ANÁLISE COMPARATIVA DE ALGUMAS TÉCNICAS PARA O ESTABELECIMENTO DE TRAJETÓRIAS EM AMBIENTES COM OBSTÁCULOS USANDO APRENDIZAGEM POR REFORÇO

AMANDA OLIVEIRA, G.

Depto. Engenharia de Computação e Automação - UFRN
e-mail: amandagondim@yahoo.com.br

JORGE MELO, D. DE, ADRIÃO NETO, D. D

Depto. Engenharia de Computação e Automação - UFRN
e-mail: jdmelo@dca.ufrn.br,adriao@dca.ufrn.br

Resumo - Este trabalho tem o objetivo de apresentar a análise comparativa de algumas técnicas para o estabelecimento de trajetórias em ambientes com obstáculos. Estas técnicas foram desenvolvidas com o uso de aprendizado por reforço, em particular os algoritmos *Q-Learning* e *R-Learning*. Para uma melhor compreensão desta análise, os resultados são apresentados em uma interface gráfica, que funciona como um ambiente de simulação para a geração das trajetórias. Nesta interface são apresentados alguns gráficos onde pode-se analisar o comportamento dos algoritmos implementados.

Palavras-chave – Aprendizado por reforço, *Q-Learning*, *R-Learning*, Robôs móveis.

1. Introdução

O problema de trajetórias em ambientes com obstáculos se apresenta na utilização de sistemas autônomos tais como robôs móveis. Tais robôs devem aceitar descrições de alto nível das tarefas que eles devem desenvolver, sem a necessidade de maiores intervenções humanas. As descrições de entrada especificam o que o usuário deseja que seja feito, e não como proceder para fazê-lo. Para tanto, estes robôs são equipados com atuadores e sensores sob controle de um sistema de computação. (Ottoni e Lages, 2003)

Os temas de pesquisa na área da robótica móvel vão desde modelagem e estratégia de controle até tipos de sistemas de locomoção e técnicas de inteligência artificial. O controle em baixo nível concentra-se nos atuadores, enquanto que o controle de alto nível projeta arquiteturas para gerenciamento de tarefas e planejamento de trajetórias, que de uma forma geral, consiste em se descobrir de que forma se pode levar um objeto a partir de uma configuração inicial até uma configuração final. (Schroeder et al., 2005)

Neste trabalho busca-se apenas o estabelecimento das trajetórias em ambientes com obstáculos, sem se preocupar com o controle em baixo nível. Desta forma, não é possível uma aplicação direta do mesmo a robôs móveis, já que isto envolveria o estudo de maiores detalhes de percepção e controle. Nada impede, porém, que estes temas sejam estudados e associados com as trajetórias estabelecidas neste simulador para que o projeto possa ser aplicado neste contexto.

As trajetórias são estabelecidas com o uso de técnicas de aprendizado de reforço (AR), onde se programa agentes utilizando recompensas e punições para resolver tarefas específicas através de interações com o ambiente.

AR é baseado na existência de um crítico externo ao ambiente e utiliza uma estrutura composta de estados, ações e recompensas, conforme mostra a Figura 1.

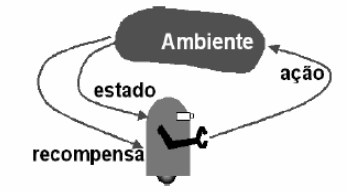

Figura 1 - Modelo Padrão de Aprendizagem por Reforço

O agente atua em um ambiente descrito por um conjunto de possíveis estados e pode executar, para cada estado, uma ação dentro de um conjunto de ações possíveis, recebendo um valor de reforço a cada vez que executa uma ação. Este reforço indica o valor imediato da transição estado - ação-novo_estado. Ao longo do tempo, este processo produz uma seqüência de pares estado-ação, e seus respectivos valores de reforços. O objetivo do agente é aprender uma política que maximize uma soma esperada destes reforços em longo prazo. (Guelpeli et al., 2003)

Sendo assim, este trabalho tem como objetivo principal estudar, implementar e comparar os tradicionais algoritmos de aprendizado por reforço, *Q-Learning* e *R-Learning*. Além disto, tem-se o desenvolvimento de uma interface gráfica apropriada para a análise dos resultados, que funcionará como um simulador para o estabelecimento das trajetórias em ambientes com obstáculos. Desta forma, este trabalho apresenta uma significativa importância neste contexto, já que a simulação é uma poderosa ferramenta no processo de aprendizagem, onde imita-se propriedades e comportamentos do sistema real, permitindo sua manipulação e um estudo detalhado.

