem de mapas, pode-se adquirir um modelo do mundo ao redor do robô.

Existem muitos tipos diferentes de mapas usados para localização de robôs em ambientes fechados, baseados na forma da informação sensorial e nos requisitos de representação da localização. Pesquisas na área produziram duas abordagens principais: o modelo métrico e o modelo topológico, ou ainda uma combinação das duas.

O modelo métrico, ou geométrico, permite a construção de um modelo que é uma representação geométrica do mundo. Um exemplo desta abordagem é um sistema de grades de ocupação (Elfes, A. (1987)), na qual grades de duas dimensões espaçados de forma simétrica apresentam a probabilidade de ocupação de cada célula da grade em correspondência àquela área no mundo. Neste modelo, as grades de ocupação distinguem lugares diferentes baseado na posição geométrica do robô dentro de uma estrutura de coordenadas globais. A posição do robô é estimada de maneira incremental, baseado na informação de odometria e nas leituras sensoriais do robô. A resolução da grade deve ser tal que o tamanho de célula seja suficiente para capturar todos os detalhes importantes do mundo.

Já o modelo topológico é mais qualitativo. Consiste de um grafo no qual os nós representam regiões sensorialmente diferentes do mundo e os arcos indicam relações espaciais entre eles. Neste modelo, a posição do robô relativa ao modelo é determinada principalmente por pontos de referência ou características sensoriais distintivas. A resolução do mapa topológico é proporcional ao grau de complexidade do ambiente.

Para os experimentos relatados neste artigo, foi utilizado um método rápido e eficiente para aquisição de mapas, que permite a construção de mapas de fácil utilização e com parâmetros ajustáveis, para comportar os testes dos algoritmos de aprendizagem por reforço. Utilizou-se integralmente o enfoque proposto em Arleo, A., Millán, J. R. e Floreano, D. (1999). Trata-se de um algoritmo para aprendizado eficiente de mapas de resolução variável para navegação autônoma de robôs em ambientes fechados, combinando os paradigmas métrico e topológico para construção de mapas (mapas cognitivos).

O mapa gerado consiste de um conjunto Pde partições de resolução variável, ou seja, o ambiente é dividido em sub-áreas $p \in P$ de diferentes tamanhos, representando regiões sensorialmente homogêneas correspondentes a modelos locais que representam o conhecimento do robô sobre as regiões. O mapa resultante é uma representação reduzida da estrutura geométrica do mundo tal como "visto" pelo robô, permitindo otimização no uso dos recursos de memória e tempo. Nenhum sistema de visão computacional é usado: apenas *encoders,* para estimação de posição por odometria, e sonares, para detecção de obstáculos e paredes.

O processo de aprendizagem do mapa pode ser descrito por um algoritmo cíclico composto por atividades de exploração e atualização do modelo. O robô explora o ambiente continuamente, baseado no conhecimento adquirido, e apenas interrompe a exploração para incorporar um obstáculo desconhecido ao modelo, atualizando a resolução das partições:

Início

Repetir

1. *Exploração: Seleção do próximo alvo.*
   O robô inicia o processo de exploração do ambiente

escolhendo uma trajetória aleatória e seguindo em linha reta, pois inicialmente o ambiente é desconhecido e P é vazio. Em outras situações, a região menos explorada é escolhida como alvo. Utiliza-se uma técnica de exploração baseada em uma estimativa de valor de utilidade de exploração, que é uma função heurística de valor real que mede o quanto uma partição merece ser explorada, calculada para cada partição e sendo escolhida aquela que maximiza o valor da função de utilidade. Utiliza-se a técnica de exploração baseada em contagem com deterioração (Thrun, S. (1998)), que fornece um valor de utilidade mais elevado para as partições que foram visitadas menor quantidade de vezes e há mais tempo.

2. *Planejamento: Cálculo de uma trajetória para alvo através das partições atuais.*

   O planejador deriva um grafo topológico a partir do conjunto de partições P atuais. Partindo do nó correspondente à partição atual, o planejador procura no grafo o menor caminho que leva ao nó associado ao alvo.

3. *Ação: Realiza movimento do robô através dos atuadores.*

   Uma vez determinado o caminho ótimo, um planejador de baixo nível calcula a trajetória do robô entre partições adjacentes no caminho. Se o robô tiver que se mover de uma partição i para uma partição adjacente j, e l é a partição limite entre eles, o robô se move, primeiro, paralelo a l até chegar em frente ao seu ponto central. Então, ele se move perpendicularmente a l até cruzar o limite. Todas as trajetórias seguidas pelo robô são retas paralelas aos eixos x e y do ambiente, movimentos simples que permitem minimizar erros de odometria no sistema de navegação.

4. *Se robô atinge o alvo, explora a região exaustivamente.*

5. Atualização do mapa: Se incoerência foi detectada (presença de novo obstáculo), robô interrompe exploração e modela o obstáculo desconhecido:

   5.1. Aproxima-se do obstáculo.

   5.2. Enquanto robô não visita um canto conhecido do obstáculo:

      5.2.1. Alinha-se com um dos limites do obstáculo.

      5.2.2. Analisa limite e o aproxima à uma linha reta.
      Esta aproximação é feita utilizando uma rede neural para gerar uma grade local de ocupação a partir das leituras dos sensores, seguida de aplicação de um método de interpolação para gerar a reta. Detalhes podem ser encontrados na seção 3.2.1.

      5.2.3. Move-se até o final do limite.
      O final do limite é detectado utilizando-se apenas as leituras cruas dos sensores. Sempre que o robô atinge o final de um limite, ele verifica se está encontrando a linha que modela a primeira borda do obstáculo. Se este for o caso, ele gira para alinhar com o novo limite, modela o limite e continua até o fim da borda. Se ele encontrar a linha que aproxima a segunda borda do obstáculo, ele considera o obstáculo inteiro como modelado, e sai do loop 5.2.

      5.2.4. Detecta canto.
      Neste ponto, o robô calcula a intersecção entre a linha reta atual e a anterior para identificar o canto visitado.

      5.2.5. Memoriza o canto do obstáculo.

   5.3. Aumenta resolução da partição.
   Uma vez que todos os cantos de um obstáculo tenha sido memorizado, é possível aumentar a resolução das partições P para modelar o novo obstáculo. Cada novo canto é conectado à borda perpendicular mais próxima de uma das partições existentes, criando, desta forma, partições retangulares.

   5.4. Atualiza o banco de dados de experiências.

   5.5. Remove partições redundantes do conjunto de partições.
   Partições adjacentes são redundantes se ambas representarem obstáculo ou espaço livre e puderam ser fundidas para produzir uma partição retangular.

   5.6. Abstrai novo grafo topológico.

Até que mapa esteja totalmente construído.

Fim.

O algoritmo utiliza estruturas simples, o que permite sua implementação em um ambiente para aprendizagem de mapa em tempo real, utilizando recursos computacionais modestos. Esta característica é atestada pela baixa complexidade do conjunto de partições de resolução variável (que permite uma economia de recursos de memória e tempo de processamento) e pela simplicidade dos grafos topológicos (que permite um planejamento de caminho com facilidade).

Existem algumas limitações no algoritmo. Primeiramente, há uma suposição de obstáculos paralelos ou perpendiculares entre si, de modo a garantir a baixa complexidade ao se tratar apenas de partições retangulares, geometricamente simples. Segundo, a capacidade de auto-localização do robô precisa ser boa. O sistema de orientação e posicionamento utilizado nos experimentos relatados neste artigo é baseado apenas em odometria, que apresenta uma grande imprecisão, compensada por algumas estratégias adotadas para melhorar a qualidade das medidas de distância (compensação *off-line* de erros sistemáticos na modelagem de limites de obstáculos e geração de trajetórias retas no planejamento e ação). E finalmente, o ambiente a ser modelado pressupõe-se estático.

