COMPARAÇÃO DE DESEMPENHO DE ALGORITMOS DE APRENDIZADO POR REFORÇO NO DOMÍNIO DO FUTEBOL DE ROBÔS

MURILO FERNANDES MARTINS, REINALDO A. C. BIANCHI

Departamento de Engenharia Elétrica, Centro Universitário da FEI
Av. Humberto de Alencar Castelo Branco, 3972, CEP 09850-901 São Bernardo do Campo, SP – Brasil
murilofm@yahoo.com.br, rbianchi@fei.edu.br

Abstract— This paper presents a comparison between three Reinforcement Learning algorithms – Q-learning, Q(λ)-learning and QS-learning – and its variants – HAQL, HAQ(λ) and HAQS – capable of using heuristics to influence action selection and speed up the learning. The experiments were done in a simulated Robot-soccer environment – SimuroSot – that reproduces the conditions of a real MiroSot competition league environment. The results obtained show that the use of heuristics enhances significantly the performance of the learning algorithms.

Keywords— Reinforcement Learning, Robot-soccer, Autonomous Mobile Robotics

Resumo— Este artigo apresenta uma comparação entre três algoritmos de Aprendizado por Reforço – Q-learning, Q(λ)-learning e QS-learning – e suas variantes – HAQL, HAQ(λ) e HAQS – capazes de utilizar heurísticas para influenciar a escolha das ações e acelerar o aprendizado. Os experimentos foram feitos em um ambiente de Futebol de Robôs simulado – SimuroSot – que reproduz as condições de um ambiente real de competição da categoria MiroSot. Os resultados obtidos mostram que o uso de heurísticas melhora significativamente o desempenho dos algoritmos de aprendizado.

Palavras-chave— Aprendizado por Reforço, Futebol de Robôs, Robótica Móvel Autônoma

1 Introdução

Ao longo dos anos, diversos algoritmos de Aprendizado por Reforço (AR) foram propostos (Sutton e Barto, 1998). No entanto, os algoritmos mais utilizados são aqueles que não necessitam de um modelo de ambiente para aprender. Dentre tais algoritmos livres de modelo, destaca-se o algoritmo Q-learning (Watkins, 1989). Embora o Q-learning seja de fácil implementação e tenha sua convergência provada matematicamente por Watkins (1989), tal convergência é muito lenta, pois apenas um par estado-ação tem seu valor atualizado a cada iteração do algoritmo.

Visando aprimorar o desempenho do Q-learning, diversas abordagens foram propostas para acelerar o aprendizado. Este artigo apresenta uma análise comparativa entre o algoritmo Q-learning, duas abordagens propostas para melhorar seu desempenho – Q(λ)-learning (Watkins, 1989; Peng e Williams, 1996), que utiliza generalizações temporais e QS-learning (Ribeiro e Szepesvári, 1996), que utiliza generalizações espaciais – e variantes desses algoritmos capazes de acelerar o aprendizado através do uso de heurísticas definidas *a priori* para influenciar a escolha das ações a serem executadas pelo agente – HAQL (Bianchi *et al.*, 2004), HAQ(λ) e HAQS.

Outra contribuição desse trabalho é estender o problema inicialmente abordado por Littman (1994), que propõe um ambiente de Futebol de Robôs simulado simplificado, a um ambiente simulado mais complexo, dinâmico e não-determinístico – a categoria *SimuroSot* da FIRA (FIRA, 2007) – que reproduz um ambiente real de competição da categoria *MiroSot* (FIRA, 2007).

O artigo segue com a Seção 2 descrevendo os algoritmos Q-learning, Q(λ) e QS, assim como algumas peculiaridades de implementação. A Seção 3 apresenta o método de aceleração do AR utilizando heurísticas definidas *a priori*, assim como os algoritmos HAQL, HAQ(λ) e HAQS. O ambiente *SimuroSot* (FIRA, 2007) utilizado para os testes, assim como as definições dos parâmetros dos algoritmos, são descritos em detalhes, juntamente com os resultados obtidos, na Seção 4. A Seção 5 conclui o artigo e apresenta os trabalhos futuros.