Este artigo está organizado da seguinte forma: Nas seções 2 e 3 são mostradas a formulação do

problema e a descrição do ambiente de treinamento, respectivamente. Detalhes da implementação e da interface do programa, são apresentados nas seções 4 e 5. Os resultados e conclusões são apresentados nas seções 6 e 7.

2. Formulação do Problema

O problema principal deste trabalho consiste em orientar um agente a se mover em um ambiente desconhecido, desviando de eventuais obstáculos que o ambiente venha a apresentar, até alcançar um objetivo estabelecido. Para aprender a movimentar-se dentro do ambiente virtual o agente recebe um treinamento utilizando algoritmos de aprendizado por reforço. Neste treinamento o agente terá como objetivos: aprender a chegar ao estado final e aprender a evitar os obstáculos do ambiente.

Para mover-se dentro do ambiente o agente poderá realizar 4 diferentes ações, são elas: um passo para cima, um passo para baixo, um passo para a esquerda, um passo para a direita.

Em cada iteração do algoritmo, uma ação é escolhida para definir o movimento do agente no ambiente. Esta escolha é feita a partir da posição atual do agente, onde o algoritmo consulta uma tabela com os valores de utilidade estimados para os pares (estado-ação), para então definir a ação mais adequada no momento. Cada ação tomada pelo agente retorna um valor de reforço, que é utilizado na atualização da função valor do algoritmo para influenciar em futuras escolhas de ações. O valor de reforço retornado está diretamente relacionado com o status do estado acessado. Sendo assim, a trajetória do agente dentro do ambiente consiste em uma seqüência de passos que são definidos através das ações em cada iteração.

O treinamento do agente consiste em fazê-lo traçar n trajetórias de pontos iniciais escolhidos aleatoriamente até o ponto objetivo. O valor de n deve ser suficientemente grande para que o agente aprenda a se deslocar de forma ótima de qualquer ponto do ambiente até o seu objetivo.

3. Descrição do Ambiente

Para o mapeamento, o ambiente foi dividido em intervalos finitos, de forma que o mesmo pudesse ser representado como uma matriz, onde cada elemento desta matriz é considerado um estado. Sendo assim, o ambiente se apresenta como uma malha de n estados, organizados em linhas e colunas.

Os estados da malha são identificados por uma numeração de 0 a n-1, onde os mesmos são ordenados dentro da malha da esquerda para a direita e de cima para baixo. Cada estado do ambiente pode apresentar o status de permitido, obstáculo ou objetivo. Onde:

Permitido: significa que o estado possui livre acesso e o agente recebe um valor de retorno nulo ao acessá-lo.

Obstáculo: significa que o estado tem acesso bloqueado e o agente recebe um retorno com valor negativo ao acessá-lo. Pode representar uma parede, um móvel, ou qualquer outro tipo de obstáculo que o robô possa encontrar dentro de um ambiente e tenha que desviar.

Objetivo: representa o estado final da trajetória, ou seja, o objetivo a ser alcançado pelo agente dentro do ambiente. Caso o agente chegue a este estado, receberá um retorno de valor positivo como recompensa.

4. Implementação do Ambiente

O sistema foi desenvolvido com a linguagem C++ e com o auxílio do ambiente de programação Qt-Designer versão 3.3.4.

Inicialmente implementou-se o algoritmo de aprendizado *Q-Learning*, que propõe que o agente aprenda uma função Q de recompensa esperada com desconto, conhecida como função valor-ação. Esta função de estimação Q é definida como sendo a soma do reforço recebido pelo agente por ter realizado a ação a_t no estado s_t em um momento t, mais o valor (descontado de γ) de seguir a política ótima daí por diante.

