APRENDIZAGEM DA NAVEGAÇÃO EM ROBÔS MÓVEIS A PARTIR DE MAPAS OBTIDOS AUTONOMAMENTE

Sildomar T. Monteiro e Carlos H. C. Ribeiro
Divisão de Ciência da Computação
Instituto Tecnológico de Aeronáutica
Praça Mal. Eduardo Gomes, 50
CEP 12228-900 São José dos Campos – SP
carlos@comp.ita.br

Resumo: Analisamos o desempenho de algoritmos de aprendizagem por reforço em um problema de navegação para o qual os mapas foram obtidos autonomamente por um robô móvel Magellan Pro™. Foram estudados os algoritmos Q-learning, Sarsa e Q(λ), e um método para criar mapas cognitivos de resolução variável foi implementado de modo a definir um verificador de desempenho para os algoritmos testados. Os resultados mostram um desempenho satisfatório dos algoritmos, com uma degradação suave em função da ambiguidade sensorial presente. 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 eficiente, permitindo uma análise adequada dos algoritmos.

Abstract: We analyzed the performance of reinforcement learning algorithms in a navigation problem where maps were generated autonomously by a mobile Magellan Pro™ robot. The algorithms assessed in this study were Q-learning, Sarsa and Q(λ), and a method to build variable resolution cognitive maps of the environment was implemented in order to create a performance verifier of the learning algorithms. The learning algorithms presented satisfactory performance, although with a graceful degradation of efficiency due to state ambiguity. The Q-learning algorithm accomplished the best performance over the experiments, followed by the Sarsa algorithm. The Q(λ) algorithm had its performance restrained by experiments parameters. The cognitive map learning method revealed to be efficient and allowed adequate algorithms assessment.

APRENDIZAGEM DA NAVEGAÇÃO EM ROBÔS MÓVEIS A PARTIR DE MAPAS OBTIDOS AUTONOMAMENTE

Resumo: Analisamos o desempenho de algoritmos de aprendizagem por reforço em um problema de navegação para o qual os mapas foram obtidos autonomamente por um robô móvel Magellan Pro™. Foram estudados os algoritmos Q-learning, Sarsa e Q(λ), e um método para criar mapas cognitivos de resolução variável foi implementado de modo a definir um verificador de desempenho para os algoritmos testados. Os resultados mostram um desempenho satisfatório dos algoritmos, com uma degradação suave em função da ambiguidade sensorial presente. 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 eficiente, permitindo uma análise adequada dos algoritmos.

Abstract: We analyzed the performance of reinforcement learning algorithms in a navigation problem where maps were generated autonomously by a mobile Magellan Pro™ robot. The algorithms assessed in this study were Q-learning, Sarsa and Q(λ), and a method to build variable resolution cognitive maps of the environment was implemented in order to create a performance verifier of the learning algorithms. The learning algorithms presented satisfactory performance, although with a graceful degradation of efficiency due to state ambiguity. The Q-learning algorithm accomplished the best performance over the experiments, followed by the Sarsa algorithm. The Q(λ) algorithm had its performance restrained by experiments parameters. The cognitive map learning method revealed to be efficient and allowed adequate algorithms assessment.

1. Introdução

Aprendizagem por reforço (AR) trata de situações onde um agente aprende uma estratégia de controle por tentativa e erro, ao atuar diretamente sobre um sistema dinâmico. Assim, 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. Aplicações reais envolvem sistemas observados por sensores com limitações inerentes à sua construção, apresentando restrições de confiabilidade, ruído e não-uniformidade nas medidas. Isto faz com que o verdadeiro estado do processo a ser controlado seja apenas indicado de forma grosseira. A ambiguidade decorrente da descrição incompleta dos estados anula a suposição de Markov, uma das bases teóricas mais importantes para a operação confiável de algoritmos de AR.