Maiores detalhes sobre o algoritmo de construção de mapas podem ser encontrados em Arleo, A., Millán, J. R. e Floreano, D. (1999).

3 RESULTADOS EXPERIMENTAIS

Os resultados foram obtidos utilizando como base o robô Magellan Pro^{TM}(IS Robotics, Inc. (2000)) e o simulador Saphira. Os experimentos para aprendizagem dos mapas cognitivos foram realizados no robô real, e a partir dos mapas adquiridos os algoritmos de aprendizagem por reforço foram implementados e testados em simulação.

3.1 Características do Robô Magellan Pro^{TM}

O Magellan Pro^{TM}(figura 1) é um robô móvel para ambientes fechados. Tem formato cilíndrico, é de tamanho compacto (40,6 cm de diâmetro e 25,4 cm de altura) e relativamente leve, apresentando alto desempenho e qualidade. Possui 16 sonares, uniformemente distribuídos ao longo de seu perímetro, o que permite uma percepção de 360 graus do ambiente. É dotado de dois motores que permitem movê-lo para frente, para trás e girar. Possui, ainda, um computador Pentium-III 400 MHz e 32MB RAM a bordo, utilizando o sistema operacional Linux para controle do robô. Permite comunicação sem fio com outro computador, via link de RF, podendo ser programado, operado e monitorado à distância.

3.2 Implementação do Gerador de Mapas

Para melhor apresentar os resultados referentes à implementação do algoritmo de aquisição de mapas, dividiu-se esta seção em duas partes, a primeira apresentando a implementação de mapas locais ao redor do robô utilizando

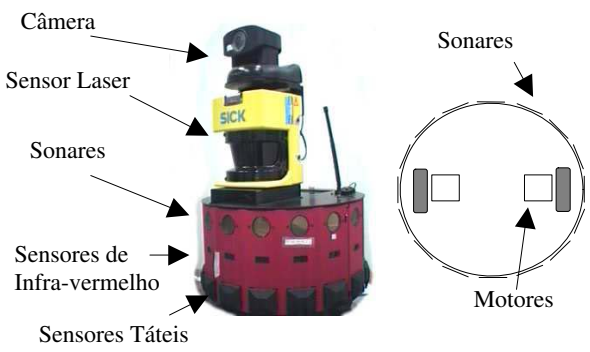


Figura 1: Robô Magellan Pro^{TM}e visão esquemática. Somente os sonares foram utilizados para os experimentos.

uma rede neural que realiza a interpretação dos dados dos sonares, e a segunda parte descrevendo o aprendizado dos mapas cognitivos globais propriamente ditos.

3.2.1 Obtenção de Mapas Locais

O treinamento da rede neural para geração de mapas locais foi baseado em um banco de dados de leituras reais de sonares. A grade para o mapa local consiste de um arranjo quadrangular de 14 x 14 células, cada uma das quais correspondendo a um quadrado de 10cm x 10 cm. A grade é local ao robô, que sempre ocupa suas 16 células centrais.

Os sonares foram os sensores escolhidos, por suas características de alcance de medida de até 4 metros, adequado para o tamanho de grade escolhido. Desprezando erros de manufatura, o arranjo simétrico de sonares permite que a aquisição do mapa local resulte do treinamento da rede neural a partir do conjunto de dados obtidos para um único quadrante da grade local. Uma representação em tamanho natural do primeiro quadrante da grade local foi então preparada, e montada sobre o piso da sala (ambiente) a mapear, com alinhamento perpendicular às paredes. Este modelo visual serviu como régua de referência para o obstáculo móvel – um objeto cilíndrico de 10 cm de diâmetro e 60cm de altura.

Foram obtidas diversas situações de leituras de sensor (figura 2), posicionando o obstáculo em diferentes células da grade e medindo: a) a distância entre a célula e o robô; e; b) o ângulo entre a célula e o sensor mais próximo. Do conjunto de 45 possíveis células para o posicionamento do obstáculo, 10 amostras de leituras de sonares – 5 com o obstáculo ocupando a célula e 5 com a célula vazia, correspondendo, respectivamente, à pro-

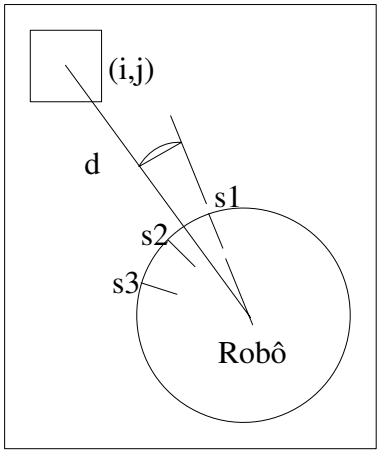


Figura 2: Leituras dos sonares (s1,s2), distância d e ângulo θ para uma célula cujo centro está posicionado nas coordenadas (i,j).

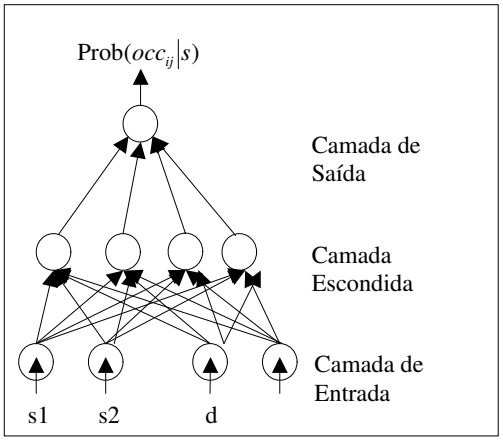


Figura 3: Rede neural que interpreta as leituras dos sensores como uma probabilidade de ocupação.

babilidades de ocupação 1 e 0 – foram obtidas. Observações empíricas mostraram que apenas os dois sonares mais próximos de uma célula ocupada precisam ser considerados. Esta observação contrasta com experimentos relatados em Thrun, S. (1998), que usa 4 sonares, e também com aqueles em Arleo, A., Millán, J. R. e Floreano, D. (1999), que usam 3 sonares. As diferenças são provavelmente causadas por dois fatores: o arranjo dos sonares – que depende do robô utilizado – e o tipo de obstáculo usado para o treinamento. Os resultados relatados aqui mostram que duas leituras podem ser suficientes para ambientes simples compostos por paredes e obstáculos: a aproximação realizada pela rede neural, seguida por integração sensorial ao longo do tempo e aproximação por linhas retas compensam a imprecisão e erros inerentes à detecção por sonares.

A rede neural treinada é uma arquitetura *feedforward* com quatro neurônios na camada escondida, conforme ilustrado na figura 3. Foram tentadas diversas configurações de rede, iniciando pelas mais simples com apenas um neurônio na camada escondida, e acrescentando-se mais neurônios até o limite de nove, e depois mais camadas escondidas até o limite de três. As entradas são as leituras normalizadas dos dois sonares mais próximos da célula, o ângulo entre a célula e o sonar mais próximo e a distância da célula ao robô. A saída produzida é a probabilidade de ocupação da célula. A função utilizada para a ativação dos neurônios foi a função logística. Para os neurônios da camada escondida utilizou-se uma função logística simétrica com valores de saída no intervalo [-0.5, +0.5]: $\sigma(y) = \frac{1}{1+e^{-y}} - 0.5$. Para a saída da rede utilizou-se a função logística assimétrica produzindo valores no intervalo [0,1]: $\sigma(y) = \frac{1}{1+e^{-y}}$, indicando a probabilidade de ocupação da referida célula para o dado padrão sensorial.