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma V(s_{t+1})$$

O pseudocódigo do algoritmo *Q-Learning* implementado pode ser observado a seguir:

```
Inicialize Q(s_t, a_t)
  Para cada episodio t repita:
    Inicialize s_t
    Repita(Para cada passo do episodio):
      Observe s_t e escolha uma ação a_t
      Observe s_t+1 , r(s_t, a_t) e atualize Q(s_t, a_t)
    Até s_t ser o estado final
  Até t igual a limite de passos.
```

Onde:

- $r(s_t, a_t)$: Reforço recebido no estado s_t
- t : Seqüência discreta de passos no tempo
- $Q(s_t, a_t)$: Valor da ação a_t no estado s_t

Os valores de utilidade $Q(s_t, a_t)$ estimados para os pares (estado, ação) podem ser armazenados em uma tabela. A atualização destes valores é feita de acordo com a seguinte equação:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\left[r(s_t, a_t) + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right]$$

Onde:

α é a taxa de aprendizado

γ é a taxa de desconto temporal

Outro algoritmo escolhido para implementar foi o *R-Learning*, que tem como objetivo maximizar a

recompensa média a cada passo, ao contrário do *Q-Learning* que maximiza os descontos acumulados de recompensa. Para esta implementação, tomou-se como base o algoritmo *R-Learning* aplicado a problemas de horizonte infinito. Cada episódio deste algoritmo consiste apenas em aprender a evitar os obstáculos do ambiente durante um numero t de iterações, sem o objetivo de alcançar um estado final. Porém, com o intuito de adequar o algoritmo ao problema proposto no trabalho, fizeram-se necessárias algumas modificações no algoritmo original. A principal delas foi a adaptação do mesmo à problemas de horizonte finito com a introdução de episódios e de um objetivo no treinamento. Além disso, a fim de evitar a retenção do agente em um pequeno grupo de estados durante o treinamento, implementou-se uma estratégia para evitar o retorno do robô ao estado visitado no momento imediatamente anterior. Isto foi feito mediante a aplicação de uma punição no valor de retorno. O pseudocódigo utilizado na implementação do algoritmo *R-Learning* pode ser observado a seguir.

```
Inicialize ρ e R(s_t, a_t)
  Para cada episodio t repita:
    Inicialize s_t
    Repita (Para cada passo do episodio):
      Observe s_t e escolha uma ação a_t
      Observe s_{t+1} e o retorno r(s_t, a_t)
      Se s_{t+1} = s_{t-1} então r = r - x
      Atualize R(s_t, a_t)
      Se R_t(s_t, a_t) = max_a R_t(s_t, a) Atualize ρ
    Até s_t ser o estado final
  Até que t igual a limite de passos
```

Onde:

- $r(s_t, a_t)$: Reforço recebido no estado s_t
- t : Seqüência discreta de passos no tempo
- $R(s_t, a_t)$: Valor da ação a_t no estado s_t
- x : Punição adicional caso o agente volte ao estado anterior.

A atualização dos valores da função $R(s_t, a_t)$ e do parâmetro ρ é feita de acordo com as seguintes equações:

$$R_{t+1}(s_t, a_t) = R_t(s_t, a_t) + \alpha\left[r - \rho + \max_a R_t(s_{t+1}, a) - R_t(s_t, a_t)\right]$$

$$\rho \leftarrow \rho + \beta\left[r - \rho + \max_a R_t(s_{t+1}, a) - \max_a R_t(s_t, a)\right]$$

Tradicionalmente, a escolha da ação a_t a partir do estado atual s_t, consistiria simplesmente em se escolher a ação com maior valor de utilidade na tabela, na linha referente ao estado s_t. Porém, com o intuito de resolver o dilema Investigação X Exploração, que consiste em decidir quando se deve aprender ou não sobre o ambiente, a escolha de a_t foi feita utilizando a técnica Є – Greedy. Nesta técnica, o agente executa a ação com o maior valor de Q e R com probabilidade 1 – Є e escolhe uma ação aleatória com probabilidade Є.

Com o intuito de acelerar o treinamento dos algoritmos *Q-Learning* e *R-Learning*, algumas estratégias adicionais foram implementadas. São estas:

- Estabelecimento de um número limite de ações por episódio, ou seja, o agente terá um número máximo de passos dentro do ambiente para conseguir alcançar o seu objetivo. Caso este limite seja excedido, o agente deverá abandonar o episódio atual e iniciar o próximo a partir de um estado inicial aleatório.
- Estabelecimento de um número limite de colisões com obstáculos por episódio, ou seja, caso o agente exceda o limite de colisões estabelecido, ele deverá abandonar o episódio atual e iniciar o próximo a partir de um estado inicial aleatório.