Este artigo apresenta uma análise de algoritmos de AR em um problema de navegação sob condições que envolvem mapas para navegação obtidos autonomamente por um robô real. Algoritmos estudados na literatura – Q-*learning*, Sarsa e Q(λ) – foram implementados em um robô Magellan Pro™. A tarefa de aprendizagem era a seguinte: 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

para adquirir modelos do ambiente real, que serviram como base para simulações dos algoritmos de AR.

2. Fundamentação Teórica

AR é um paradigma computacional de aprendizagem em que um agente procura maximizar uma medida de desempenho baseada nos reforços (recompensas ou punições) que recebe ao interagir com um ambiente desconhecido [Sutton, R. e Barto, A.]. 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 (sequência de ações) que maximize a soma esperada destes reforços, descontando-os (usualmente de modo exponencial) proporcionalmente ao seu atraso temporal.

2.1. Processos Decisórios de Markov

Um ambiente satisfaz a propriedade de Markov se o seu estado resume o passado de forma compacta. Pode-se predizer qual será o próximo estado e recompensa esperados, dados o estado e ação atuais. Um processo de AR que satisfaz a propriedade de Markov é chamado Processo Decisório de Markov (MDP - *Markov Decision Process*).

Formalmente, um MDP 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 uma função de recompensa R, que especifica o reforço recebido pelo agente ao 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 podem ser utilizadas para determinar uma política ótima de ações. AR é usada quando o modelo não está disponível ou quando existe apenas um modelo de simulação que não permite a formulação analítica necessária para o uso de técnicas de Controle Ótimo.

2.2. Aprendizagem por Reforço

Em um MDP, 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:

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

onde a taxa de desconto $0 < \gamma \leq 1$ determina o peso temporal relativo dos reforços.

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:

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

e a política ótima de ações $V^*(s) = \max_{\pi} V^{\pi}(s)$ é aquela que maximiza o valor de estado.

Existe sempre ao menos uma política ótima π^*, 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,

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} \qquad (3)$$

e

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \qquad (4)$$

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

Q-*learning* [Watkins, C. e Dayan, P.] é um algoritmo que permite estabelecer autonomamente uma política de ações. A idéia básica é o aprendizado da função de valor de ação ótima sobre o espaço de pares estado-ação, segundo a equação a seguir:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)] \qquad (5)$$

onde α é a taxa de aprendizagem, *r* é a recompensa, ou custo, resultante de tomar a ação *a* no estado *s* e γ é o fator de desconto.

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 gera estimativas Q_t que convergem para o valor de Q^*. Na prática, a política de ações converge para a política ótima de forma lenta.

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

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

A forma procedimental é similar à do algoritmo Q-*learning*. Sarsa converge para uma política e valor de função de ação ótimos se os pares estado-ação forem visitados um número infinito de vezes e a política de escolha da ação convirja, no limite, para uma política que utilize a melhor ação (aquela que maximiza a recompensa futura esperada).

O algoritmo Q(λ), proposto em [Peng, J. e Williams, R.] 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. Algoritmos de AR usando traços de elegibilidade são computacionalmente mais intensivos, mas em contrapartida oferecem aprendizado rápido, particularmente quando os reforços são atrasados por muitos passos. O algoritmo Q(λ) é o seguinte:

Para cada s,a faça Q(s,a)=0 e Tr(s,a)=0
Observe s
Repita

- *Selecione a ação a que maximiza Q(s,a)*
- *Execute a ação a*
- *Receba a recompensa imediata r*
- *Observe o novo estado s'*
- $e' \leftarrow r + \gamma \max_{a'}(s', a') - Q(s, a)$
- $e \leftarrow r + \gamma \max_{a'}(s', a') - \max_{a} Q(s, a)$
- *Para cada par estado-ação faça:*
  $Tr(s, a) \leftarrow \gamma\lambda Tr(s, a)$
  $Q(s, a) \leftarrow Q(s, a) + \alpha Tr(s, a)e$
- $Q(s, a) \leftarrow Q(s, a) + \alpha e'$
- $Tr(s, a) \leftarrow Tr(s, a) + 1$
- $s \leftarrow s'$