O treinamento foi conduzido de modo não-interativo (*off-line*), usando-se o algoritmo *backpropagation* (Haykin, S. (1999)) e os seguintes parâmetros:

- Inicialização aleatória de pesos na faixa ± 0.001;
- Taxa de aprendizado = 0.01;
- Variação máxima de magnitude dos pesos = 1.75;
- Fator de momento = 0.01;
- Fator de decremento de pesos = -0.001.

O robô utiliza a rede neural enquanto acompanha os limites do obstáculo. Leituras consecutivas das interpretações neurais dos sensores podem ser integradas no tempo com o objetivo de melhorar a confiabilidade da grade de ocupação. Sendo as leituras consecutivas dos sonares l^1, l^2, ..., l^M, a probabilidade de ocupação da célula de coordenadas (i, j) é Prob(oc_{ij} | l^1, l^2, ..., l^M). A integração no tempo é obtida aplicando-se a regra de Bayes. Tem-se:

$$Prob(oc_{ij}|l^1, l^2, \cdots, l^M) = 1 - \left(1 + \prod_{m=1}^{M} \frac{Prob(oc_{ij}|l^m)}{1 - Prob(oc_{ij}|l^m)}\right)^{-1}$$

A interpretação neural pode compensar eventuais erros devido à reflexão de sinal do sonar, e a integração no

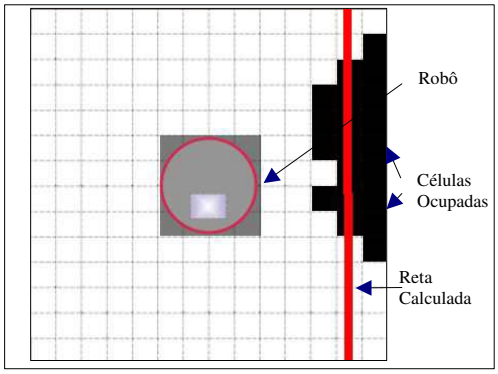


Figura 4: Grade de ocupação, interpretação neural dos sensores, integração no tempo e aproximação por reta.

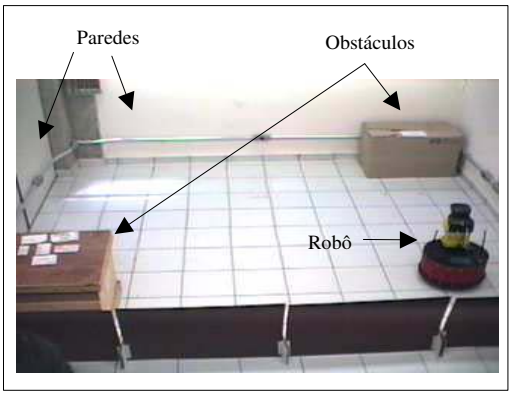


Figura 5: Ambiente para experimentos com robô Magellan.

tempo de interpretações consecutivas produz uma reconstrução aceitável apesar da limitada informação sensorial produzida pelos sonares.

A partir da grade de ocupação local obtida, os limites do obstáculos são aproximados por uma linha reta que será utilizada para construir o conjunto de partições P. Para calcular esta linha utilizou-se uma versão do método χ^2. Mais detalhes são encontrados em Arleo, A., Millán, J. R. e Floreano, D. (1999).

A figura 4 mostra um exemplo de detecção local de uma parede ao lado direito do robô. Neste caso, a reta calculada apresentou um erro de menos de 10cm em relação à posição real da parede. Note que algumas células à direita não foram identificadas pela rede neural como ocupadas. O processo de integração ao longo do tempo combinado à aproximação por linha reta, porém, foi suficiente para produzir uma aproximação com razoável precisão para a posição da parede. Em situações reais de funcionamento, a precisão do cálculo pode se tornar pior, em função de erros de posicionamento angular do robô (não ortogonalidade em relação aos obstáculos) e imprecisões de leituras de sonar.

3.2.2 Obtenção dos Mapas Cognitivos Globais

Os experimentos foram conduzidos em uma sala de 3m X 3,5m, preparada com obstáculos, conforme mostra a figura 5.

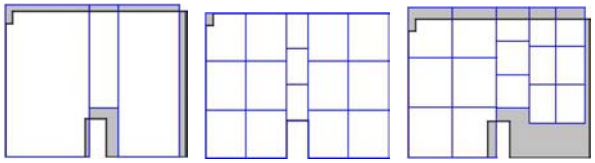
Adotou-se um procedimento de compensação para diminuir os erros na leitura de odometria. Verificou-se que ao seguir linhas retas o erro cometido pelo robô era pequeno, porém, ao realizar giros de 90 graus, o erro existente entre o ângulo calculado por odometria e o efetivamente obtido era considerável. Este erro violava a suposição de ortogonalidade aos eixos do ambiente.

Foram então realizados n testes em que o robô deveria realizar giros de 90 graus. Observou-se o erro entre o ângulo lido por odometria e o ângulo real. Então, calculando a média sobre todos os testes realizados temos uma estimativa da incerteza esperada do ângulo: $e = (1/n)\sum_{i=1}^{n} e_i$. Este fator foi utilizado para compensar o erro.

Decidiu-se, ainda, dividir as partições obtidas em partições menores (subparticionamento), permitindo uma observação mais detalhada do ambiente e um melhor aprendizado de política de ações. O critério adotado foi dividir as partições originais nos eixos x e y, gerando partições menores, mantendo os limites da partição original e observando a seguinte regra: uma subpartição não poderia ter tamanho, em cada eixo, menor que três vezes o raio do robô, uma vez que o comando de movimento avança o robô a uma distância de um raio a cada ação "ir adiante" (ver seção 3.3 para a definição das ações que podem ser executadas pelo robô).

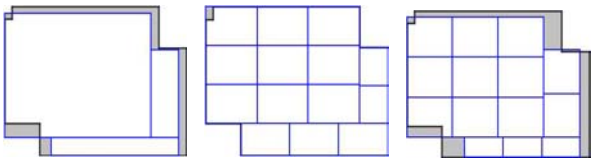
Para permitir uma comparação de desempenho dos algoritmos de aprendizagem por reforço, foram definidas três versões de particionamento para cada mapa: a) melhor particionamento original obtido no mundo real em várias tentativas de mapeamento, sem subparticionamento; b) particionamento perfeito com base nas partições obtidas pelo robô e subparticionamento; e c) pior mapa obtido utilizando o subparticionamento. Desta forma obteve-se mapas 2, 3 e 4 e respectivas partições, mostrados nas figuras 6, 7 e 8 (o mapa 1, sem obstáculos, foi usado apenas em testes preliminares). Os limites do ambiente são representados por linhas pretas, e as partições obtidas são traçadas em linhas mais claras. A área em cinza é a área mal classificada.

Uma forma de medir quantitativamente a qualidade do



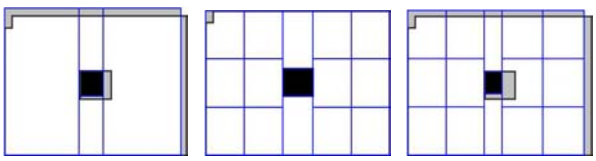
(a) 2A (b) 2B (c) 2C

Figura 6: Mapa 2: sala com um obstáculo lateral. Versões de particionamento: (a) 2A (Original), (b) 2B (Perfeito), (c) 2C (Pior).



(a) 3A (b) 3B (c) 3C

Figura 7: Mapa 3: sala com dois obstáculos nos cantos. Versões de particionamento: a) 3A (Original), b) 3B (Perfeito), c) 3C (Pior).