Estas técnicas foram implantadas tanto no *Q-Learning* quanto no *R-Learning*, e o seu uso possibilitou uma significante melhoria no desempenho dos algoritmos no que se diz respeito à velocidade de aprendizado. Esta melhoria deve-se a eliminação do gasto de tempo computacional utilizados em casos desnecessários, onde o agente perdia muito tempo colidindo com obstáculos ou explorando longos caminhos que não o levavam ao objetivo.

5. Interface

A interface do programa é o espaço em que o usuário pode simular trajetórias em ambientes configurados por ele mesmo. Este ambiente virtual permite ao usuário escolher as posições dos obstáculos e do objetivo do ambiente em questão, bem como escolher o algoritmo de treinamento do agente, *Q-Learning* e/ou *R-Learning*. Uma outra característica deste ambiente é possibilitar a comparação de desempenho entre os dois algoritmos implementados.

Através da figura 2, pode-se notar que a interface do programa apresenta algumas regiões de configuração e de interpretação de resultados. A configuração do ambiente consiste em informar os parâmetros de construção do mesmo e os de treinamento dos algoritmos. Na construção do ambiente virtual deve-se representar o mesmo através de uma malha, definindo-se a quantidade de linhas e colunas que a compõe. A partir daí então, deve-se escolher as posições dos obstáculos e do objetivo. Os parâmetros de treinamento estão relacionados às estratégias de aceleração de aprendizado implementadas. Deve-se configurar então o número de iterações para o treinamento, o número limite de ações por episódio e o número limite de colisões por episódio.

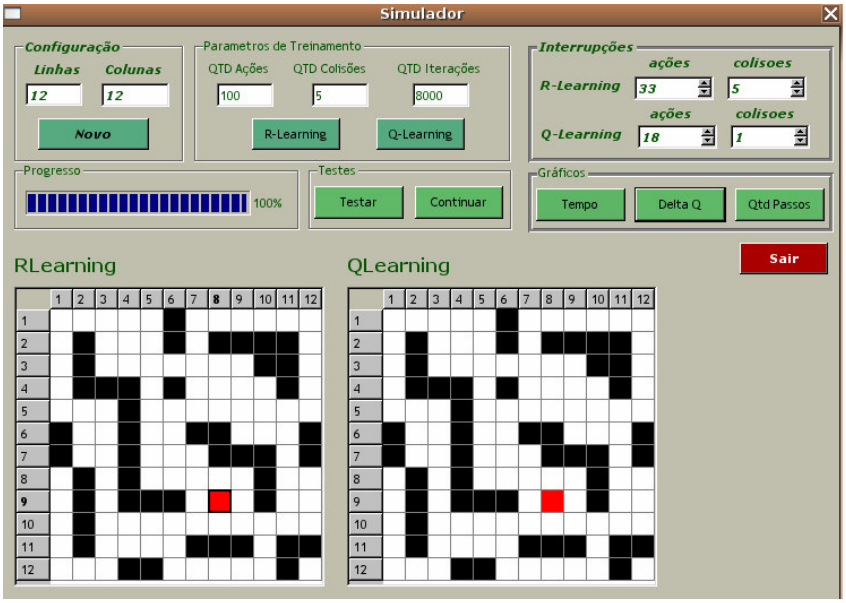

Figura 2 – Interface do Simulador

Para a visualização do resultado obtido com o treinamento, a interface apresenta duas malhas idênticas, representando o mesmo ambiente virtual. Cada uma destas malhas apresenta a trajetória sugerida por um dos algoritmos de aprendizado, desde um ponto qualquer escolhido pelo usuário até o objetivo do ambiente. Desta forma, é possível analisar e comparar o desempenho dos dois algoritmos para a situação configurada. Outra maneira de observar o comportamento dos algoritmos no treinamento é através da análise dos gráficos gerados pelo simulador. Além disso, esta interface também apresenta um quadro de dados onde é possível analisar o comportamento das iterações de cada algoritmo no decorrer do treinamento.