2 Os Algoritmos de Aprendizado por Reforço

O Processo Markoviano de Decisão (PMD) (Sutton e Barto, 1998) é a formulação matemática mais utilizada para se modelar um agente de AR. Um PMD define que as transições de estados são independentes de qualquer estado anterior do ambiente ou qualquer ação anteriormente selecionada pelo agente. É composto por um espaço de estados S, um conjunto de ações A, uma função de recompensa $R: S \times A \rightarrow \Re$ que especifica a tarefa do agente e uma função de transição de estados $T: S \times A \rightarrow \Pi(S)$, onde $\Pi(S)$ representa uma distribuição de probabilidades sob o espaço de estados. Para solucionar um PMD é necessário computar a política $\pi: S \times A$ que maximiza a recompensa recebida pelo agente ao longo do tempo.

2.1 O algoritmo Q-learning

O Q-learning é o algoritmo de AR mais popular, minuciosamente estudado e capaz de tornar simples a tarefa de um agente aprender uma política ótima quando modelado em um PMD. É definida uma fun-

ção $Q(s,a)$ que representa a máxima recompensa acumulada ao longo do tempo, que pode ser obtida a partir do estado atual s, quando a ação a é selecionada. A tabela 1 apresenta o algoritmo Q-learning implementado nesse trabalho.

Tabela 1. O algoritmo Q-learning

Para todo estado s e toda ação a, inicialize $Q(s,a)$ e $V(s)$ com zero;
Observe o estado atual s;
Repita infinitamente:
(1) Selecione uma ação a utilizando a regra ε – *Greedy*;
(2) Receba a recompensa imediata r;
(3) Observe o novo estado s';
(4) Compute o erro de diferenças temporais TD(0) e':
$e' = [r(s,a) + \gamma \cdot V(s') - Q(s,a)]$
(5) Atualize $Q(s,a) = Q(s,a) + \alpha_n . e'$
(6) $s \leftarrow s'$

O estado atual é representado por s, enquanto s' é o estado futuro observado e a é a ação executada em s. Define-se $V(s)=max_a Q(s,a)$ como sendo o valor máximo de recompensa acumulada que se pode receber estando em s. O parâmetro γ é um fator de ponderação entre recompensas imediatas e recompensas futuras que varia entre $0 \leq \gamma < 1$.

A taxa de aprendizado α é um fator de ponderação das atualizações dos valores $Q(s,a)$ cuja função é satisfazer as condições de convergência para uma política ótima em um ambiente não-determinístico. Nesse trabalho, o valor de α é atualizado conforme a regra:

$$\alpha_n = \frac{1}{1 + visitas_n(s,a)} \quad (1)$$

Sendo α_n o valor de α na n-ésima iteração e $visitas_n(s,a)$ o número de vezes em que o agente passou pelo estado s e selecionou a ação a, também na enésima iteração. Entretanto, caso $\alpha < 0.125$, seu valor é mantido em 0.125, sendo que a taxa de aprendizado nunca chega a zero.

No Q-learning define-se política ótima como sendo $\pi^* \equiv argmax_a Q^*(s,a)$. Para assegurar que o agente explore o ambiente com o qual está interagindo, a seleção de ações segue a regra gulosa ε – *Greedy*:

$$\pi(s) = \begin{cases} \arg\max_a Q(s,a) & q \leq p \\ a_{random} & q > p \end{cases} \quad (2)$$

Sendo $q \in [0,1]$ é um valor aleatório e p ($0 \leq p \leq 1$) um parâmetro que define a taxa de exploração/explotação do ambiente.

2.2 O algoritmo Q(λ)-learning

O algoritmo Q(λ) (Watkins, 1989; Peng e Williams, 1996) é uma técnica que combina o Q-learning com generalizações temporais – o Método de Diferenças Temporais TD(λ) (Sutton e Barto, 1998) – aproveitando a experiência de apenas uma interação do agente com o ambiente para atualizar valores de múltiplos pares estado-ação visitados anteriormente, representados pelo rastro de elegibilidade (Sutton e Barto, 1998).