(a) 4A (b) 4B (c) 4C

Figura 8: Mapa 4: sala com um obstáculo central. Versões de particionamento: a) 4A (Original), b) 4B (Perfeito), c) 4C (Pior).

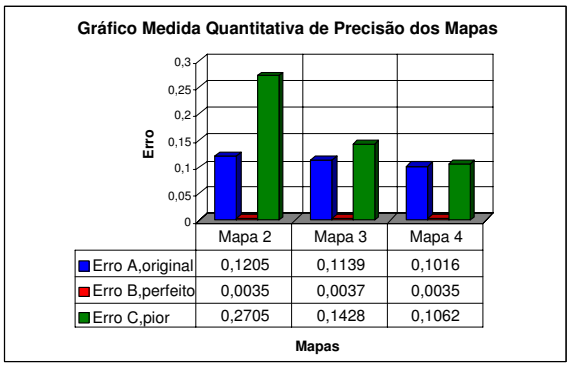


Figura 9: Erro medido para cada um dos mapas testados

mapa aprendido é calcular a fração entre a área mal classificada e a área total do ambiente. Partições mal classificadas são aquelas áreas livres classificadas como ocupadas, ou vice-versa. Seja a soma das superfícies mal classificadas A_e e a superfície total do ambiente A_{tot}. Assim, o erro será: $e = A_e/A_{tot}$. Aplicando aos mapas esta medida, obteve-se os dados da figura 9.

Observe que, mesmo no caso dos mapas tipo B, existe um pequeno erro, porém menor que 0,4% da área total. Este erro provém de uma pequena coluna localizada no canto da sala que, pelo seu tamanho, não foi observada pelo robô. Manteve-se o obstáculo, na representação simulada, apenas por fidelidade ao ambiente real. Observe, também, que a diferença existente entre o melhor mapa, tipo A, e o pior, tipo C, foi maior no mapa 2, com apenas um obstáculo, sendo que no mapa 3 a diferença diminuiu e no mapa 4 foi muito pequena. Este comportamento foi causado não pela complexidade maior de um ambiente em relação ao outro, mas devido ao processo de aprimoramento do programa que implementa o algoritmo de aprendizado de mapas cognitivos. A seqüência de desenvolvimento natural iniciou com um mapa sem obstáculos, passando para um obstáculo, e assim por diante, desta forma, os mapas seguintes foram construídos de forma mais precisa, uma vez que o programa já estava bem ajustado e testado. É evidente que a complexidade do ambiente afeta a qualidade do mapa aprendido, mas a diferença entre várias tentativas de aquisição do mapa é pequena ao se utilizar a mesma versão do programa, salvo mudanças nos parâmetros do algoritmo.

3.3 Implementação dos Algoritmos de Aprendizagem por Reforço

Decidiu-se realizar testes com os algoritmos de aprendizagem por reforço em simulação, e não no robô real. Isto foi feito em virtude dos erros de odometria cumu-

lativos observados durante o processo de aprendizado dos mapas. Sabe-se que os algoritmos de aprendizagem por reforço necessitam interagir bastante com o ambiente, visitando diversas vezes os pares estado-ação do sistema, para que se observe uma convergência satisfatória dos algoritmos. Como o sistema de localização utilizado é baseado apenas em odometria, a exploração do ambiente exigida pelos algoritmos de aprendizagem acrescentaria aos experimentos um erro cumulativo de observação de estados, desvirtuando o objetivo principal da pesquisa que era verificar a sensibilidade dos algoritmos à qualidade dos mapas.

O simulador Saphira (Konolige, K. e Myers, K. L. (1996)) foi escolhido para realizar a simulação. O Saphira compreende uma aplicação cliente para robôs móveis e um ambiente de desenvolvimento, tendo sido desenvolvido originalmente para permitir o controle e desenvolvimento de software para o robô móvel Pioneer. Os fatores que levaram à escolha deste simulador foram: a) reproduz com fidelidade os erros das leituras dos sonares e dos codificadores das rodas e b) aceita alterar os parâmetros do robô a ser simulado, permitindo a simulação de outros robôs com características físicas diferentes das do robô Pioneer.

Os experimentos foram portanto realizados no simulador Saphira, utilizando os parâmetros do robô Magellan Pro^{TM}e os mapas cognitivos obtidos no robô real. A tarefa a ser aprendida pelo robô foi aproximar-se de um alvo, e ao mesmo tempo desviar dos obstáculos e paredes.

A figura 10 ilustra o ambiente de simulação para um dos casos testados (mapa 2). O alvo fornece um reforço ao robô inversamente proporcional à distância. Na figura, os nomes "corr 1", "corr 3", etc., são representações das paredes e obstáculos da sala real, utilizados como pontos de referência para o simulador. O Saphira simula os erros de leitura de sonar e odometria, replicando o problema de estimação de localização que teríamos no robô real. No entanto, este simulador conta com um algoritmo específico que permite diminuir os erros de odometria utilizando pontos de referência no ambiente.

Os estados do robô para os testes dos algoritmos foram definidos utilizando as partições dos mapas cognitivos gerados com o robô no ambiente real. Um estado para o robô foi definido como sendo a combinação de duas informações: a sua localização (partição em que ele se encontra), e a sua direção (para qual lado ele está virado: norte, sul, leste ou oeste).

As ações possíveis para o robô em cada estado, são: "ir adiante", "virar à direita"e "virar à esquerda". A ação

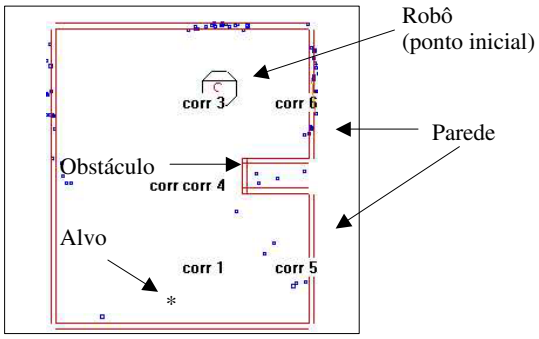


Figura 10: Ambiente Saphira. As indicações corr1, etc. são indicações automáticas do simulador, e não têm relevância para os experimentos e resultados.

Tabela 1: Parâmetros de aprendizado

Parâmetro	**Valores**
Reforços	$r = -1$ quando o robô se aproximar demais ou bater em um obstáculo; ou
	$0 \leq r \leq 1$, obedecendo a uma função linear decrescente com valor proporcional à distância em relação ao alvo, em qualquer outro caso. Ou seja, quanto mais distante do alvo, menor o valor de reforço.
Desconto temporal	$\gamma = 0,95$
Taxa de aprendizado	$\alpha = 0,9$
Experiências atualizadas (*Dyna*)	30 experiências por passo
Q(λ): Traço de elegibilidade	$\lambda = 0,5$

"ir adiante"movimenta o robô a uma distância correspondente ao raio de sua base física. A ação "virar à direita"corresponde a um giro de 90 graus, para a direita, e a ação "virar à esquerda", gira o robô 90 graus para a esquerda. Estas ações também são aquelas definidas para o robô real, quando da construção do mapa.

Os algoritmos de aprendizagem por reforço implementados foram aqueles apresentados na seção 2.2: Q-*learning*, Sarsa e Q(λ). A tabela 1 apresenta os parâmetros (obtidos empiricamente) usados nos experimentos realizados. Para os três algoritmos foram utilizados os mesmos valores de reforços, taxa de aprendizado e desconto temporal.

