Aprendizado por reforço relacional para o controle de robôs sociáveis

Renato Ramos da Silva

Aprendizado por reforço relacional para o controle de robôs sociáveis

Renato Ramos da Silva

Orientadora: ***Profa. Dra. Roseli Aparecida Francelin Romero***

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional.

USP - São Carlos
Janeiro/2009

.

A essa etapa concluída como parte
de um grande processo que está
sobre a mão divina do Senhor.

Agradecimentos

Primeiramente a Deus pelo dom da vida, e pela graça de todos os dias encontrar motivos para amá-Lo mais em todas as coisas e, a cada descoberta, poder amá-Lo mais que todas as coisas por meio das quais Ele se revela.

Aos meus pais, por terem me dado a riqueza de conhecimentos mais valiosos do que o melhor dos resultados que esta dissertação pudesse mostrar. Pelo carinho incondicional e pela perseverança em amar, sempre. Aos meus irmãos, primos, familiares e entes queridos, por estarem sempre próximos de coração mesmo quando a distância torna raros os encontros e os abraços.

À minha namorada Thaís, na qual eu dedico cada linha desta dissertação ao amor verdadeiro que de Deus brota em nossos corações. Que está seja oferta de todo carinho e compreensão que tive de você neste tempo.

À professora Dra. Roseli Aparecida Francelin Romero, pela amizade e confiança adquiridas durante esses anos de convivência, e por todo apoio para a realização deste trabalho. Do que ficou do aprendizado da sua orientação científica esta dissertação até pode dizer um pouco. Contudo, o que ficou do aprendizado de coragem, honestidade, humildade e de vida não cabe aqui em palavras.

Aos amigos do Grupo de Oração Universitário, e a todo o Ministério Universidades Renovadas da Renovação Carismática Católica. Obrigado pela amizade e por serem minha família em todo lugar onde eu caminhe. Sejam canais de Deus para fazer a diferença na vida de muitos nas universidades mundo afora, como fizeram na minha!

Ao amigo Claudio, que não somente iniciou esse projeto, mas também por toda a dedicação, amizade e paciência. Obrigado por me fazer acreditar que este trabalho vale a pena!

Ao Prof. Ednaldo Brigante Pizzolato, pela ajuda e participação nesta jornada.

A todos os amigos do Labic, Biocom e LAR. Muito obrigado pela convivência, amizade e pelo crescimento que me proporcionaram. A todos os professores, funcionários e alunos do ICMC-USP, pela saudável convivência e por todo apoio dado durante a realização deste trabalho.

Aos amigos Carlos, Mário, Lucas, Fernando e Julio, pela amizade, carinho e pela convivência de vida. Obrigado por cada momento.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro.

Resumo

A Inteligência Artificial não busca somente entender mas construir entidades inteligentes. A inteligência pode ser dividida em vários fatores e um deles é conhecido como aprendizado. A área de aprendizado de máquina visa o desenvolvimento de técnicas para aprendizado automático de máquinas, que incluem computadores, robôs ou qualquer outro dispositivo. Entre essas técnicas encontra-se o Aprendizado por Reforço, foco principal deste trabalho. Mais especificamente, o aprendizado por reforço relacional (ARR) foi investigado, que representa na forma relacional o aprendizado obtido através da interação direta com o ambiente. O ARR é bem interessante no campo de robótica, pois, em geral, não se dispõe do modelo do ambiente e se requer econômia de recursos utilizados. A técnica ARR foi investigada dentro do contexto de aprendizado de uma cabeça robótica. Uma modificação no algoritmo ARR foi proposta, denominada por ETG, e incorporada em uma arquitetura de controle de uma cabeça robótica. A arquitetura foi avaliada no contexto de um problema real não trivial: o aprendizado da atenção compartilhada. Os resultados obtidos mostram que a arquitetura é capaz de exibir comportamentos apropriados durante uma interação social controlada, através da utilização do ETG. Uma análise comparativa com outros métodos foi realizada que mostram que o algoritmo proposto conseguiu obter um desempenho superior na maioria dos experimentos realizados.

Abstract

The Artificial Intelligence search not only understand but to build intelligent entities. The intelligence can be divided into several factors and one of them is known as learning. The area of machine learning aimed at the development techniques for automatic learning of machinery, including computers, robots or any other device. Reinforcement Learning is one of those techniques, main focus of this work. Specifically, the relational reinforcement learning was investigated, which is use relational representation for learning obtained through direct interaction with the environment. The relational reinforcement learning is quite interesting in the field of robotics, because, in general, it does not have the model of environment and economy of resources used are required. The relational reinforcement learning technique was investigated within the context of learning a robotic head. A change in the relational reinforcement learning algorithm was proposed, called TGE, and incorporated into an architecture of control of a robotic head. The architecture was evaluated in the context of a real problem not trivial: the learning of shared attention. The results show that the architecture is capable of displaying appropriate behavior during a social interaction controlled through the use of TGE. A comparative analysis was performed with other methods show that the proposed algorithm has achieved a superior performance in most experiments.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Símbolos

AM Aprendizado de Máquina

AR Aprendizado por Reforço

ARR Aprendizado por Reforço Relacional

DT Diferença Temporal

IA Inteligência Artificial

LPO Linguagem de Primeira Ordem

PD Programação Dinâmica

PDM Processo de Decisão de Markov

SARSA *State-Action-Reward-State-Action*

TDIDT *Top-down Induction of Decision Trees*

TILDE *Top-down Induction of Logical Decision Trees*

A Conjunto de ações

a Ação

a_t Ação no tempo t

$ergodic$ a probabilidade de se sair de qualquer estado e ir para outro sob uma política deve ser diferente de zero

S Conjunto de estados

s Estado

s_t Estado no tempo t

γ Fator de desconto

δ Função de transição

$\delta(s, a)$ Estado resultante da execução da ação a sobre o estado s

$\delta(s_t, a_t)$ Estado resultante da execução da ação a sobre o estado s no tempo t

$e_t(s)$ O traço de eligibilidade de um estado s no tempo t

λ Parâmetro de decaimento do traço de elegibilidade

π Política

π^* Política ótima

r Recompensa

r_t Recompensa recebida no tempo t

$r(s_t, a_t)$ Recompensa recebida por executar a ação a sobre o estado s no tempo t

ρ Recompensa média

Q Valor Q

$\hat{Q}$ Estimativa ou hipótese do valor Q(aproximação da função Q)

$R^{\pi}(s, a)$ valor ajustado da média ao realizar uma ação a no estado s, seguindo uma política π

V Função de avaliação ou custo

V^{π} Função de avaliação ou custo seguindo a política π

V^* Função de avaliação ou custo seguindo a política ótima

$V^{\pi}(s)$ Valor do estado s seguindo a política π

$V(s_t)$ Valor do estado s

α taxa de aprendizagem

ϑ° Parâmetro de fóvea

θ_r Posição da cabeça do robô

θ_a Posição da cabeça do ser humano

θ_{oi} Ângulo entre este objeto e o foco do robô

CAPÍTULO

1

Introdução

1.1 Contexto e motivação

Existem na literatura muitas definições sobre Sistemas Inteligentes. Em Russell e Norvig (2003) podem ser encontradas oito definições sobre Sistemas Inteligentes, tais como, sistemas que pensam como seres humanos (Haugeland, 1985; Bellman, 1978), sistemas que agem como seres humanos (Kurzweil, 1990; Rich e Knight, 1991), sistemas que pensam racionalmente (Charniak e McDermott, 1991; Winston, 1998) e sistemas que agem racionalmente (Poole et al., 1998; Nilsson, 1998). Apesar das diferenças existentes, elas se ajudam na construção dessas entidades.