A abordagem de Watkins (1989) reinicializa o rastro de elegibilidade sempre que uma ação é selecionada aleatoriamente pela regra ε – *Greedy*, enquanto a variante de Peng e Williams (1996) não diferencia uma ação aleatória de uma ação selecionada seguindo uma política, não reinicializando o rastro de elegibilidade. Como conseqüência, para uma política fixa não gulosa, a função Q do algoritmo Q(λ) de Peng e Williams (1996) não converge nem para Q^π, tampouco para Q^*, mas para algo híbrido entre as duas. No entanto, Sutton e Barto (1998) afirmam que, para uma política que se torne gulosa ao longo do tempo, esse método pode convergir para Q^*, além de apresentar um desempenho significativamente melhor que o algoritmo Q(λ) de Watkins (1989).

O algoritmo Q(λ) utiliza o fator λ, que varia entre $0 \leq \lambda \leq 1$, para descontar o erro de diferenças temporais TD(λ) dos próximos passos nas atualizações de $Q(s,a)$. O rastro de elegibilidade é utilizado para computar esses erros de maneira incremental, para que as atualizações possam ser efetuadas. Dessa forma, um valor $l(s,a)$ de rastro de elegibilidade é salvo para cada par estado-ação. O algoritmo Q(λ) implementado nesse trabalho segue a abordagem de Peng e Williams (1996), está descrito em Wiering e Schmidhuber (1998) e cuja regra de atualização é apresentada na tabela 2.

Tabela 2. O algoritmo Q(λ)

(1) Compute o erro de diferenças temporais TD(0) $e' = [r(s,a) + \gamma \cdot V(s') - Q(s,a)]$
(2) Compute o erro de diferenças temporais TD(λ) $e = [r(s,a) + \gamma \cdot V(s') - V(s)]$
(3) Para cada par estado-ação $(x,u) \in L$, faça:
(3a) Compute o decaimento do rastro $l(x,u) = \gamma \cdot \lambda \cdot l(x,u)$
(3b) Atualize a função $Q(x,u) = Q(x,u) + \alpha_n \cdot e \cdot l(x,u)$
(3c) Se $l(x,u) < \varepsilon$:
(3c.1) $L \leftarrow L \setminus (x,u)$
(3c.2) visitado(x,u) $\leftarrow 0$
(4) Atualize a função $Q(s,a) = Q(s,a) + \alpha_n \cdot e$
(5) Atualize o rastro de elegibilidade: $\forall u \neq a : l(s,u) \leftarrow 0; l(s,a) \leftarrow 1$
(6) Se visitado(s,a) = 0:
(6a) visitado(s,a) $\leftarrow 1$
(6b) $L \leftarrow L \cup (s,a)$

Nesse algoritmo, α obedece à mesma regra apresentada em (1), x representa um estado, u representa uma ação, representando os pares (x,u) de estados visitados. Esses pares visitados (x,u) são inseridos em uma lista duplamente ligada L e, caso o rastro de elegibilidade $l(x,u)$ seja menor que um limiar definido por $\varepsilon \geq 0$, o par é removido da lista. Para garantir que um par estado-ação não seja inserido mais de uma vez na lista L, variáveis binárias $visitado(x,u)$ são utilizadas para indicar que o par já foi visitado e se encontra presente na lista L.

Remover os pares *(x,u)* da lista *L* quando seu valor decai abaixo de ε pode aumentar a velocidade do algoritmo significativamente. Com a utilização do rastro de elegibilidade por substituição (*replacing eligibility traces*) (Sutton e Barto, 1998), conforme o quinto passo do algoritmo da tabela 2, é possível calcular o número máximo de pares estado-ação que a lista *L* pode conter, de acordo com os valores $\gamma\lambda$ e ε. Logo, o número de atualizações a cada iteração torna-se gerenciável. Também é importante observar que, caso $\lambda = 0$, o algoritmo Q(λ) torna-se idêntico ao algoritmo Q-learning.