Partindo do ponto inicial indicado na figura 10, o robô

utiliza os algoritmos de aprendizagem por reforço para aprender uma política de ações que o conduzam ao alvo maximizando os reforços recebidos. Com o objetivo de fazer o robô explorar também a região próxima ao alvo (e não apenas o caminho até atingi-lo), cada episódio de aprendizagem era iniciado com o robô no ponto inicial e terminado 25 passos após o robô atingir uma área em que recebesse um reforço $r > 0{,}65$.

Definiu-se a escolha das ações a serem tomadas através de uma política exploratória. A probabilidade de exploração era, inicialmente, de 20%, isto é, em cada escolha de ação haveria 20% de chance de o algoritmo escolher uma ação totalmente aleatória. Para permitir a convergência dos algoritmos, a probabilidade de exploração decaía ao longo do tempo, mantendo os outros parâmetros inalterados. No episódio 10, a probabilidade de exploração recebia o valor 5%, e no episódio 16 (próximo ao término do treinamento), recebia o valor 1%.

3.4 Resultados Obtidos

Foram testados os três algoritmos de aprendizagem por reforço, aplicados aos três mapas de ambientes distintos, utilizando as três configurações diferentes de partições. Isto totaliza 27 configurações de experimentos. Cada experimento, por sua vez, consiste num conjunto de 10 repetições de treinamentos, cada um dos quais correspondendo a 20 episódios.

Cada experimento demorou, em média, 25 horas para ser concluído, sendo que cada episódio demorava de 1 a 4 horas, dependendo do estágio do aprendizado. As experiências foram realizadas simultaneamente em vinte microcomputadores, em sua maioria Pentium 400 MHz com 128 MBytes de RAM.

3.4.1 Resultados para o Mapa 2

As figuras a seguir apresentam os gráficos de aprendizagem para os algoritmos aplicados aos mapas do tipo 2 (figura 6). Os gráficos indicam quantos passos, em média, o robô realiza para concluir cada episódio no decorrer do experimento. Para cada gráfico, foi traçada a curva com a média de passos em cada episódio, e para cada episódio o desvio padrão encontrado nas dez repetições do experimento (linhas verticais). A figura 11 apresenta os gráficos do algoritmo Q-*learning*, a figura 12, do algoritmo Sarsa, e a figura 13, do algoritmo Q(λ).

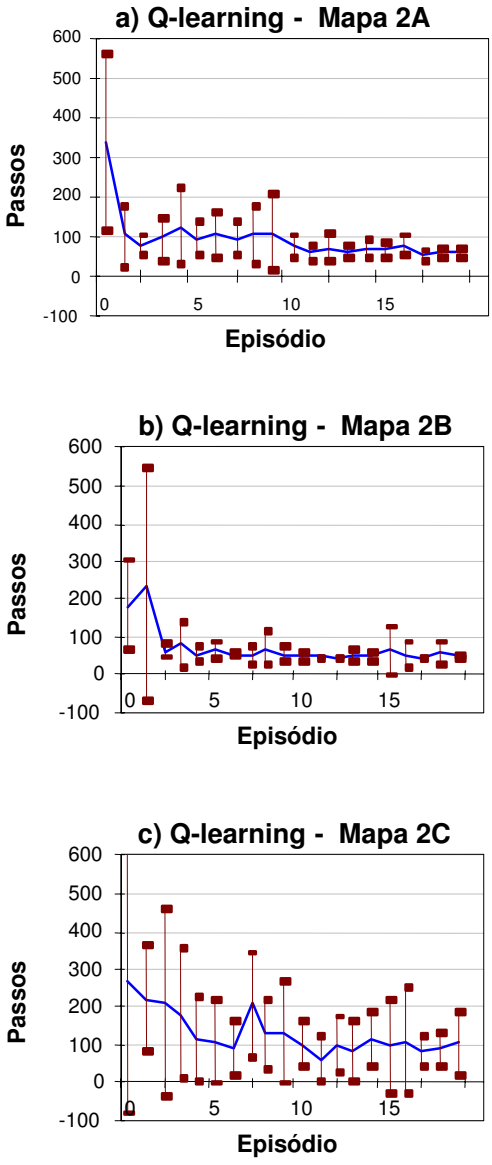
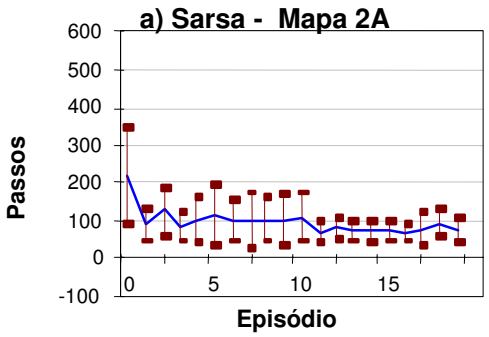
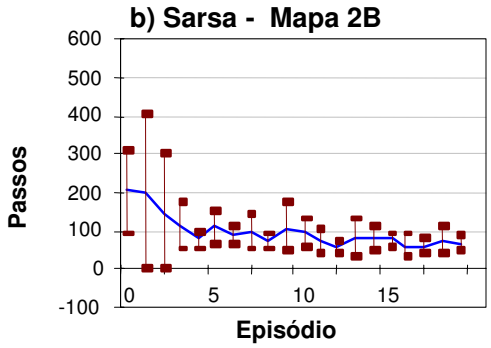


Figura 11: Curvas de aprendizagem para Q-learning, mapa 2. Particionamento: a) Original b) Perfeito c) Pior.

3.4.2 Resultados para o Mapa 3

Esta seção apresenta gráficos (figuras 14, 15 e 16) para os resultados obtidos na aplicação dos algoritmos de aprendizagem por reforço aos mapas do tipo 3 (sala com dois obstáculos nas extremidades, figura 7). Para cada gráfico, foi traçada a curva com a média de passos em cada episódio, e para cada episódio o desvio padrão encontrado nas dez repetições do experimento (linhas verticais).





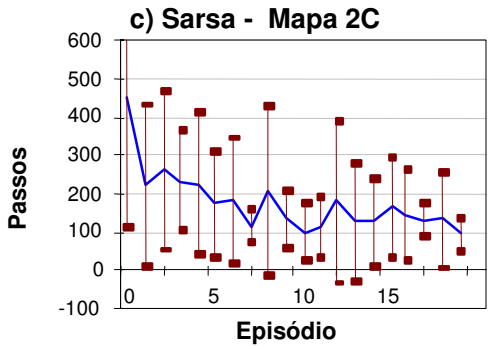
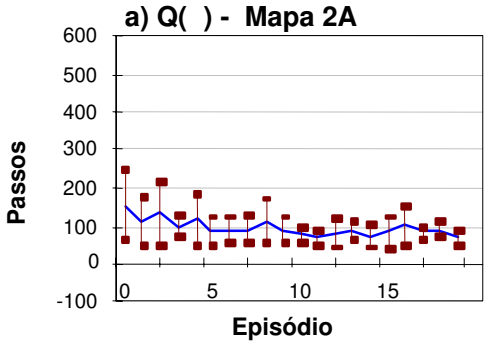
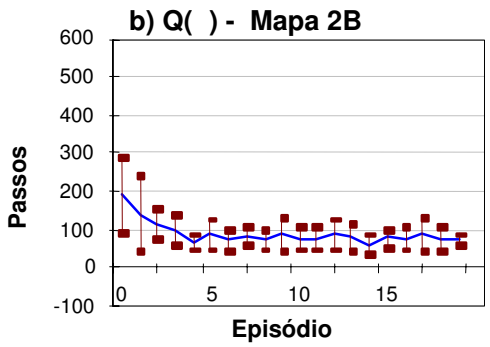


Figura 12: Curvas de aprendizagem para Sarsa, mapa 2. Particionamento: a) Original b) Perfeito c) Pior.