Adotando-se a definição de sistemas que agem como seres humanos, Russell e Norvig (2003) cita a utilização do teste de *Turing* em Turing (1950), mostrando que para um agente ser considerado inteligente ele deverá ter demonstrar as seguintes capacidades:

- **Processamento de linguagem natural** - para capacitá-lo a comunicar-se.
- **Representação do conhecimento** - para armazenar o seu conhecimento.
- **Raciocínio** - para utilizar o conhecimento para responder questões ou fazer novas conclusões.
- **Aprendizado de máquina** - para adaptar-se a novas circunstâncias e para detectar e extrapolar padrões.

Dentre as capacidades citadas, somente o processamento de linguagem natural não será abordado de forma explícita no prsente trabalho.

A área de Aprendizado de Máquina (AM) é dedicada ao desenvolvimento de algoritmos e técnicas que permitam ao computador aprender, isto é, que permitam ao computador aperfeiçoar o desempenho em alguma tarefa. Nos últimos anos, pesquisas nesta área têm procurado substituir a programação explícita pelo processo de ensinar uma tarefa. Muitos tipos diferentes de paradigmas de aprendizado podem ser encontrados na literatura. (Mitchell, 1997; Faria, 2000).

Dentre esses paradigmas encontra-se o paradigma de Aprendizado por Reforço (AR). O AR consiste no aprendizado via experimentação direta. Não se assume a existência de um professor provedor de exemplos a partir do qual o aprendizado se desenvolve. Em AR, a experiência é o único elemento que atua como auxiliar no processo de aprendizado. Com raizes históricas no estudo de reflexos condicionados, o AR logo atraiu o interesse de Engenheiros e Cientistas da Computação por sua relevância teórica e aplicações potenciais em campos tão diversos quanto Pesquisa Operacional e Robótica (Ribeiro, 1999).

Computacionalmente, AR opera em um ambiente de aprendizagem composto por dois elementos: o aprendiz e um processo dinâmico. A passos de tempo sucessivos, o aprendiz faz uma observação do estado de processo, seleciona uma ação e a aplica de volta ao processo. Sua meta é descobrir uma política de ações que controle o comportamento deste processo dinâmico, usando para isso sinais (reforços) indicativos de quão bem está sendo executada a tarefa em questão. Estes sinais normalmente são associados a alguma condição dramática - por exemplo, realização de uma subtarefa (recompensa) ou completo fracasso (castigo), e a meta do aprendiz é aperfeiçoar seu comportamento baseado em uma medida de desempenho (função dos reforços recebidos) (Ribeiro, 1999).

O ponto crucial é que, para fazer isto, o aprendiz tem que avaliar as condições (associações entre estados observados e ações escolhidas) que produzem recompensas ou castigos. Em outras palavras, tem que aprender a atribuir crédito a ações e estados passados, através de uma correta estimação dos custos associados a estes eventos. É utilizando essa metodologia que as diversas técnicas vêm sendo desenvolvidas (Ribeiro, 1999).

As técnicas de AR têm atraído muita atenção no contexto de sistemas robóticos. As razões frequentemente citadas para tal atração são: a existência de fortes garantias teóricas sobre a convergência (Szepesvári e Littman, 1996), elas são fáceis de usar, e não dependem de um modelo do ambiente. Além disso, elas também têm sido aplicados com sucesso para resolver uma grande variedade de problemas de planejamento e controle (Celiberto et al., 2008).

Porém, os problemas enfrentados por esse método surgem quando o ambiente é muito grande, consequentemente possui muitos estados (Kaelbling et al., 1996).

O Aprendizado por Reforço Relacional (ARR) é uma proposta recente que vem sendo utilizada para resolver esse problema. Ele possui as mesmas caracteristicas do AR, bem como utilizam algoritmos dessa técnica. A diferença está no método utilizado para armazenar as informações e, principalmente, na representação do ambiente e das ações. A forma de armazenar está relacionada a estrutura e ao mecanismo de generalização, pois é através desse mecanismo que os dados são

acessados e que podem proporcionar uma melhor generalização da informação. Adicionalmente, a utilização de uma representação relacional de primeira ordem (Otterlo, 2005) pode prover um modo econômico para representar o conhecimento necessário a grandes espaços de busca e decisão (Driessens, 2004; Policastro, 2008).

O ARR possui todas as vantagens desejavéis para um mecanismo de aprendizado para robôs sociáveis. Robôs sociáveis são agentes que fazem parte de um grupo heterogêneo, uma sociedade de robôs ou humanos. Estes robôs não devem ser confundidos com sistemas multi-robôs, nos quais diversos robôs devem interagir para a execução de tarefas específicas (Policastro, 2008).

Eles devem poder reconhecer outros robôs e seres humanos e se engajar em interações sociais. Robôs sociáveis devem ser capazes de interagir, de entender, de se comunicar e de se relacionar com seres humanos de uma maneira natural (Breazeal, 2002). Adicionalmente, esses robôs devem ser capazes de aprender a partir das interações com os seres humanos, adquirindo novos conhecimentos e adaptando seus comportamentos em resposta aos estímulos do ambiente (Dautenhahn, 1995; Dautenhahn e Billard, 1999).

Existem motivações científicas e práticas para o desenvolvimento de robôs sociáveis (Breazeal, 2002). Pode-se aprender muito sobre a natureza humana com o processo de desenvolvimento de robôs sociáveis (Breazeal, 2002; Webb, 2000). Robôs sociáveis também são importantes em domínios de problema nos quais os robôs devem interagir com os seres humanos para resolver tarefas específicas, ou em domínios nos quais estes robôs possam ser utilizados como *máquina persuasiva* (Robins et al., 2004; Scassellati, 2001; Björne e Balkenius, 2005). Ainda, nos últimos anos, tem aumentado também o interesse na exploração de robôs sociáveis como robôs de entretenimento com comportamentos semelhantes aos animais (Michaud e Caron, 2000; Scheeff, 2000; Kaplan, 2001).