2.3 O algoritmo QS-learning

O algoritmo QS (Ribeiro e Szepesvári, 1996) faz uso de generalizações espaciais e conhecimento prévio sobre o domínio para melhorar o desempenho do algoritmo Q-learning. Uma única experiência pode atualizar mais de um par estado-ação, de acordo com a similaridade entre esses pares. A similaridade é determinada através de uma função de espalhamento $\sigma(x,u,s,a)$, sendo $0 \leq \sigma(x,u,s,a) \leq 1$.

A similaridade pode ocorrer tanto no espaço de estados, como no conjunto de ações. Entretanto, esse trabalho considera apenas as similaridades que ocorrem no espaço de estados. Dessa forma, a função de espalhamento é definida como $\sigma(x,u,s,a) = g(x,s)\delta(u,a)$, onde $\delta(u,a)$ é o delta de Kronecker ($\delta(u,a)$ se $u = a$, ou $\delta(u,a) = 0$ se $u \neq a$). A função $g(x,s)$ determina a similaridade entre estados e é definida como $g(x,s) = \tau^d$, onde τ pode ser uma constante e d é um valor que quantifica a similaridade entre os estados x e s. A prova de convergência do algoritmo QS é mostrada em Ribeiro e Szepesvári (1996). A tabela 3 descreve o algoritmo QS.

Tabela 3. O algoritmo QS

```
Observe o estado atual s;
Repita infinitamente:
(1) Selecione uma ação a utilizando a regra
ε − Greedy;
(2) Receba a recompensa imediata r;
(3) Observe o novo estado s′;
(4) Para todo (x,u), atualize Q(x,u) se
σ(x,u,s,a) ≠ 0:
(4a) Compute o erro de diferenças temporais
TD(0) e′:
e'=[r(s,a)+γ·V(s')−Q(x,u)]
(4b) Atualize  Q(x,u)=Q(x,u)+σ(x,u,s,a)·α_n·e'
(5) s ← s′
```

No algoritmo QS, α também obedece à mesma regra apresentada em (1). Para satisfazer as condições necessárias à convergência do algoritmo, a função $\sigma(x,u,s,a)$ deve decair mais rapidamente que α.

É importante observar que, caso a função $g(x,s)$ não considere nenhuma similaridade entre os estados, seu valor será zero para qualquer outro estado x que não seja o próprio estado s, resultando em uma função de espalhamento $\sigma(x,u,s,a)$ também igual a zero. Logo, o algoritmo QS torna-se idêntico ao algoritmo Q-learning.

3 Uso de Heurísticas para Aceleração do Aprendizado por Reforço

A utilização de heurísticas para aceleração do AR, proposta por Bianchi (2004), pode ser definida como uma maneira de se resolver um PMD utilizando uma função heurística $H: S \times A \rightarrow \Re$ para influenciar a escolha das ações durante o aprendizado. A função heurística $H(s,a)$ define o quão desejável é a seleção da ação a, quando em s. Bianchi (2004) propõe, ainda, que a função heurística pode ser estacionária ou não-estacionária. Esse trabalho utiliza uma função heurística estacionária, baseado em Bianchi *et al.* (2004).

A função heurística é definida a partir de conhecimento prévio sobre o domínio e é utilizada apenas na seleção das ações a serem executadas pelo agente. Para tal, a regra de seleção de ações ε – *Greedy* é modificada para incluir a função heurística:

$$\pi(s) = \begin{cases} \arg\max_a [Q(s,a) + \xi.H(s,a)] & q \leq p \\ a_{random} & q > p \end{cases} \quad (3)$$

Nessa nova regra de seleção de ações, ξ é um número real que pondera a influência da heurística, existente para validar a prova de convergência, cujo valor é normalmente igual a 1 (Bianchi, 2004). Essa proposta mantém as principais características dos algoritmos de AR, minimizando o tempo necessário para a convergência.