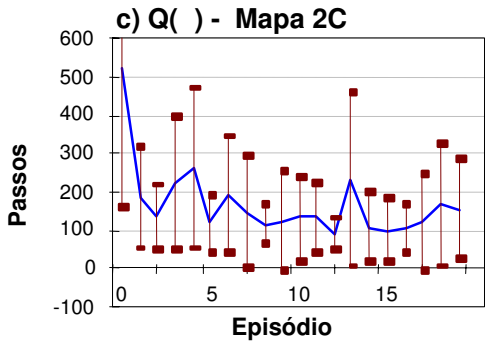


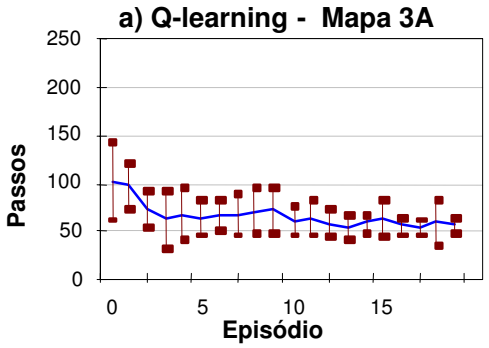
Figura 13: Curvas de aprendizagem para Q(λ), mapa 2. Particionamento: a) Original b) Perfeito c) Pior.

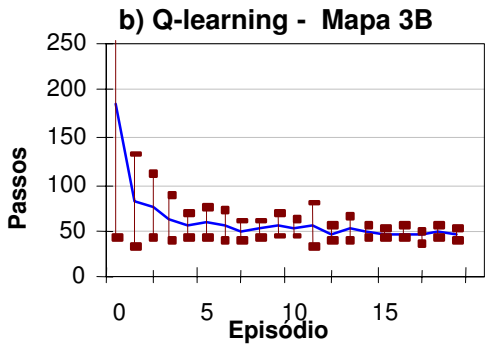
3.4.3 Resultados para o Mapa 4

Esta seção apresenta gráficos (figuras 17, 18 e 19) para os resultados obtidos na aplicação dos algoritmos de aprendizagem por reforço aos mapas do tipo 4 (sala com um obstáculo central, figura 8). Para cada gráfico, foi traçada a curva com a média de passos em cada episódio, e para cada episódio o desvio padrão encontrado nas dez repetições do experimento (linhas verticais).

3.4.4 Resumo

Para resumir as informações referentes aos resultados de todos os mapas e algoritmos, as figuras 20 e 21 apresentam uma tabela e um gráfico com as médias de valores encontrados no último episódio das experiências. A figura 20 apresenta a média de passos realizados no último episódio até a conclusão da tarefa (ou seja, o número médio de ações executadas pelo robô do ponto inicial ao ponto final, incluindo-se os 25 passos próximos ao alvo).





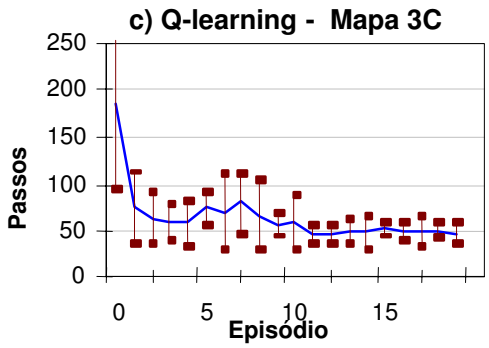
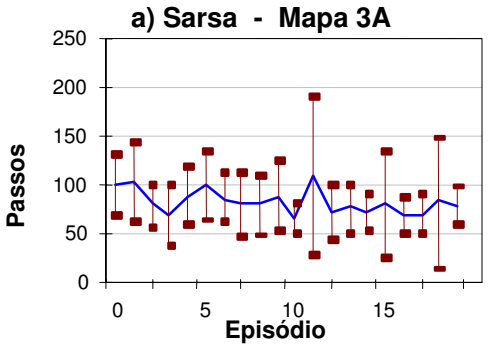
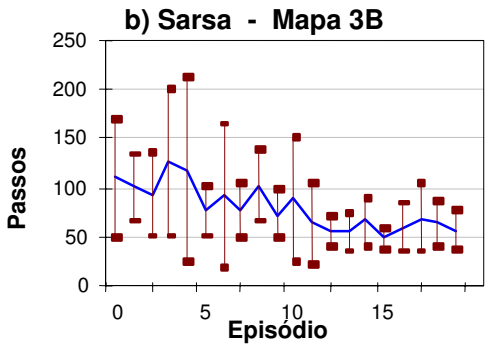


Figura 14: Curvas de aprendizagem para Q-learning, mapa 3. Particionamento: a) Original b) Perfeito c) Pior.





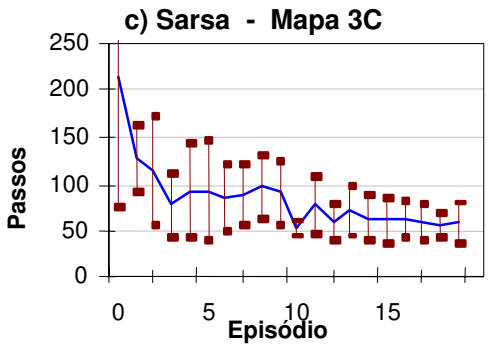


Figura 15: Curvas de aprendizagem para Sarsa, mapa 3. Particionamento: a) Original b) Perfeito c) Pior.

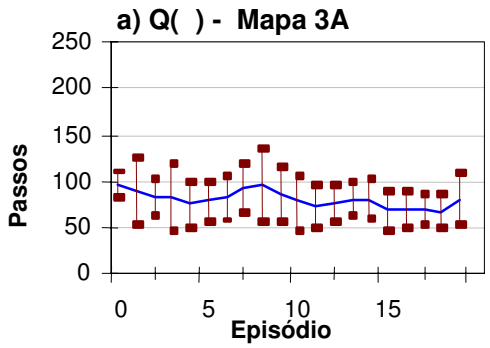
A figura 21 apresenta a média dos reforços obtidos durante a trajetória executada no último episódio.

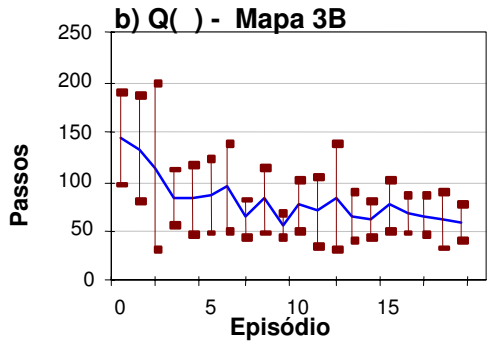
4 CONCLUSÕES

Os resultados apresentados nas seções anteriores fornecem embasamento para várias considerações tanto referentes aos mapas quanto aos algoritmos de aprendizagem.

Todos os algoritmos foram afetados pela observabilidade parcial dos estados. Ainda assim, o desempenho da aprendizagem por reforço utilizando como base os mapas cognitivos adquiridos originalmente pelo robô Magellan Pro^{TM}real foi satisfatório, de maneira geral, para todos os mapas e variações testadas.