Até que critério de parada seja satisfeito

O fator *Tr(s,a)* é o traço de elegibilidade do par *(s,a)*. Esta descrição não especifica como ações são escolhidas durante os episódios. É comum a escolha aleatória de ações, com uma tendência em direção à ação aparentemente ótima, por exemplo através do uso de uma distribuição de Boltzmann, com a probabilidade de escolher uma ação *a* no estado *s* proporcional a $\exp\left(Q(s,a)/T\right)$, onde *T* é um parâmetro de temperatura.

Para este algoritmo não há prova de convergência para Q^*. A estratégia de gradualmente reduzir λ ou a temperatura ao longo do aprendizado permite, freqüentemente, convergência para políticas de boa qualidade [Peng, J. e Williams, R.].

O termo *Dyna* [Singh, S. e Sutton, R.] define uma técnica para integrar aprendizado, planejamento e atuação. O agente interage com o ambiente e gera experiências utilizadas tanto para aperfeiçoar um modelo do ambiente quanto para melhorar as funções de valor e política de ações (planejamento sobre o modelo).

Após cada transição $s_t, a_t \rightarrow s_{t+1}, r_t$, *Dyna* armazena, para cada (s_t, a_t), o par (s_{t+1}, r_t) observado. Durante o planejamento, são escolhidas amostras aleatórias de pares estado-ação já experimentados e contidos no modelo. Realizam-se experiências simuladas nestes pares e são aplicadas atualizações baseadas em AR sobre essas experiências, como se estivessem realmente ocorrendo. Tipicamente, o mesmo método de AR é usado para o aprendizado a partir da experiência real e para as experiências simuladas.

Nos experimentos descritos neste artigo, *Dyna* foi utilizado com a intenção de aumentar a velocidade de aprendizado dos algoritmos de aprendizagem por reforço.

2.3. Ambigüidade Sensorial

Um problema comum em aplicações de robótica é que, freqüentemente, o estado não é totalmente observável. Nas situações práticas, os sensores do agente fornecem apenas informações parciais, que não percebem o estado real do ambiente. Este fato implica a quebra da condição de Markov, resultando em *perceptual aliasing* [Chrisman, L.], situação na qual dois ou mais estados são perceptualmente idênticos, mas necessitam de respostas (ações) distintas. Um agente que aprende comportamentos em função das percepções imediatas dos sensores estará sujeito ao *perceptual aliasing*.

Problemas envolvendo *perceptual aliasing* são conhecidos como Processos Decisórios de Markov Parcialmente Observáveis. Pesquisas vêm sendo realizadas para resolver esta classe de problemas e existem alguns métodos sub-ótimos propostos, geralmente usando memória ou atenção [Chrisman, L.].

No contexto de aprendizagem por reforço, o processo de aquisição de valores de utilidade através do recebimento de reforços 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 poderá ser propagado para estados correspondentes a visitações anteriores. Este processo de propagação de erro corresponde a um efeito típico de *perceptual aliasing*.

2.4. Construção de Mapas

Foi utilizada uma técnica [Arleo, A., Millán, J. e Floreano, D.] que permite a aprendizagem rápida de mapas com parâmetros ajustáveis, para comportar os testes dos algoritmos de AR. O mapa gerado consiste de um conjunto P de partições $p \in P$ de diferentes tamanhos, representando regiões sensorialmente homogêneas correspondendo a modelos locais que representam o conhecimento do robô sobre aquelas. O mapa resultante é portanto uma representação simplificada da estrutura geométrica do mundo. A localização é calculada por estimação de posição via odometria e sonares.