Nesse trabalho as variantes dos algoritmos Q-learning, Q(λ) e QS que utilizam heurísticas para a aceleração do aprendizado foram implementadas considerando a nova regra de seleção de ações, sendo essa a única mudança necessária nos algoritmos.

O algoritmo Q-learning modificado para a utilização de heurísticas foi proposto por Bianchi *et al.* (2004), sendo denominado HAQ-learning. No entanto, até o presente momento, a técnica de aceleração de AR através de heurísticas não havia sido utilizada para os algoritmos Q(λ) e QS. Por esse motivo, considera-se como inéditas as variantes dos algoritmos Q(λ) e QS aceleradas por heurísticas, denominadas, respectivamente, HAQ(λ) (*Heuristically Accelerated Q(λ)-learning)* e HAQS (*Heuristically Accelerated QS-learning*).

4 Experimentos e Resultados

O ambiente de Futebol de Robôs simulado, proposto por Littman (1994), é definido por uma grade de 5x4 regiões que determina o espaço de estados e um conjunto de 5 ações – "Mover ao norte" (N), "Mover ao sul" (S), "Mover a leste" (E), "Mover a oeste" (W) e "Ficar parado" (*Stand*) – e não há informações escondidas. Dois jogadores (agente e oponente) competem entre si, sendo que a bola está sempre de posse de algum deles.

No simulador *SimuroSot* (FIRA, 2007), cuja tela é mostrada na figura 1, o ambiente é definido, seguindo as regras da categoria *MiroSot*, por um campo

de dimensões de 220x180 cm e robôs em forma de cubos de aresta de 7,5 cm com arquitetura cinemática diferencial. O simulador retorna as informações de posição e orientação dos robôs e posição da bola a cada iteração, desempenhando a função de um sistema de visão computacional. No entanto, tarefas de estratégia, planejamento de trajetória e controle de posicionamento e velocidade dos robôs são os problemas em aberto a serem resolvidos, tornando o ambiente do simulador muito mais complexo que o ambiente simplificado de Littman (1994).

Figura 1. O ambiente *Simurosot* utilizado nesse trabalho

4.1 Espaço de estados e conjunto de ações

A abordagem do problema de aprendizado foi feita com o intuito de manter o mesmo nível de abstração do espaço de estados e do conjunto de ações do ambiente proposto por Littman (1994). O campo foi dividido em uma grade de 7x5 regiões simétricas. Mas, como as regiões têm dimensões maiores que os robôs e a bola, os dois robôs e a bola podem ocupar uma mesma região ao mesmo tempo.

O conjunto de ações foi estendido para que os robôs possam manipular a bola, visto que a bola desloca-se livremente pelo campo, pois se trata de um ambiente dinâmico (o jogo não pára enquanto ações são selecionadas) e não-determinístico (selecionar uma ação *a* quando em *s* pode resultar diferentes estados futuros a cada execução de *a*). Além das 5 ações idênticas às de Littman (1994), duas ações, "Pegar a bola" e "Chutar no gol", foram adicionadas. A ação "Pegar a bola" possibilita que o robô se desloque até a região em que está a bola, posicionando-se imediatamente atrás (considerando o lado de ataque de cada jogador) da bola. A ação "Chutar no gol" determina o ponto de impacto do robô com a bola calculando-se do ângulo da bola em relação ao gol do adversário, fazendo com que o robô tente empurrar a bola até esse gol. A ação "Chutar no gol" apenas pode ser executada quando o robô estiver na mesma região em que estiver a bola, caso contrário considera-se impossível executar tal ação e o jogador permanece parado.

4.2 Função de recompensa e parâmetros de aprendizado

As recompensas utilizadas foram 1000 para cada gol feito pelo agente, -1000 para cada gol sofrido pelo agente, -10 para cada ação executada que não resultar em gol e -50 para cada ação cuja movimentação seja impossível de ser executada (movimentações para fora do campo, por exemplo).