O mapa 2, com um obstáculo lateral, permitiu a melhor comparação dos algoritmos, por possuir a maior variação na qualidade dos particionamentos. O maior erro





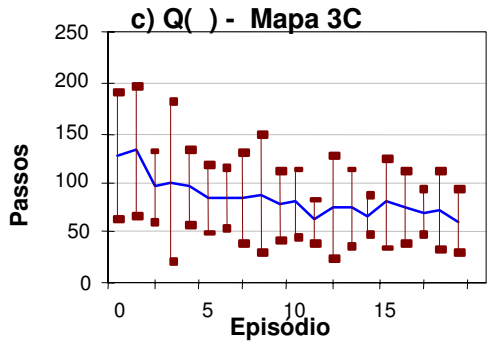
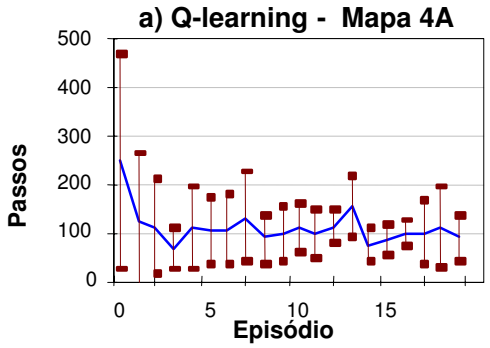
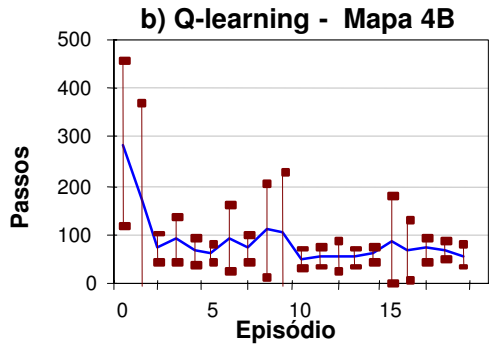


Figura 16: Curvas de aprendizagem para Q(λ), mapa 3. Particionamento: a) Original b) Perfeito c) Pior.





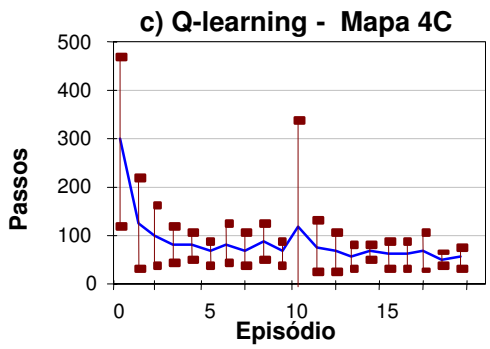
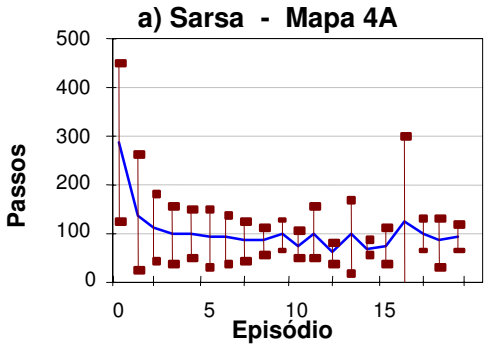


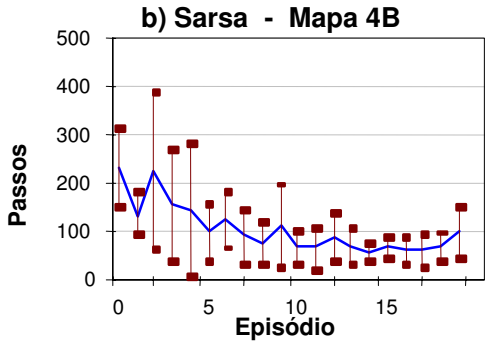
Figura 17: Curvas de aprendizagem para Q-learning, mapa 4. Particionamento: a) Original b) Perfeito c) Pior.

do mapa está localizado na região próxima ao ponto inicial do aprendizado e sua influência tende a diminuir no decorrer do aprendizado, quando a escolha de ações torna-se menos exploratória. Apesar disso, a influência do erro no aprendizado foi evidente, e pode ser observada claramente nos gráficos.

O mapa 3, com dois obstáculos nas extremidades, foi o que apresentou aprendizado mais fácil para os algoritmos, uma vez que os obstáculos estavam longe da trajetória entre o ponto inicial e o alvo.

O mapa 4, com obstáculo central, era o mapa mais complexo. Devido a isso, a complexidade das partições era maior também. Mesmo no mapa original obtido pelo robô, as partições eram mais refinadas, tornando o aprendizado não tão penoso. Também devido ao fato deste ter sido o último mapa adquirido, após o algoritmo de aprendizagem de mapas ter sido exaustivamente testado, não havia muita diferença na qualidade dos mapas,





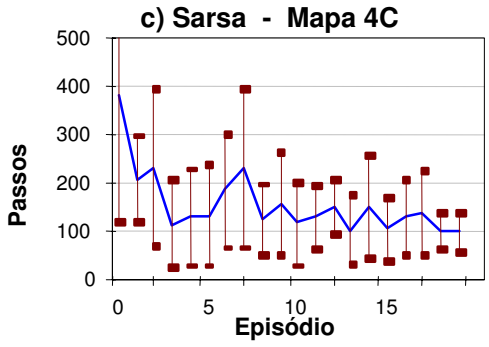
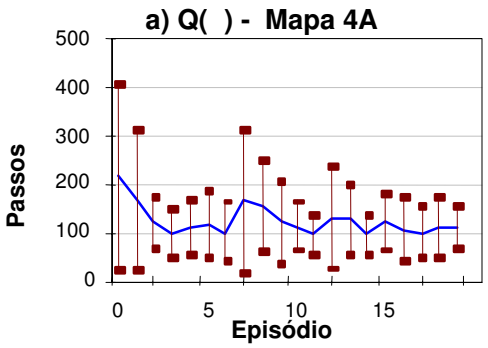
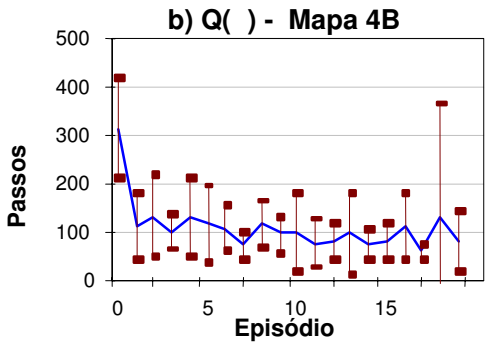


Figura 18: Curvas de aprendizagem para Sarsa, mapa 4. Particionamento: a) Original b) Perfeito c) Pior.





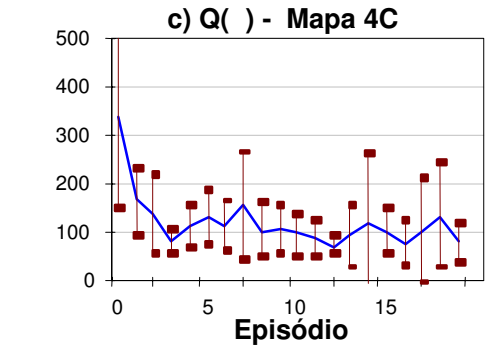
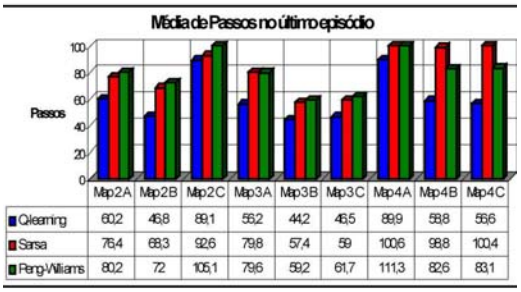


Figura 19: Curvas de aprendizagem para Q(λ), mapa 4. Particionamento: a) Original b) Perfeito c) Pior.