Embora muitos robôs socialmente interativos tenham sido construídos e utilizados com sucesso, ainda existem muitas limitações a serem superadas com o auxílio do desenvolvimento de novas técnicas e a melhoria das já existentes. A maioria dos robôs existentes possuem habilidades limitadas de percepção, de cognição e de comportamento, em comparação com seres humanos. O desafio é o desenvolvimento de robôs que possuam a noção de socialização, que possam desenvolver habilidades sociais e que possam mostrar empatia e entendimento do mundo real. Esses robôs ainda representam um objetivo distante e o alcance deste requer a contribuição de outras áreas do conhecimento, como a Psicologia, a Ciência Cognitiva e a Sociologia (Dautenhahn, 1997; Dautenhahn e Billard, 1999; Scassellati, 2000).

Para interagir com os seres humanos, os robôs precisam *perceber* e *entender* a riqueza do comportamento humano. Portanto, estes robôs devem possuir sistemas de percepções visuais e auditivas que permita a interação em tempo real, integrando as diversas percepções em uma interface multimodal que possibilite o entendimento e a formação de conceitos sobre os objetos e eventos no ambiente. O desafio aqui é o desenvolvimento de interfaces multimodais que permitam a interação do robô em tempo real, direcionando seus recursos computacionais para o processamento dos estímulos mais relevantes do ambiente (Salichs et al., 2006; Gockley et al., 2007).

Nas interações humano-robô, outro grande desafio é o desenvolvimento de mecanismos eficientes que permitam ao robô compartilhar com uma pessoa a atenção sobre um objeto do evento do ambiente, caracterizando uma habilidade denominada *atenção compartilhada*. A atenção compartilhada é um dos grandes desafios a ser solucionado pelos pesquisadores da robótica social (Kaplan e Hafner, 2004). Esta habilidade é considerada a base essencial para o desenvolvimento das habilidades sociais e cognitivas (Deák et al., 2001; Smith e Ulvund, 2003; Nagai et al., 2003; Kanda et al., 2004). Ela foi definida na literatura como a capacidade de utilizar gestos e contato ocular para coordenar a atenção de outras pessoas de forma a compartilhar experiências sobre objetos ou eventos interessantes (Bosa, 2002; Dube et al., 2004; Kaplan e Hafner, 2004), possibilitando o aprendizado do que é importante no ambiente (Deák e Triesch, 2005a).

Como o projeto de robôs sociáveis pode variar muito em termos de estrutura, técnicas empregadas e objetivos sociais, uma vez que estes robôs podem ser desenvolvidos para diversas aplicações, a organização das estruturas e métodos em uma arquitetura robótica composta por componentes reutilizáveis pode reduzir o tempo e esforço requerido para a construção destes robôs. Tal arquitetura precisa possuir estruturas e mecanismos que permitam a interação social apropriada a partir da aprendizagem e das interações com o ambiente, além de mecanismos que permitam ao robô *perceber* e *entender* a riqueza do comportamento humano (Policastro, 2008).

Neste trabalho, foi efetuado um levantamento sobre o AR e o ARR. Foi efetuado um estudo sobre a arquitetura robótica desenvolvido por Policastro (2008). Um novo algoritmo de ARR foi proposto, incorporado á arquitetura robótica e testado na realização de várias tarefas que serão descritas nesta dissertação.

1.2 Objetivos

O principal objetivo desta pesquisa foi de adquirir uma visão abrangente sobre ARR. Para tanto, um levantamento foi realizado, considerando os seguintes objetivos:

- Realização de estudos bibliográficos para contextualização do estado da arte sobre as técnicas de ARR para desenvolvim ento de sistemas de aprendizado que utilizem representação relacional;
- Desenvolvimento de um mecanismo de aprendizado baseado no ARR para ser inserido como módulo de aprendizado de uma arquitetura robótica.
- Análise estatística dos dados obtidos durante todos os testes realizado sobre o mecanismo desenvolvido.

Vale aqui ressaltar que até o momento, segundo o conhecimento deste pesquisador, não existem trabalhos similares, no desenvolvimento de técnicas de ARR na ótica da robótica sociável, sendo desenvolvidos no Brasil.

1.3 Metodologia

O algoritmo proposto foi modelado e avaliado em um simulador do Mundo dos Blocos. Após esta validação, foram realizadas diversas adaptações ao sistema de aprendizado para ser inserido na arquitetura robótica. Com o novo mecanismo de aprendizado a arquitetura foi avaliada sobre um simulador de Interações Sociais. Foi realizado também um comparativo com outras técnicas de aprendizado já avaliadas sobre a arquitetura. Na última etapa, o algoritmo proposto foi adaptado as condições da cabeça robótica e foram realizados testes em ambiente social controlado.

1.4 Organização do trabalho

O restante desta dissertação está organizado da seguinte forma. No Capítulo 2, são apresentados os conceitos elementares relacionados ao AR, incluindo alguns métodos encontrado na literatura. No Capítulo 3, são apresentados os conceitos sobre ARR e uma descrição das principais técnicas que influenciaram no desenvolvimento de um dos algoritmos de ARR. No Capítulo 4, é apresentada a proposta de algoritmo de aprendizado, TG econômico(TGE), que foi desenvolvido para uma arquitetura robótica. No Capítulo 5, são apresentados a arquitetura robótica e o problema da atenção compartilhada, problema este no qual o algoritmo TGE foi testado visando o aprendizado da cabeça robótica . No Capítulo 6, são descritos os ambientes nos quais foram realizados os testes sobre a proposta e também os resultados obtidos da análise comparativa do método proposto com outros existentes na literatura. Finalmente, no Capítulo 7, são apresentadas as principais conclusões juntamente com os trabalhos futuros.

CAPÍTULO

2

Aprendizagem por Reforço

Neste capítulo, será apresentado a definição de aprendizado por reforço e alguns dos principais métodos existentes que utilizam este paradigma de aprendizado. Antes de realizar as considerações finais sobre o Capítulo é apresentado também algumas técnicas para melhorar o desempenho dos algoritmos de AR.

2.1 Introdução

O Aprendizado por Reforço está associado a questão na qual o agente deve aprender comportamentos onde não exista um auxílio externo, utilizando somente as iterações de tentativa e erro em um ambiente dinâmico. Baseado na idéia que, se uma ação é seguida de estados satisfatórios, ou por uma melhoria no estado, então a tendência para produzir esta ação é aumentada, isto é, reforçada (Faria, 2000).

Nesse tipo de aprendizado, não é informado ao agente quais ações devem ser tomadas em determinadas situações, apenas é informado a ele um conjunto de alternativas na qual o agente deve selecionar uma e o desempenho dessa seleção é associado a essa escolha. O acúmulo de experiências acaba por gerar um conjunto de alternativas mais adequadas às expectativas. Assim, sistemas inteligentes podem adquirir conhecimento exclusivamente a partir de sua interação com o ambiente. Esta capacidade é essencial quando não há fonte de conhecimento disponível (inclusive, modelos cognitivos), e.g. exploração espacial ou submarina (Borsato e Figueiredo, 2007).