Os parâmetros utilizados nos experimentos foram os mesmos para todos os algoritmos. A taxa de aprendizado, iniciada com $\alpha = 1$, decai de acordo com a regra (1). A taxa de exploração/explotação $p = 0.2$ e o fator de desconto $\gamma = 0.9$ são os mesmos utilizados por Littman (1994). Para os algoritmos Q(λ) e HAQ(λ), o valor $\lambda = 0.3$ foi determinado empiricamente, de acordo com as considerações feitas por Wiering e Schmidhuber (1998). Ainda, para os algoritmos QS e HAQS, o espalhamento segue a descrição de Ribeiro *et al*. (2002). No entanto, o valor inicial $\tau = 0.7$ decai de acordo com a regra:

$$\tau_n = \left[0.7 - 0.1 \cdot visitas_n(x,u)\right]^d \qquad (4)$$

Dessa forma, o valor de τ decai a zero a uma taxa maior que α, assegurando a convergência do algoritmo, lembrando que α nunca será menor que 0.125. De acordo com o quarto passo do algoritmo da tabela 3, para cada *(x,u)*, a quantização de similaridade *d* entre os estados assume valor 0 quando $x = s$. Assume valor 1 quando a posição do oponente em *x* for uma região horizontal ou verticalmente vizinha à sua posição em *s*. Quando a posição do oponente em *x* for uma região diagonalmente vizinha à sua posição em *s*, então *d* assume valor 2. Para os demais casos, *d* assume valor infinito. No entanto, para evitar que seja necessário percorrer todos os pares *(x,u)* possíveis, definiu-se manualmente os nove pares *(x,u)* a serem atualizados, seguindo as regras de quantização de similaridade. Um eixo de simetria horizontal foi definido, dividindo o campo exatamente ao meio. Isso foi feito para que o complemento dos nove pares *(x,u)* também fossem atualizados. Um exemplo simples consiste em imaginar ambos os robôs e a bola na região mais acima, à esquerda. O complemento desse estado resulta em ambos os robôs e a bola na posição mais abaixo, à esquerda, de acordo com o eixo de simetria horizontal. Ações também são complementadas quando necessário. Para o exemplo dado, caso um jogador tenha selecionado a ação "Mover acima", o complemento dessa ação será "Mover abaixo". Dessa forma, outros nove pares *(x,u)* são atualizados, aproveitando ainda mais a experiência de uma única iteração.

4.3 Experimentos

Os experimentos foram realizados em um microcomputador Pentium D 3.4 GHz com 1 GB de RAM. O aprendizado do agente ocorreu contra um oponente aleatório durante 5 rodadas de 500 jogos cada, consumindo 72 horas de simulação ininterrupta para ca-

da rodada de cada algoritmo. Cada jogo tem a duração de 5 minutos, independente do número de gols que ocorrem durante esse tempo. A maneira a qual a função de recompensa foi modelada determina que o objetivo do agente seja aprender a maximizar o saldo de gols ao longo do tempo em que está aprendendo. Por esse motivo, os gráficos apresentam o saldo de gols acumulado ao longo dos 500 jogos de aprendizado, com média e desvio padrão.

4.4 Resultados

O gráfico da figura 2 apresenta uma comparação entre os algoritmos Q-learning e HAQ-learning, enquanto a figura 3 apresenta o gráfico com as curvas dos algoritmos QS e HAQS.

Ao verificar os gráficos das figuras 2 e 3, percebe-se uma semelhança entre os algoritmos Q-learning e QS, assim como HAQ-learning e HAQS. Essa semelhança, também observada em Ribeiro *et al.* (2002), se deve ao fato de que o espalhamento das experiências ocorre apenas no início do aprendizado, onde o agente tem pouca experiência e poucas recompensas significativas (fazer ou sofrer gols) recebidas. Como a função de espalhamento decai rapidamente para manter o critério de convergência, os algoritmos QS e HAQS passam a se comportar exatamente como os algoritmos Q-learning e HAQ-learning, respectivamente, em pouco tempo de aprendizado.