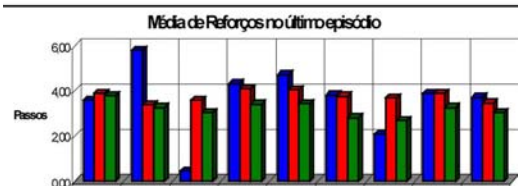
do original ao pior.

Para cada ambiente, o mapa obtido apenas com as partições originais permitiu realizar o aprendizado da tarefa no ambiente quase tão bem quanto nos casos que incluíam o subparticionamento, apesar da ambigüidade gerada pelo tamanho das partições – que tornava indistinguíveis obstáculos de vazio, na passagem de uma partição à outra. Isto possivelmente deve-se ao fato das partições maiores serem menos afetadas por pequenas imperfeições, cujos efeitos são filtrados pelo próprio tamanho das partições. O pior mapa também permitiu aprendizado e foi útil na comparação dos resultados. Como conclusão principal no que se refere aos mapas, os resultados indicam que o efeito de *perceptual aliasing*, embora prejudique a aprendizagem, não o faz de maneira abrupta em função da qualidade do mapa gerado pelo método utilizado, e tampouco inviabiliza o treinamento baseado em aprendizagem por reforço.



	Map2A	Map2B	Map2C	Map3A	Map3B	Map3C	Map4A	Map4B	Map4C
Qlearning	60,2	46,8	89,1	56,2	44,2	46,5	89,9	58,8	56,6
Sarsa	76,4	68,3	92,6	79,8	57,4	59	100,6	98,8	100,4
Peng-Williams	80,2	72	105,1	79,6	59,2	61,7	111,3	82,6	83,1

Figura 20: Gráfico de barras e tabela para o número médio de passos no último episódio. Valores menores indicam melhor aprendizado.



	Map2A	Map2B	Map2C	Map3A	Map3B	Map3C	Map4A	Map4B	Map4C
Qlearning	3,59	5,79	0,45	4,31	4,71	3,80	2,07	3,87	3,70
Sarsa	3,87	3,39	3,57	4,08	4,04	3,76	3,68	3,89	3,45
Peng-Williams	3,79	3,24	3,03	3,40	3,43	2,81	2,68	3,24	3,03

Figura 21: Gráfico de barras e tabela para a média dos reforços obtidos durante a realização do último episódio. Valores maiores indicam melhor aprendizado.

O algoritmo Q-*learning* foi o que aprendeu a melhor política de ações em todos os casos, ou seja, realizou menos passos para atingir o alvo, forneceu reforços maiores, e a trajetória aprendida aproximou-se mais da ótima.

O algoritmo Sarsa foi o menos afetado pelas variações de qualidade do mapa, obtendo média de reforços no último episódio sempre positivos. Este comportamento foi decorrente da trajetória mais conservadora obtida pelo algoritmo, ou seja, como o robô aprendeu um caminho mais distante dos obstáculos, recebe menos reforços negativos. O Sarsa é um algoritmo muito sensível à política e provavelmente a obtenção de uma trajetória conservadora decorreu da política pouco exploratória.

O algoritmo Q(λ) foi o algoritmo mais afetado pela variação na qualidade do mapa. Apresentou também as piores trajetórias, mesmo no final do aprendizado. Vários fatores podem justificar este mau comportamento: parâmetros não ajustados corretamente, tamanho e característica de partições inadequados, e tipo de tarefa e de estratégia de aplicação de reforços não favorecendo o uso de traços de elegibilidade.

Algoritmos com traços de elegibilidade são indicados para problemas modelados como processos de decisão Markovianos parcialmente observáveis. No entanto, a estrutura do problema testado nos experimentos prejudicou o desempenho do Q(λ). Primeiramente, os reforços positivos eram distribuídos ao longo da trajetória, baseado em uma função proporcional à distância. Desta forma, o robô não precisava atingir o alvo para receber reforços positivos. Isto acelerou o aprendizado para todos os algoritmos, mas diminuiu a vantagem relativa dos traços de elegibilidade de distribuir o crédito do reforço para os pares estado-ação envolvidos em trajetórias bem sucedidas. Em segundo lugar, e mais importante, o tamanho das partições e a definição das ações possíveis tornou freqüente o problema de o robô revisitar um par estado-ação antes que o traço devido à primeira visita tivesse decaído totalmente a zero, provocando um incremento no traço que freqüentemente tornava-se maior que um. Isto provocou problemas em casos em que, estando em um estado, uma ação errada é escolhida pelo robô algumas vezes antes de uma certa (assumindo como ação certa a proveniente de uma política ótima ideal). Assim, o traço referente à ação errada estaria provavelmente maior que o referente à ação certa. Apesar da ação correta ter sido tomada mais recentemente, a ação errada foi selecionada mais vezes. Quando o reforço é recebido, o valor para a ação errada provavelmente será maior que o valor para ação certa, fazendo com que posteriormente a escolha da ação errada seja tomada muitas vezes antes da ação certa, e tornando mais provável que a ação errada tenha traço maior novamente. Desta forma, o robô entra em um círculo vicioso que eventualmente é quebrado, mas atrasa o aprendizado. Isto poderia ser resolvido utilizando alguma estratégia como a substituição de traços (*replacing traces*), conforme sugerido em Singh, S. P. e Sutton, R. S. (1996).

AGRADECIMENTOS

Os autores agradecem o apoio da FAPESP (proc. no. 00/06147-3) e do CNPq (proc. no. 301228/97-3 NV).

REFERÊNCIAS

Arleo, A., Millán, J. R. e Floreano, D. (1999). Efficient learning of variable-resolution cognitive maps for autonomous indoor navigation. *IEEE Transactions of Robotics and Automation.* Vol. 15, No. 6, pp. 990-1000.

Bellman, R. (1957). *Applied dynamic programming.* Princeton University Press.

Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: The perceptual distinctions ap-

proach. *Procs. of the 10^{th} National Conf. on artificial Intelligence*, pp. 183-188.

Elfes, A. (1987). Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3): 249-265, Junho 1987.

Haykin, S. (1999). *Neural networks: a comprehensive foundation*. 2^a ed. New Jersey: Prenctice-Hall Inc.

IS Robotics, Inc. (2000). *Magellan Pro compact mobile robot user's guide*. USA: Real World Interface Division.

Konolige, K. e Myers, K. L. (1996). The Saphira architecture for autonomous mobile robots. *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press.

Lovejoy, W. S. (1991). A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research,* No. 28:47-66.

Peng, J. e Williams, R. J. (1996). Incremental multi-step Q-learning. *W. W. Cohen e H. Hirsh (eds.), Proceedings of the Eleventh International Conference on Machine Learning*, pp. 226-232. San Francisco: Morgan Kaufmann.

Ribeiro, C. H. C. (1999). A Tutorial on reinforcement learning techniques. *Supervised Learning track tutorials of the 1999 International Joint Conference on Neuronal Networks*. Washington: INNS Press.

Singh, S. P. (1994). Learning to solve Markovian decision processes. Tese de doutoramento, University of Massachusetts.

Singh, S. P. e Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning,* No. 22:123-158.

Sutton, R. S. e Barto, A. G. (1998). *Reinforcement learning: An introduction*. Massachusetts: MIT Press.

Thrun, S. (1998). Learning maps for indoor mobile robot navigation. *Artificial Intelligence*, Vol. 99, p. 21-71.

Watkins, C. J. C. H. e Dayan, P. (1992). Q-learning. *Machine Learning,* n. 8 (3/4):279-292.