O processo de geração do mapa é descrito por atividades de exploração e atualização. O robô explora o ambiente e só interrompe a exploração para incorporar um obstáculo desconhecido ao modelo, atualizando a resolução das partições:

Início
Repetir
1. *Exploração: Seleção do próximo alvo.*
2. *Planejamento: Cálculo de uma trajetória para alvo através das partições atuais.*
3. *Ação: Realiza movimento do robô através dos atuadores.*
4. *Se robô atinge o alvo, explora a região exaustivamente.*
5. *Atualização: Se incoerência é detectada, pára exploração e modela obstáculo:*
    - *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.*
        - *5.2.3. Move-se até o final do limite.*
        - *5.2.4. Memoriza o canto do obstáculo.*
    - *5.3. Aumenta resolução da partição.*
    - *5.4. Atualiza o banco de dados de experiências.*
    - *5.5. Remove partições redundantes do conjunto de partições.*
    - *5.6. Abstrai novo grafo topológico.*

Até que mapa esteja totalmente construído.
Fim.

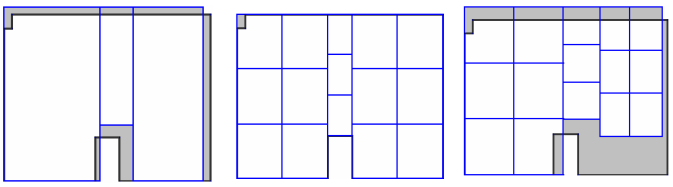
O algoritmo é eficiente computacionalmente, mas existem limitações. Há uma suposição de obstáculos paralelos ou perpendiculares entre si, e o ambiente a ser modelado pressupõe-se estático.

Para os experimentos, foram inicialmente obtidos mapas (de acordo com o algoritmo acima) para uma sala de 3m X 3,5m preparada com obstáculos. Utilizou-se como base o robô Magellan Pro™ [IS Robotics, Inc.], um robô móvel cilíndrico para ambientes fechados, dotado de 16 sonares distribuídos ao longo de seu perímetro.

Decidiu-se, ainda, dividir as partições obtidas pelo algoritmo de geração de mapas em partições menores (subparticionamento), permitindo uma observação mais detalhada do ambiente e um melhor aprendizado de política de ações. O critério adotado foi subdividir as partições originais observando a seguinte regra: uma subpartição não poderia ter tamanho, em cada eixo, menor que três vezes o raio do robô, já que o comando de movimento avança o robô a uma distância de um raio a cada ação "ir adiante" (ver seção 3 para a definição das ações).

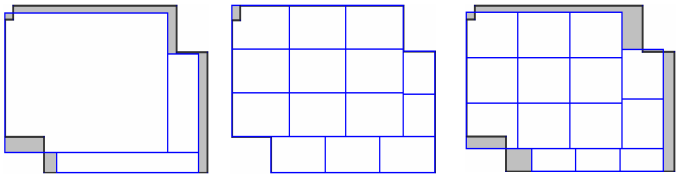
Detalhes sobre a obtenção dos mapas estão em [Monteiro, S. e Ribeiro, C.].

Foram definidas três versões de particionamento para cada mapa: a) melhor particionamento original obtido no mundo real, sem subparticionamento; b) particionamento perfeito com base nas partições obtidas pelo robô e subparticionamento; e c) pior mapa obtido utilizando o subparticionamento. Assim, obteve-se os mapas 2, 3 e 4 e respectivas partições, mostrados nas figuras 1, 2 e 3 (o mapa 1, sem obstáculos, foi usado em testes). 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.



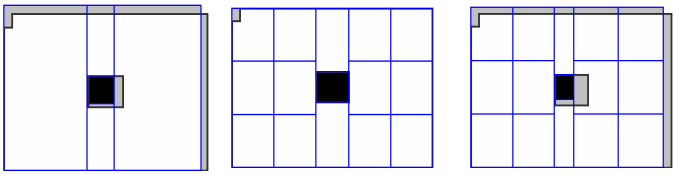
a) 2A **b) 2B** **c) 2C**

Figura 1 - Mapa 2. Versões: a) 2A (Original), b) 2B (Perfeito), c) 2C (Pior).