O gráfico apresentado na figura 4 apresenta média e desvio padrão do saldo de gols acumulado para os algoritmos Q(λ) e HAQ(λ). Nota-se uma diferença de desempenho entre os algoritmos do gráfico da figura 4 em comparação com os demais.

Duas observações importantes podem ser feitas para os três gráficos apresentados. A primeira observação é que a utilização de heurísticas melhora significativamente o desempenho dos algoritmos, resultando em um acúmulo positivo no saldo de gols desde o início do aprendizado. A heurística é capaz de direcionar o agente ao seu objetivo quando o mesmo não tem experiência suficiente, enquanto sem a heurística o agente acaba por ter um comportamento excessivamente exploratório no início. A segunda observação é que, aproximando-se as curvas dos algoritmos por retas, nota-se uma inclinação maior nas retas que correspondem aos algoritmos que utilizam heurísticas, confirmando a aceleração do aprendizado.

Em relação às generalizações temporais em comparação com as generalizações espaciais, um ponto a ser considerado é o fato de que, enquanto os algoritmos Q(λ) e HAQ(λ) espalham a recompensa recebida por um rastro dos últimos pares estado-ação visitados, representando uma seqüência de diferentes posições dos jogadores e da bola e diferentes ações (diferentes experiências) que visam maximizar a recompensa, os algoritmos QS e HAQS espalham uma mesma posição do agente e da bola e uma mesma ação selecionada pelo agente (apenas uma experiência) por estados cuja posição do oponente tenha alguma similaridade com sua posição original. Dessa forma, a propagação das recompensas descontadas ao longo do tempo nos algoritmos QS e HAQS será muito mais próxima à propagação que ocorre nos algoritmos Q-learning e HAQ-learning. Esse fato justifica a semelhança entre os algoritmos Q-learning e QS, assim como entre os algoritmos HAQ-learning e HAQS, além de justificar o motivo pelo qual o algoritmo Q(λ) supera em eficiência os algoritmos Q-learning e QS, enquanto HAQ(λ) supera HAQ-learning e HAQS, conforme pôde ser observado nos gráficos apresentados.