6. Resultados

A análise dos resultados baseou-se principalmente nos dados contidos nos gráficos do simulador. Estes gráficos analisaram três aspectos das performances dos algoritmos: tempo de treinamento por episódio, convergência dos dados e qualidade das trajetórias geradas.

O experimento foi realizado levando-se em consideração três diferentes situações: ambientes sem obstáculos, com poucos obstáculos e com muitos obstáculos. Para cada uma destas situações foram testados 10 ambientes com configurações diferentes, de forma que a análise pudesse ser feita tomando-se como base as médias obtidas em cada situação.

No aspecto tempo de treinamento por episódio, analisou-se o tempo, em *ms*, que cada algoritmo utilizou para chegar ao estado final do ambiente, partindo de um estado inicial escolhido aleatoriamente. Como os treinamentos utilizando o *Q-Learning* e o *R-Learning* são independentes entre si, a comparação dos tempos de treinamento foi realizada através da média destes, como pode ser observado na figura 3. Como para cada uma situações estudadas 10 ambientes foram simulados, as médias do tempo de treinamento por episódio obtidas, em *ms*, podem ser observadas na tabela 1.

Ambiente	Sem Obstáculos	Poucos Obstáculos	Muitos Obstáculos	Média Amostral
Média QL	0.01781	0.01736	0.01325	**0.01614**
Média RL	0.01650	0.01630	0.01308	**0.01529**

Tabela 1 – Médias do tempo de treinamento por episódio

Como pode-se observar, os dois algoritmos apresentam estimativas de médias de tempo de treinamento bastante próximas, o que faz com que este aspecto não possua grande influência na comparação da performance dos algoritmos.

O aspecto convergência foi analisado através da variação dos valores de utilidade das tabelas referentes às funções R e Q, de forma que quanto menores forem as variações ao longo do treinamento, melhor será a convergência do algoritmo.

A variação dos valores de utilidade, calculada ao fim de cada episódio, é dada por:

$$\Delta Q = \max_{i,j}\left[Q_{t+1}(i,j) - Q_t(i,j)\right] \ , \ \Delta R = \max_{i,j}\left[R_{t+1}(i,j) - R_t(i,j)\right]$$

Onde: $Q(i, j)$ e $R(i, j)$ correspondem ao valor associado ao par estado-ação (s = i, a = j).

A figura 4 apresenta o gráfico de análise da convergência, onde mostram-se as variações dos valores em cada episódio, bem como a média destas variações nos últimos 10% dos episódios do treinamento. As médias obtidas em cada situação estudada podem ser observadas na tabela 2.

Ambiente	Sem Obstáculos	Poucos Obstáculos	Muitos Obstáculos	Média Amostral
Média QL	$3{,}46\text{x}10^{-4}$	$3{,}41\text{x}10^{-4}$	$3{,}09\text{x}10^{-4}$	$3{,}32\text{x}10^{-4}$
Média RL	$5{,}49\text{x}10^{-4}$	$5{,}51\text{x}10^{-4}$	$5{,}19\text{x}10^{-4}$	$5{,}40\text{x}10^{-4}$

Tabela 2 – Médias das variações dos valores de utilidade nos últimos 10% de episódios.