A aquisição de informações de aprendizado por experiência direta é uma prática que normalmente não está sob o total controle de um agente: ele pode escolher ações, mas não pode determinar as conseqüências destas observações com antecedência para todos estados, pois normalmente

ele não possui um modelo suficientemente preciso do processo no qual são baseados os julgamentos. Por isso, o agente deve estimar o custo esperado através de visitação direta aos estados, dessa forma, ele deve escolher uma ação, receber um resultado e propagá-lo aos estados anteriores, seguindo o procedimento de Diferença Temporal (DT), sendo essa uma das formas para resolver essa questão.

A técnica de DT parte da idéia de propagar o reforço pelo tempo, assim, os estados que foram visitados anteriormente e que conduziram a esta condição, serão associados a uma predição de conseqüências futuras. Isto está baseado em uma suposição importante em processos dinâmicos, chamada Processo de Decisão de Markov (PDM) (Faria, 2000).

Alguns dos problemas conhecidos como sendo PDM foram solucionados com sucesso pela aplicação de algoritmos de AR, entre esses problemas estão: *job-shop scheduling* (Zhang e Dietterich, 1995), *backgammon* (Tesauro, 1995), controle de elevador (Crites e Barto, 1996), manutenção de máquina (Mahadevan et al., 1997), alocação dinâmica de canal (Singh e Bertsekas, 1997), alocação de poltrona aérea (Gosavi, 2004), etc (Dung et al., 2008).

2.2 Modelo Padrão de Aprendizado por Reforço

Mitchell (1997) nos mostra que existe varias maneiras de formular ou modelar um problema de AR. Uma forma bastante conhecida utilizada atualmente é o PDM.

Em um PDM o agente possui um conjunto S de estados distintos do ambiente e um conjunto A de ações que possa desempenhar. A cada espaço de tempo t, o agente observa o estado atual do ambiente s_t, escolhe uma ação a_t e a executa sobre o ambiente. Esse responde a ação informando ao agente por um reforço escalar, $r_t = r(s_t, a_t)$, e pela produção de um novo estado $s_{t+1} = \delta(s_t, a_t)$ (Mitchell, 1997).

A Figura 2.1 nos ajuda a enteder como o AR utiliza o PDM para modelar o seu funcionamento.

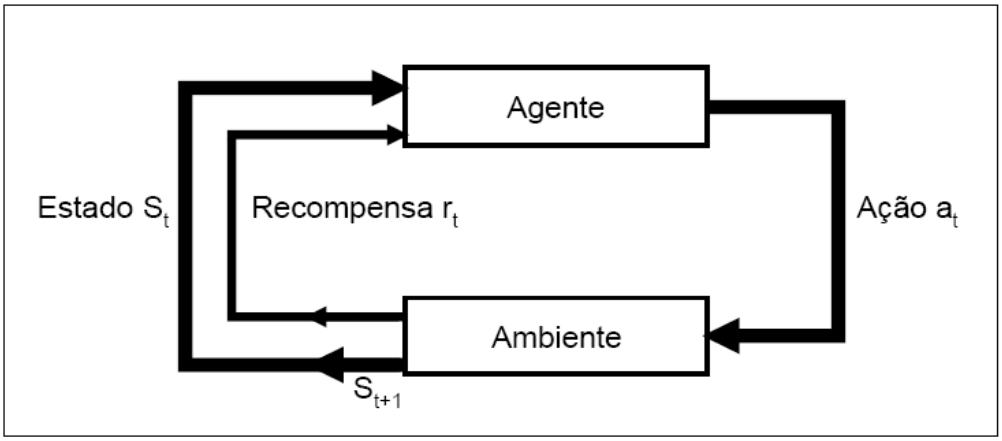

Figura 2.1: Interação entre o agente e o ambiente durante o processo de aprendizado por reforço.

Assim, o trabalho do agente é aprender uma política, π: $S \rightarrow A$, para selecionar sua ação a_t baseada na observação do estado corrente s_t; $\pi(s_t) = a_t$.

Definição 1 *Podemos assim fazer uma definição formal do modelo aprendizado por reforço como:* ***Dado***

- *um conjunto de estados S,*
- *um conjunto de ações A,*
- *uma (possivelmente desconhecido) função de transição $\delta(s, a)$: S x $A \rightarrow S$,*
- *uma desconhecida função de recompensa r: S x $A \rightarrow \mathbb{R}$.*

Encontrar

uma política π^*: $S \rightarrow A$ que maximiza o valor da função $V^\pi(s)$ para todo estado $s_t \in S$. O valor $V^\pi(s)$ é baseado na recompensa recebida iniciando do estado s e seguindo a política π (Mitchell, 1997; Kaelbling et al., 1996; Driessens, 2004).

As funções δ e r, que podem ser determinística ou não-determinística, fazem parte do ambiente e não necessariamente é conhecida pelo agente. Em um PDM, as funções $\delta(s_t, a_t)$ e $r(s_t, a_t)$ dependem apenas do estado e ação corrente, e não de estados e ações anteriores (Mitchell, 1997).

A definição de AR apresentada acima mostra a utilização de uma função de avaliação ou custo, V, além dos fatores já conhecidos.

A função de avaliação estima *quão bom* é para o agente estar em um estado (ou executar uma ação). A noção de "*quão bom*" é definida aqui em termos de recompensas futuras do retorno esperado. As recompensas que o agente espera receber, depende das ações que ele escolherá. Deste modo, funções de custo são definidas de acordo com políticas particulares (Faria, 2000).

Uma das formas de representar a função de avaliação é com a utilização da função de recompensa cumulativa descontada (*Discounted Cumulative Reward*), que pode ser vista como:

$$V^\pi(s_t) \equiv \sum_{i=0}^{\infty} \gamma^i r_{t+1} \tag{2.1}$$

onde o parâmetro γ é chamado fator de desconto, onde $0 \leq \gamma < 1$. Caso $\gamma = 0$ apenas custos imediatos são considerados; a medida que γ aproxima-se do valor 1, os custos futuros são mais relevantes em relação aos custos imediatos. Assim, através do valor de γ é possível balancear as conseqüências das decisões a longo-prazo em relação as decisões a curto-prazo (Mitchell, 1997).

Existem outras definições de recompensa que também são exploradas. Uma possível opção é a recompensa de horizonte finito (*finite horizon reward*), que dado um certo tempo, o agente deve otimizar a recompensa esperada para os próximos h passos, não precisando se preocupar com o que acontecerá depois disso:

$$V^{\pi}(s_t) \equiv \sum_{i=0}^{h} r_{t+i} \tag{2.2}$$

Mitchell (1997) nos mostra outra possibilidade conhecida como recompensa média (*average reward*). Esta considera a recompensa média a cada passo de tempo do agente ao longo de todo o seu período de vida:

$$V^{\pi}(s_t) \equiv \lim_{h \to \infty} \frac{1}{h} \sum_{i=0}^{h} r_{t+i} \tag{2.3}$$

Em resumo, a questão do aprendizado é encontrar uma política ótima, denotada por π^*, i.e., uma política que irá maximizar a função de avaliação conhecida por V^*. Para que isso aconteça existem alguns desafios.