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

Figura 2 - Mapa 3. Versões: a) 3A (Original), b) 3B (Perfeito), c) 3C (Pior).



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

Figura 3 - Mapa 4. Versões: a) 4A (Original), b) 4B (Perfeito), c) 4C (Pior).

3. Resultados

A partir dos mapas adquiridos, os algoritmos de AR foram implementados e testados utilizando-se o simulador Saphira [Konolige, K. e Myers, K.]. A tarefa a ser aprendida pelo robô foi aproximar-se de um alvo, desviando de obstáculos e paredes. O Saphira simula erros de leitura de sonar e odometria, replicando o problema de estimação de localização que teríamos no robô real.

Os estados do robô foram definidos como a combinação de duas informações: a sua localização (partição em que se encontra), e a sua direção (para qual lado está virado: norte, sul, leste, oeste). Foram usadas as partições obtidas com o robô no ambiente real.

As ações possíveis para o robô são: "ir adiante", "virar à direita" e "virar à esquerda". A ação "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° para a direita, e a ação "virar à esquerda", gira o robô 90° para a esquerda. Estas ações também são aquelas definidas para o robô real, quando da construção do mapa.

Os parâmetros (obtidos empiricamente) para os algoritmos de AR testados foram:

- r=-1 quando robô se aproxima de obstáculo; $0 \leq r \leq 1$, segundo função linear com valor inversamente proporcional à distância do alvo, em outros casos.
- $\gamma = 0{,}95$, $\alpha = 0{,}9$, $\lambda = 0{,}5$ (traço de elegibilidade para $Q(\lambda)$).

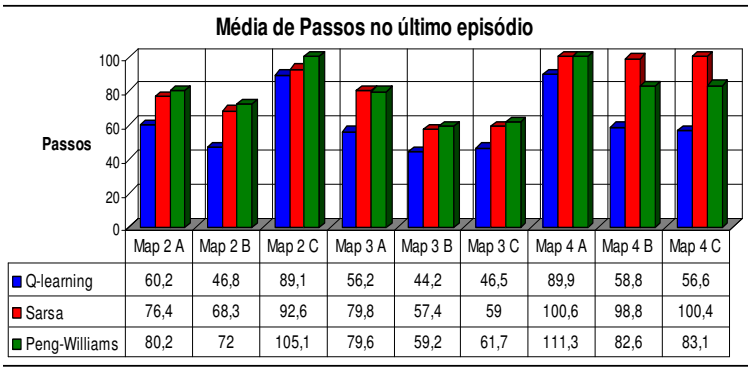
- Experiências atualizadas (*Dyna*): 30 experiências por passo.

Para fazer o robô explorar a região próxima ao alvo (e não apenas o caminho até atingi-lo), cada episódio 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 do algoritmo escolher uma ação aleatória. Esta probabilidade decaiu ao longo do tempo, mantendo os outros parâmetros inalterados. No episódio 10, a probabilidade era de 5%, e no episódio 16 (próximo ao fim do treinamento), de 1%.

Foram testados os três algoritmos de AR, aplicados aos três mapas de ambientes distintos, utilizando as três configurações de partições (27 configurações de experimentos). Cada experimento consistiu num conjunto de 10 repetições de treinamentos, cada um dos quais com 20 episódios. Cada experimento demorou, em média, 25 horas para ser concluído, As experiências foram realizadas simultaneamente em 20 microcomputadores Pentium 400 MHz.

As figuras 5 e 6 apresentam tabelas e gráficos de barras com médias de valores no último episódio das experiências. A figura 5 apresenta a média de passos no último episódio até a conclusão da tarefa (número médio de ações executadas pelo robô do ponto inicial ao final, incluindo-se os 25 passos próximos ao alvo). A figura 6 apresenta a média dos reforços obtidos na trajetória executada no último episódio.



	Map 2 A	Map 2 B	Map 2 C	Map 3 A	Map 3 B	Map 3 C	Map 4 A	Map 4 B	Map 4 C
Q-learning	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 5 - Gráfico e tabela, passos no último episódio.