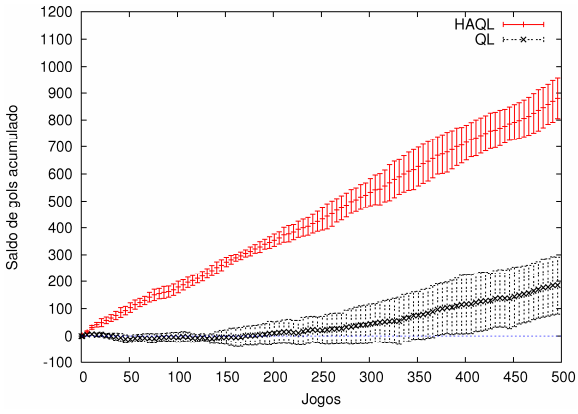

Figura 2. Saldo de gols acumulado dos algoritmos Q-learning e HAQ-learning

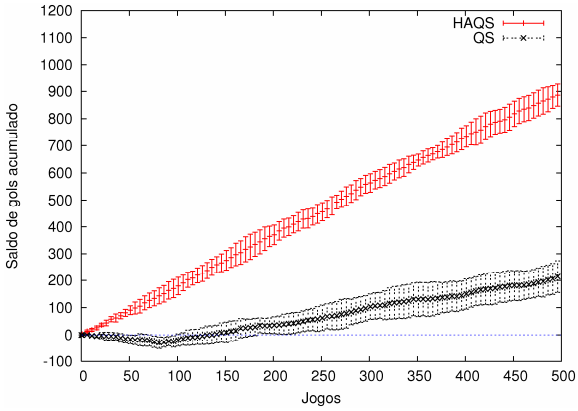

Figura 3. Saldo de gols acumulado dos algoritmos QS e HAQS

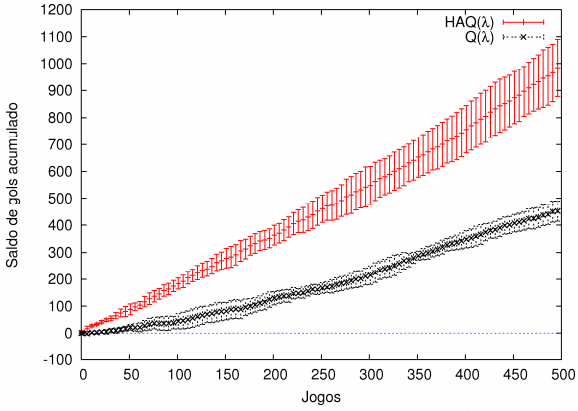

Figura 4. Saldo de gols acumulado dos algoritmos Q(λ) e HAQ(λ)

5 Conclusão e Trabalhos Futuros

Esse trabalho apresentou uma comparação entre algoritmos de AR bem conhecidos na literatura (Q-learning, QS e Q(λ)) com algoritmos que utilizam heurísticas para a aceleração do aprendizado por reforço – HAQ-learning e dois inéditos (HAQS e HAQ(λ)), sendo que os resultados obtidos indicam que a utilização de heurísticas acelera significativamente o aprendizado.

Para a realização das experiências, foi estendido o ambiente proposto por Littman (1994) a um ambiente mais complexo, dinâmico e não-determinístico, permitindo uma avaliação mais realística e visando o uso futuro em robôs reais da categoria *MiroSot*.

Trabalhos futuros irão abordar a execução de testes entre agentes que utilizem algoritmos de AR acelerados por heurísticas e oponentes que utilizem os algoritmos de AR tradicionais. Outros trabalhos futuros incluem a comparação entre os algoritmos Minimax-Q (Littman, 1994) e outros baseados em heurísticas, no ambiente simulado aqui apresentado e em robôs reais, além da verificação da possibilidade de aproveitamento da experiência adquirida em experimentos simulados em situações similares (com um mesmo espaço de estados), mas mais complexas (com um conjunto de ações expandido), por exemplo.

Referências Bibliográficas

Bianchi, R. A. C. (2004) "Uso de Heurísticas para a Aceleração do Aprendizado por Reforço". 174 f. Tese (Doutorado em Engenharia) – Escola Politécnica, Universidade de São Paulo, São Paulo.

Bianchi, R. A. C.; Ribeiro, C. H. C.; Costa, A. H. R. (2004). Heuristically Accelerated Q-Learning: a New Approach to Speed Up Reinforcement Learning. *Lecture Notes in Artificial Intelligence*, v. 3171, p. 245-254.

FIRA (2007) "Federation of International Robot-soccer Association", http://www.fira.net/soccer/, Fevereiro, 2007.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ICML'94)*, p. 157-163.

Peng, J.; Williams, R. J. (1996). Incremental Multi-Step Q-Learning. *Machine Learning*, v. 22, p. 283-290.

Ribeiro, C. H. C.; Szepesvári, C. (1996). Q-Learning combined with spreading: Convergence and results. In *Proceedings of the ISRF-IEE International Conference on Intelligent and Cognitive Systems (Neural Networks Symposium)*, p. 32-36.

Ribeiro, C. H. C.; Pegoraro, R.; Costa, A. H. R. (2002). Experience Generalization for Concurrent Reinforcement Learners: the Minimax-QS Algorithm. In: *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, p. 1239-1245.

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

Watkins, C. J. C. H. (1989) "Learning from Delayed Rewards", Tese (Doutorado) – Universidade de Cambridge, Cambridge.

Wiering, M.; Schmidhuber, J. (1998). Fast Online Q(λ). *Machine Learning*, v. 33, n. 1, p. 105-115.