Um dos principais desafios enfrentados pelo aprendizado por reforço é a necessidade de balancear exploração do ambiente e conhecimento adquirido. A cada passo o agente deve fazer uma escolha entre duas opções: uma ação já conhecida com maior recompensa ou uma ação desconhecida e explorar o ambiente, descobrindo talvez uma ação mais vantajosa (Driessens, 2004).

Uma forma de resolver essa questão é utilizando técnicas de seleção randômica.

A técnica mais simples é definir uma porcentagem para que seja realizada a exploração do ambiente. A vantagem é a não interferência desse fator no processo de aprendizagem, já que essa probabilidade é fixa.

Driessens (2004) utiliza a Equação (2.4) para escolha aleatória de uma ação, onde uma ação é escolhida de acordo com a seguinte distribuição de probabilidade:

$$P(a|s) = \frac{\epsilon^{Q(s,a)/T}}{\sum_{a'} \epsilon^{Q(s,a')/T}} \tag{2.4}$$

onde, a temperatura T pode ser reduzida ao longo do tempo. Considerando que uma temperatura elevada provoca alta exploração sobre o ambiente, uma baixa temperatura coloca a ênfase nos valores Q armazenados (Driessens, 2004).

Vários métodos vem sendo propostos para a resolução de problemas PDM e alguns deles são citados abaixo.

2.2.1 Programção dinâmica e método de Monte Carlo

Programação Dinâmica (PD) é um método básico que utiliza uma política estácionaria π^* para problemas estocásticos e requer um modelo completo e explícito do processo a ser controlado, ou seja, deve estar disponível as probabilidades de transição. Por outro lado, aprendizagem autônoma está completamente baseada em experiência interativa e não requer nenhum modelo. Em AR as transições são simuladas em amostras geradas por se ter disponível as probabilidades de transição.

Entretanto, é interessante considerar o caso intermediário, o qual faz uma ponte entre PD e AR. Considere o problema: para uma dada política de ação fixa π, como é possível calcular por simulação - i.e., sem a ajuda de PD - a função de avaliação V^{π}? A solução mais simples é executar muitas trajetórias simuladas para cada estado e calcular a média aritmética dos reforços acumulados obtidos, ou seja, uma *simulação de Monte Carlo* (Faria, 2000).

2.2.2 Diferença Temporal

Os métodos de aprendizagem baseados em DT realizam predições a longo prazo sobre sistemas dinâmicos. Esses métodos são considerados promissores para a resolução de problemas de controle (Kaelbling et al., 1996; Sutton e Barto, 1998). Eles possuem características comuns a outros métodos de aprendizagem por reforço: Monte Carlo e programação dinâmica. Métodos DT são capazes de aprender diretamente por meio da experiência, sem necessitar de um modelo do ambiente, como o métodos Monte Carlo. Métodos DT também são capazes de atualizar suas avaliações utilizando estimativas anteriormente aprendidas, sem esperar por um resultado ao final de uma sequência de ações realizadas, como na programação dinâmica (Sutton e Barto, 1998). Os métodos DT esperam somente uma transição de estado para atualizar $V(s_t)$:

$$\Delta V(s_t) = \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right] \tag{2.5}$$

O ajuste da avaliação do estado s_t é baseado no retorno imediato obtido r_{t+1} e na estimativa do próximo estado $V(s_{t+1})$. Deste modo, o valor da avaliação para os métodos de diferença temporal é $[r_{t+1} + \gamma V(s_{t+1})]$, que constitui uma previsão para o objetivo da aprendizagem desta estratégia (Sutton e Barto, 1998).

Segundo Sutton e Barto (1998), existem dois tipos de freqüência para atualização das estimativas da função avaliação, caracterizando os sistemas como problemas de um ou de múltiplos passos, sendo que para problemas reais os sistemas de múltiplos passos predominam. Em sistema de um passo, toda a informação sobre a predição da avaliação realizada é obtida a cada passo de tempo. Em sistema de múltiplos passos, o agente pode acessar somente a informação parcial sobre a avaliação desejada, sendo sua totalidade revelada somente após vários instantes de tempo. Dessa forma, os métodos de um passo envolvem a utilização do valor de avaliação do estado (ou do par estado-ação) considerando um único sucessor. Já os métodos de múltiplos passos se baseiam na distribuição completa dos valores de avaliação de todos os possíveis estados sucessores. A seguir é apresentado os principais algoritmos de aprendizado por diferença temporal.

2.3 Algoritmos

O AR está dividido em: “métodos independentes de modelo” e “métodos baseados em modelos”. Um modelo consiste em se ter conhecimento prévio da função de transição e da função de

recompensa. Os métodos independentes de modelo (*model-free*) aprendem um controlador sem ter um modelo explícito dos efeitos das ações. Já os métodos baseados em modelos (*model-based*) aprendem e usam um modelo de ações enquanto simultaneamente aprendem um controle ótimo (Faria, 2000).

Os algoritmos TD(λ) (Sutton e Barto, 1998), *Q-learning* (Watkins, 1989), SARSA (Rummery e Niranjan, 1994) e R-$learning$ (Schwartz, 1993) são exemplos de métodos indepedentes de modelo, enquanto DYNA (Singh e Sutton, 1996) e H-$learning$ (Tadepalli e Ok, 1994) são métodos baseados em modelo.

2.3.1 Algoritmo TD(λ)

O algoritmo geral de diferença temporal TD(λ) utiliza elementos da programação dinâmica, dos métodos de Monte Carlo e métodos de diferença temporal utilizando um mecanismo denominado traço de elegibilidade. Este é um registro temporário sobre a ocorrência de um determinado evento, como a visita a um estado ou a seleção de uma ação. O traço atribui os parâmetros de memória associados com os eventos ocorridos, de forma a classificá-los como elegíveis para mudanças no aprendizado em andamento. Quando o processo incorre em erro, somente os estados ou ações elegíveis recebem crédito ou culpa pelo erro. Desta forma, os traços de elegibilidade ajudam a integrar os eventos e a informação de treinamento relacionadas com os mesmos. Em sua forma mais simples, é atribuído um traço a cada estado do ambiente. O traço de eligibilidade de um estado s no tempo t é representado por $e_t(s)$. Em cada passo s da trajetória, os traços para todos os estados visitados decaem de acordo com uma taxa representada por $\gamma\lambda$ e o traço correspondente ao estado visitado s_t é incrementado de 1 (Sutton e Barto, 1998):

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & se \quad s \neq s_t \\ \\ \gamma\lambda e_{t-1}(s) + 1 & se \quad s = s_t \end{cases} \tag{2.6}$$

na qual γ é o parâmetro de desconto usualmente utilizado e λ é o parâmetro de decaimento do traço de elegibilidade, $0 \leq \lambda \leq 1$.