	Map 2 A	Map 2 B	Map 2 C	Map 3 A	Map 3 B	Map 3 C	Map 4 A	Map 4 B	Map 4 C
Q-learning	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 6 - Gráfico e tabela, reforços durante a realização do último episódio.

4. Conclusões

Todos os algoritmos foram afetados pela observabilidade parcial dos estados. Ainda assim, o desempenho da aprendizagem utilizando como base os mapas adquiridos pelo robô real foi satisfatório, para todos os mapas e variações testadas.

O mapa 2 permitiu a melhor comparação dos algoritmos, por possuir a maior variação na qualidade dos particionamentos. O maior erro está localizado na região próxima ao ponto inicial, e sua influência tende a diminuir no decorrer do aprendizado, quando a escolha de ações torna-se menos exploratória. O mapa 3 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 era o 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 menos 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, melhor e pior entre diferentes tentativas.

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, filtradas pelo próprio tamanho das partições. 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, e não inviabiliza o treinamento baseado em AR.

O algoritmo Q-*learning* foi o que aprendeu a melhor política de ações em todos os casos, ou seja, utilizou menos passos para atingir o alvo e forneceu reforços maiores.

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 de política pouco exploratória.

O algoritmo Q(λ) foi o mais afetado pela variação na qualidade do mapa, e apresentou as piores trajetórias aprendidas. A estrutura do problema testado nos experimentos certamente foi decisiva neste aspecto. Primeiramente, os reforços positivos eram distribuídos ao longo da trajetória, baseado em uma função proporcional à distância. Assim, 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 estados envolvidos em trajetórias bem sucedidas. Em segundo lugar, o tamanho das partições e a definição das ações 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 ação certa (aquela proveniente de uma política ótima). Assim, o traço referente à ação errada poderia tornar-se maior que o referente à ação certa. Mesmo que a ação certa tenha sido escolhida recentemente, a ação errada foi selecionada mais vezes. Ao ser recebido o reforço, o valor para a ação errada provavelmente será maior que o valor para ação certa, fazendo com que posteriormente a escolha da ação errada ocorra antes da escolha da ação certa, e tornando mais provável que a ação errada tenha traço maior novamente. O robô entra assim em um círculo vicioso que eventualmente é quebrado, mas que atrasa o aprendizado. Isto poderia ser resolvido utilizando uma estratégia de substituição de traços, como sugerido em [Singh, S. e Sutton, R.].

5. Agradecimentos

Os autores agradecem o apoio da FAPESP (00/06147-3) e do CNPq (301228/97-3 NV).

6. Bibliografia

Arleo, A., Millán, J. e Floreano, D. Efficient Learning of Variable-Resolution Cognitive Maps for Autonomous Indoor Navigation. *IEEE Trans. on Robotics and Automation.* 15(6), pp. 990-1000, 1999.

Chrisman, L. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. *Procs. of the 10th Nat. Conf. on Artificial Intelligence*, pp. 183-188, 1992.

IS Robotics, Inc.. *Magellan Pro Compact Mobile Robot User's Guide*. USA: Real World Interface Division, 2000.

Konolige, K. e Myers, K. The Saphira Architecture for Autonomous Mobile Robots. *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, 1996.

Monteiro, S. e Ribeiro, C. Obtenção de Mapas Cognitivos para o Robô Magellan Pro. Anais do XIV Cong. Brasileiro de Automática, CD-ROM, pp. 1543-1548, 2002.

Peng, J. e Williams, R. Incremental multi-step Q-learning. *W. W. Cohen e H. Hirsh (eds.), Proc.s of the 11th Int. Conf. on Machine Learning*, pp. 226-232, 1996.

Singh, S. e Sutton, R. Reinforcement learning with replacing eligibility traces. *Machine Learning,* 22, pp. 123-158, 1996.

Sutton, R. e Barto, A. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

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