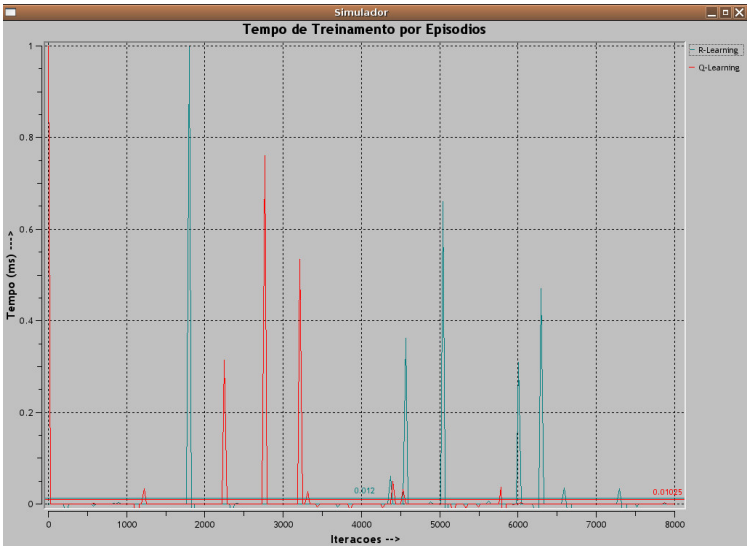

Figura 3 – Gráfico do tempo de treinamento por episódio

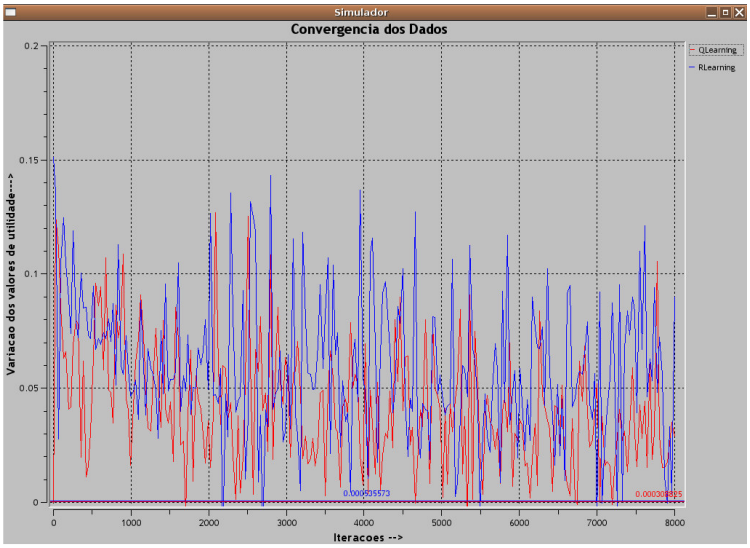

Figura 4 – Gráfico de análise de convergência

Através destas informações é possível observar que os dois algoritmos apresentaram uma boa convergência nos dados, independente da situação analisada. Além disso, de acordo com o gráfico, pode-se notar que durante todo o treinamento o *Q-Learning* quase sempre apresenta uma variação de valores menor que o *R-Learning*.

Embora não se tenha alcançado a convergência ideal (variação dos dados igual a zero), os algoritmos foram capazes de gerar trajetórias bastante satisfatórias.

A qualidade das trajetórias geradas foi medida através da quantidade de passos que cada algoritmo necessita para alcançar o objetivo partindo de um estado inicial aleatório. Sendo assim, o gráfico apresentado na figura 5 mostra exatamente as diferenças das quantidades de passos necessários no *R-Learning* e no *Q-Learning* em trajetórias aleatórias, bem como a média destas diferenças. Para gerar o gráfico foram testadas um número de trajetórias igual a 50% da quantidade de estados que compõem o ambiente.

As médias obtidas da diferença de passos utilizados em cada algoritmo encontram-se na tabela 3.

Ambiente	Sem Obstáculos	Poucos Obstáculos	Muitos Obstáculos	Média Amostral
Média da Diferença	- 0.36944	- 0.54666	- 0.41666	- 0.4442

Tabela 3 – Médias da diferença de passos no R-Learning e no Q-Learning

Como pode-se notar, independente da configuração do ambiente, as médias das diferenças são sempre negativas, o que indica que o *R-Learning* apresenta trajetórias melhores que o *Q-Learning*, contendo em média 0,44 passos a menos em suas trajetórias.

Uma outra forma de se analisar a performance dos algoritmos implementados é através do quadro de interrupções que pode ser observado no canto superior direito da figura 3. Através deste quadro é possível analisar o comportamento das iterações ao longo do treinamento, ou seja, observar o número de iterações que foram interrompidas por exceder o limite de ações e/ou colisões.