O traço de elegibilidade da Equação (2.6) é denominado traço cumulativo, uma vez que ele é incrementado a cada vez que o estado é visitado e decai gradualmente quando o contrário ocorre. A qualquer instante de tempo, o conjunto de traços registra quão recentemente um estado foi visitado. O traço indica, então, o quanto os estados são importantes para que o aprendizado ocorra (Sutton e Barto, 1998).

O algoritmo $TD(\lambda)$ é obtido pela combinação do traço de elegibilidade dado pela Equação (2.6) com o erro de diferença temporal utilizado no algoritmo TD, dado pela Equação (2.7) (Sutton e Barto, 1998):

$$\Delta V_t(s) = \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]e_t(s), \; para \; \forall \; s \; \in \; S \tag{2.7}$$

na qual $e_t(s)$ é o traço de elegibilidade de um estado s no tempo t.

O erro da diferença temporal para a predição do valor do estado s em t é representado por δ_t

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{2.8}$$

Este erro sofre variação a cada passo de tempo. Estados diferentes recebem crédito por meio do erro DT a cada instante, de acordo com o tamanho de seu traço de elegibilidade. Assim, as atualizações são proporcionais ao valor do traço de todos os estados visitados:

$$\Delta V_t(s) = \alpha \delta_t e_t(s),\ para \, \forall \, s \ \in \ S \tag{2.9}$$

O algoritmo $TD(\lambda)$ é uma família de algoritmos parametrizada pelo fator λ de decaimento do traço. Os dois algoritmos extremos desta família são denominados TD(0) e TD(1). Se $\lambda = 0$, então o algoritmo considera somente o próximo estado para atualizar a avaliação e todos os traços de elegibilidade (Equação 2.6) recebem valor zero no tempo t, exceto o traço correspondente ao estado visitado. Se $\lambda = 1$, o traço associado a cada estado s decai de acordo com γ, enquanto o traço correspondente ao estado s_t recebe este mesmo decaimento e é incrementado de 1. A forma geral desse método é apresentado no Algoritmo 1 (Sutton e Barto, 1998).

Algoritmo 1 Algoritmo TD(λ)

Inicie $V(s)$ arbitrariamente e $e(s) = 0$, para todo $s \in S$
para cada tentativa **faça**
  Inicie s
  Obtenha o estado do ambiente s_t
  para cada passo em uma tentativa **faça**
    Selecione uma ação a_t
    Execute a ação a_t
    Obtenha o retorno r_{t+1}
    Obtenha o estado s_{t+1}
    Calcule o traço e elegibilidade do estado s_t {Equações 4.3 e 4.5}
    Calcule o traço de elegibilidade dos demais estados s {Equação 4.3}
    Atualize o valor da avaliação $V(s_t)$ para todos os estados s {Equação 4.6}
    Faça $s_t = s_t + 1$
  fim para
fim para

A avaliação $V(s_t)$ pode ser atualizada a cada passo de tempo t (sistema "online"), ou somente ao final da trajetória ("offline" ou horizonte finito) (Sutton e Barto, 1998).

2.3.2 Q-Learning

O Algoritmo *Q-learning*, uma técnica proposta por Watkins (1989), é um método iterativo para o aprendizado de uma política de ação em agentes autônomos (Faria, 2000). E é um dos algoritmos

mais utilizados pertencente ao paradigma de aprendizado por reforço, devido a sua simplicidade de implementação.

O *Q-learning* (Watkins, 1989) é conhecido por ser um algoritmo de diferença temporal que computa o Valor Q (Q) para uma política ótima associada com cada par (estado, ação), sendo este construido por uma exploração arbitrária do ambiente.

A base do algoritmo de Watkins (1989) está na estreita ligação entre a função avaliação V e o Q. A diferença entre ambos é que a primeira tem como argumento um estado s e mostra o custo total seguindo a política π desde o primeiro instante, enquanto o segundo tem como argumento uma ação $a \in A$. Assim todas as ações possíveis para o estado s podem ser classificadas de acordo com o custo total que sua escolha acarreta.

Dessa forma, para selecionar a melhor ação no estado s, a ação a deve maximizar a soma das recompensas imediatas mais o valor V^* do estado sucessor, descontado por γ. A política é definida como:

$$\pi = \arg\max_a [r(s,a) + \gamma V^\pi(\delta(s,a))] \tag{2.10}$$

O valor de Q é calculado pela soma da recompensa recebida imediatamente a execução da ação a sobre o estado s e o valor V^π, descontado de γ, por seguir a política π. Neste caso, o Q de um par (estado, ação) está ligado a política π relacionada à função valor definida como:

$$Q(s,a) \equiv r(s,a) + \gamma V^\pi(\delta(s,a)) \tag{2.11}$$

Pela equação (2.11), podemos reescrever a equação (2.10) em relação ao Q:

$$\pi = \arg\max_a Q(s,a) \tag{2.12}$$

Essa equação mostra que se o agente aprende a função Q em vez da função V^*, ele será capaz de selecionar a melhor ação sem conhecer as funções r e δ (Mitchell, 1997).

Isto é feito pela Estimativa ou hipótese do valor Q ($\hat{Q}$), da atual função Q. Está hipótese é representada por uma tabela (matriz), que armazena cada valor $\hat{Q}$ para cada par estado ação (Mitchell, 1997).

A figura 2.2 ilustra a operação do algoritmo *Q-learning*, mostrando a simples ação tomada pelo agente, bem como o refinamento $\hat{Q}$. Ele aplica a regra de treinamento

$$Q(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a') \tag{2.13}$$

para refinar as estimativas do $\hat{Q}$ para a transição do estado-ação executada. No exemplo, o agente move uma célula à direita no tabuleiro e recebe uma recompensa imediata de zero para esta transição. De acordo com as regras de treinamento, a nova estimativa $\hat{Q}$ para esta transição é a soma

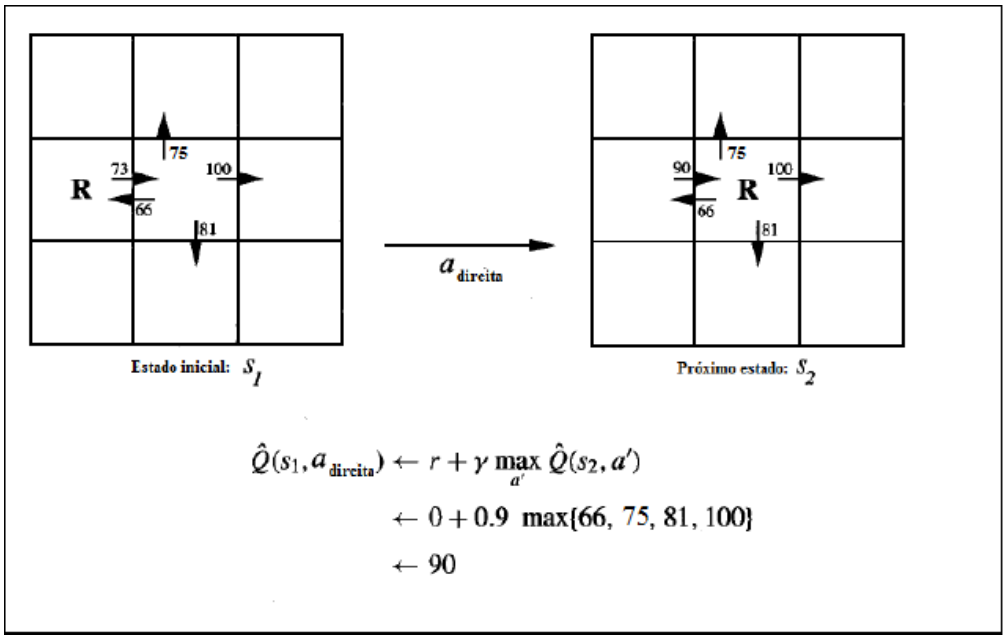

Figura 2.2: Atualização do $\hat{Q}$ após a execução de uma ação (Mitchell, 1997).