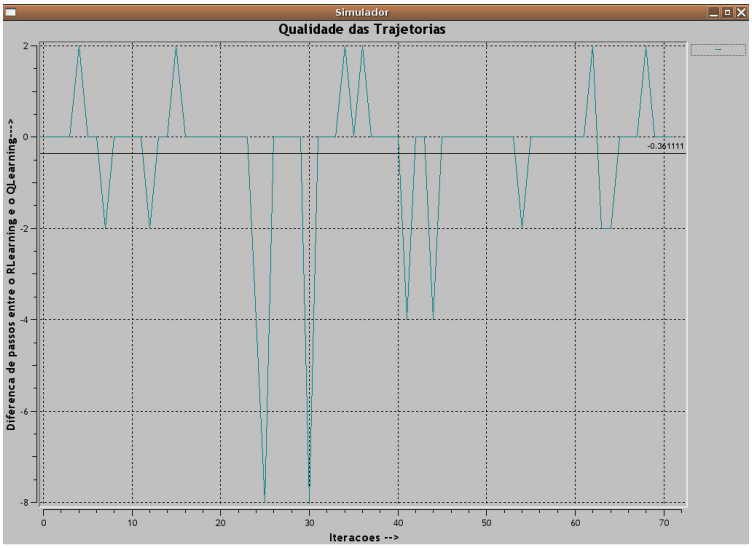

Figura 5 – Gráfico de análise das trajetórias

As médias das quantidades de iterações interrompidas são apresentadas na tabela 4.

	Sem Obstáculos		Poucos		Muitos		Média Amostral	
	QL	RL	QL	RL	QL	RL	QL	RL
Ações	17,6	61,7	21,9	43,4	26	44,5	21,8	49,8
Colisões	1,9	1,7	2,3	1,9	2,6	4,3	2,26	2,63

Tabela 4 – Médias das quantidades de iterações interrompidas

Como pode-se observar, independente da quantidade de obstáculos que o ambiente possua, o *R-Learning* sempre apresenta um número maior de iterações interrompidas por exceder o limite de ações que o *Q-Learning*. Além disso, durante os testes, notou-se que o *R-Learning* mostrou-se bem mais sensível à diminuição do limite de ações por episódio que o *Q-Learning*. Isto significa que no início do treinamento o *R-Learning* apresenta uma maior dificuldade para convergir e que esta interrupção por limite de ações funciona como um acelerador neste estágio inicial.

7. Conclusões

Tendo-se observado os resultados obtidos, pode-se concluir que os dois algoritmos mostraram-se bastante satisfatórios em todas as configurações de ambientes.

Como o tempo e a convergência apresentaram valores muito próximos para os dois algoritmos, estes aspectos não apresentaram grande influência na comparação da performance dos mesmos. Porém no que se diz respeito à qualidade das trajetórias, pode-se dizer que o *R-Learning* mostrou-se mais eficiente que o *Q-Learning*, apesar de apresentar maior dificuldade para convergir.

Uma importante observação a se fazer é que o desempenho dos algoritmos está diretamente ligado à escolha dos valores dos parâmetros de aprendizado, que devem ser testados e adequados corretamente ao problema em questão.

Sendo assim como conclusão geral, têm-se que, as trajetórias geradas pelo *R-Learning* são mais otimizadas, porém, como conseqüência da dificuldade de convergência, o mesmo necessita de um número maior de iterações que o *Q-Learning* para gerá-las.

8. Referências Bibliográficas

[1] Bianchi, R. A da C. Uso de heurísticas para a aceleração do aprendizado por reforço. Tese (Doutorado) - Escola politécnica da universidade de São Paulo, 2004.

[2] Guelpeli, M.V.C; Ribeiro, C.H.C; Omar, N. Utilização de Aprendizagem por reforço para modelagem autônoma do aprendiz em um tutor inteligente. XIV Simpósio Brasileiro de Informática na Educação - NCE - IM/UFRJ, p. 2, 2003.

[3] Ottoni, G. de L; Lages, W.F. Navegação de robôs móveis em ambientes desconhecidos utilizando sonares de ultra-som. SBA: Controle & Automação Sociedade Brasileira de Automática, p. 3, 2003.

[4] R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge, Massachusetts, 1998.

[5] Schroeder, G.N.; Espindola, D.B; Botelho,S.S. da C. ; Bicho, A. de L; Oliveira, V.M. de. Simulador Gráfico para Controle de Robôs Móveis Omnidirecionais. Journal of Computer Science - Volume 4 - Número 4, p.3, 2005.

[6] Silva, M. M; Dória, A.D; Melo, J.D. Um experimento didático de utilização da aprendizagem por reforço em robôs móveis. UNP – UFRN. Relatório Técnico, 2004.