da recompensa recebida (zero) e o mais alto valor $\hat{Q}$ associado com o estado de resposta (100), descontado por γ (0.9) (Mitchell, 1997).

O algoritmo 2 descreve o funcionamento do algoritmo *Q-learning* para processos de decisão Markoviano determinístico. Esse utiliza a equação (2.13), conhecida como equação de Bellman.

Algoritmo 2 Q-Learning

para todo $s \in S, a \in A$ **faça**
  inicializa tabela $Q(s, a)$
fim para
gera um estado inicial
repita
  seleciona uma ação a e executa
  recebe recompensa $r = r(s, a)$
  observa o novo estado s'
  atualiza a tabela $Q(s, a)$ como segue:
  $Q(s, a) \leftarrow r + \gamma max_{a'} Q(s', a')$
  $s \leftarrow s'$
até que s seja terminal

Para casos onde o ambiente é não-determinístico, outra equação de Bellman é utilizada:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha_t[r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)] \tag{2.14}$$

na qual a taxa de aprendizagem (α) varia entre 0 e 1 ($0 < \alpha < 1$) e γ é o parâmetro de desconto ($0 < \gamma < 1$) (Mitchell, 1997; Policastro, 2008).

O AR, e *Q-learning* em particular, encontram dois grandes problemas quando encontram um grande espaço de estados. Primeiro, o aprendizado com a função de avaliação Q na forma tabular pode ser inviável por causa da quantidade excessiva de memória necessária para armazenar a tabela, e porque a função de avaliação Q apenas converge após o fato que cada estado tenha sido visitado múltiplas vezes. Segundo, as recompensas no espaço do estado podem ser escassas com exploração aleatória onde as descobertas são extremamente lentas (Driessens e Džeroski, 2004).

Um exemplo

Para melhor exemplificar o funcionamento do algoritmo *Q-learning*, o grupo de pesquisa possui um programa para o problema de locomoção de agente empregando o *Q-learning*. O programa foi desenvolvido por Faria (2000) e a sua interface é apresentada na Figura 2.3.

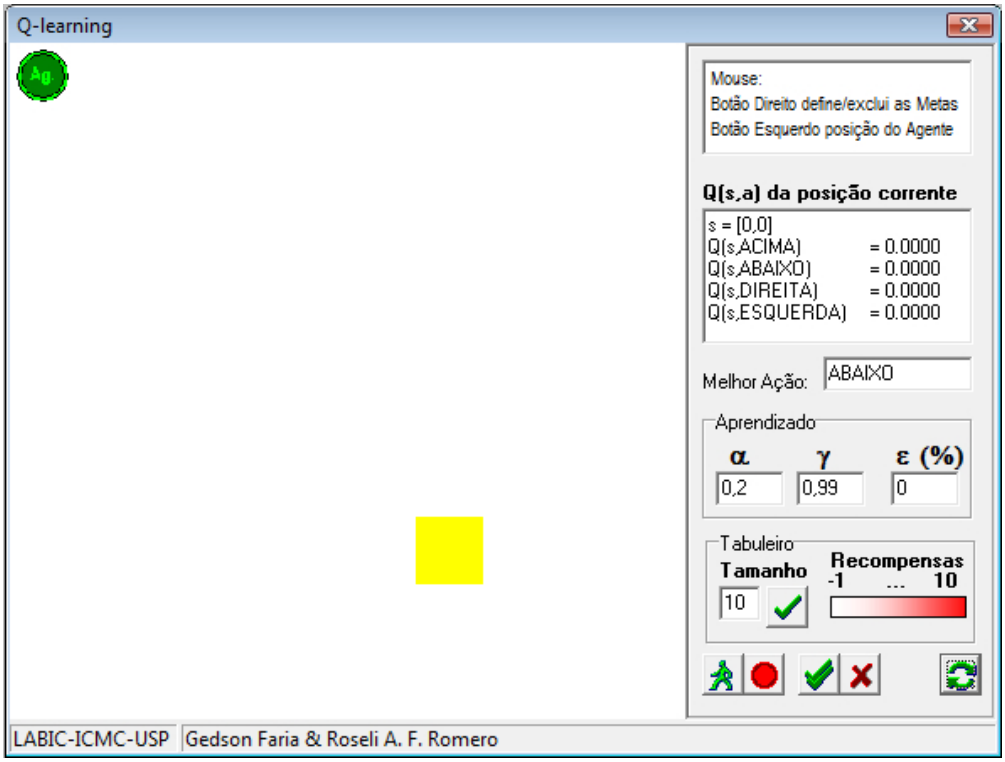

Figura 2.3: Interface do programa *Q-learning* para o problema de locomoção.

O programa pode ser dividido em duas partes, a esquerda está a configuração que podem ser feitas e algumas informações e a direita está o ambiente na qual acontecerá a interação.

A explicação do menu de configurações é feita do topo em direção ao ponto mais baixo do programa. A primeira caixa contém informa sobre como definir a posição do agente e dos objetivos (é posivél colocar mais de um). A segunda caixa mostra a posição do agente no ambiente e os valores Q para cada possível ação. Na terceira caixa mostra a melhor ação a ser tomada pelo agente segundo o valor Q de cada possível ação. Todas caixa mostrada anteriormente contêm, somente, informações ao usuário.

Na caixa que contém como título de aprendizado contém três fatores (α,γ e $E(\%)$) que podem ser ajustado, sendo referidos a taxa de aprendizado, o parâmetro de desconto e o fator de exploração do ambiente, respectivamente. A caixa com o título Tabuleiro contêm o tamanho do ambiente, sempre configurado de forma quadrangular que na Figura tem 10 linhas e 10 colunas e uma faixa de cores que está associada as recompensas.

No ponto mais baixo do programa existem 5 botões. O primeiro está identificado por um boneco e mostra cada passo tomado pelo agente e deve ser acionado para cada um. O segundo é um circulo vermelho e ao ser acionado o agente parte a procura da meta. A seguir, um botão com dois "v's" sobrepostos está referenciando ao aprendizado contínuo e só termina quando o usúario pressiona o botão "X" que está em vermelho. O último botão limpa as configurações feitas no ambiente.

O ambiente é composto por um agente, que está na forma circular e um identificador ("Ag."). Uma ou mais metas, que está representada por um quadrado na cor amarela, a ser atingida pelo agente que é configurada pelo usuário.

A medida que o agente interage com o ambiente a procura da meta, ele vai definindo a política a ser seguida e fica representada no ambiente pela difente tonalidade de cores. A Figura 2.4 nos mostra como fica a política traçada pelo agente após algumas interações com o ambiente.

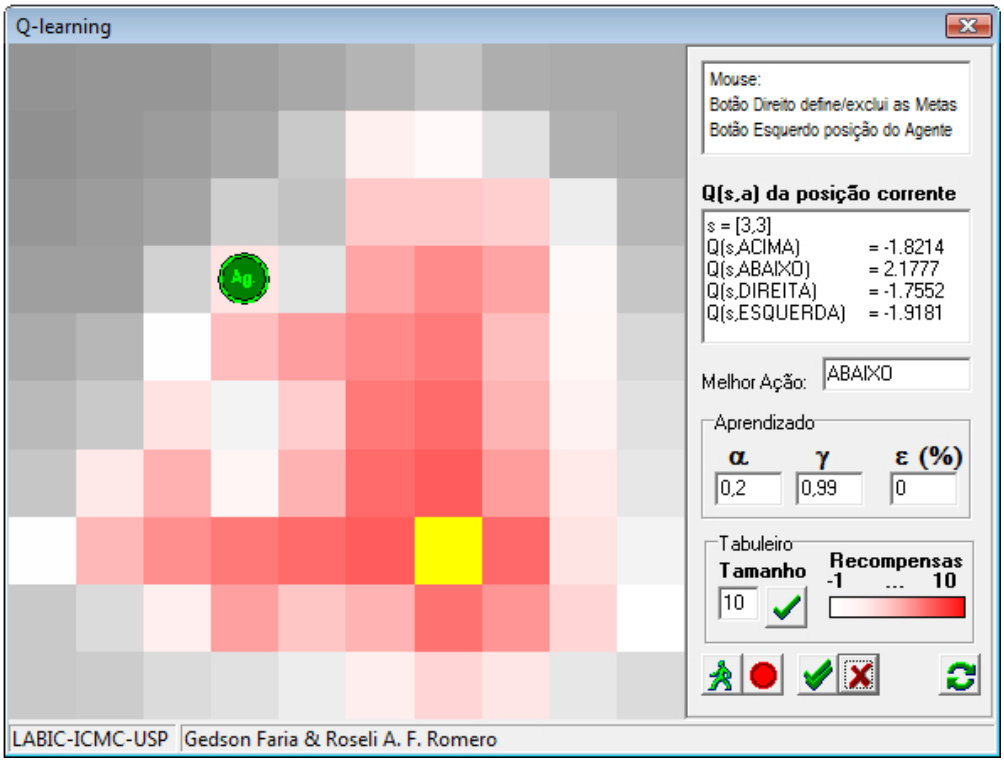

Figura 2.4: Interface do programa *Q-learning* após algumas etapas de interação do agente com o ambiente.

2.3.3 Algoritmo SARSA

O algoritmo SARSA (*State-Action-Reward-State-Action*) é considerado uma variação do algoritmo *Q-Learning*, que foi introduzido na nota técnica "on-line utilizando Q-Learning Connectionist Systems" por Rummery e Niranjan (1994) onde o nome alternativo SARSA só foi mencionado como uma nota de rodapé.

O algoritmo de aprendizagem SARSA é um algoritmo de política em que aprende baseado na quíntupla $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$. Dado o par estado-ação (s_t, a_t), SARSA simula a ação no estado em s_t para obter a recompensa r_t e de transição para o estado s_{t+1}. O algoritmo então usa a sua política ótima corrente para a produção de sua próxima ação a_{t+1} (Greenwald, 2005). SARSA atualiza os custos das ações realizando em cada passo a equação (2.15).

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s, a) + \alpha_t[r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)] \tag{2.15}$$

Naturalmente, caso a ação escolhida seja $\arg\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$, esse algoritmo será equivalente ao $Q - learning$ padrão. Por eliminar o uso do operador $\max$ sobre as ações, este método é mais rápido que o Q-learning para situações onde o conjunto de ações tenham alta cardinalidade. A descrição do algoritmo SARSA é mostrada no Algoritmo 3 (Faria, 2000).

Algoritmo 3 SARSA

inicialize Q(s,a) arbitrariamente
$s \leftarrow$ estado atual
$a \leftarrow$ ação derivada de Q {cada passo do episódio}
repita
  execute a ação a
  receba a recompensa r e o próximo estado s'
  $a \leftarrow$ ação derivada de Q
  $Q(s, a) \leftarrow Q(s, a) + \alpha\left[r + \gamma Q(s', a') - Q(s, a)\right]$
  $s \leftarrow s'$
  $a \leftarrow a'$
até que s seja terminal

Algumas variações são obtidas combinando o $TD(\lambda)$ com algoritmos *Q-learning* e SARSA. Tais variações são conhecidas respectivamente como $Q(\lambda)$ e $SARSA(\lambda)$ e suas descrições podem ser encontradas em Sutton e Barto (1998).

2.3.4 *R-learning*

O *R-learning* é uma técnica proposta por Schwartz (1993) que maximiza a recompensa média de cada passo. O $R - learning$ é um método de controle independente de política para versões avançadas de aprendizado por reforço nas quais não se utiliazam descontos e nem dividem as experiências em episódios distintos com retornos finitos (Faria, 2000).

O algoritmo *Q-learning* não maximiza a recompensa média, mas descontos acumulados de recompensa, por isso $R - learning$ deve fornecer de fato resultados melhores que o *Q-learning*. Em *R-learning* utiliza-se o modelo de recompensa média que é definido pela equação 2.3. (Faria, 2000).

Será assumido que este é um processo $ergodic$ (a probabilidade de se sair de qualquer estado e ir para outro sob uma política deve ser diferente de zero) e deste modo ρ^{π}, função de avaliação para uma política π, é independente do estado inicial (Faria, 2000).

O custo da ação $R^{\pi}(s, a)$ representa o valor ajustado da média ao realizar uma ação a no estado s, seguindo uma política π subsequentemente, ou seja:

$$R^{\pi}(s_t, a_t) = r(s_t, a_t) - \rho_t^{\pi} + \sum_{s_{t+1} \in S} P(s_{t+1}|s_t, a_t) V^{\pi}(s_{t+1}), \tag{2.16}$$

onde $V^{\pi}(s_{t+1}) = \max_{a \in A} R^{\pi}(s_{t+1}, a)$

Os valores de R são ajustados a cada ação baseados na regra:

$$R_{t+1}(s_t, a_t) \leftarrow R_t(s_t, a_t) + \alpha_t[r(s_t, a_t) - \rho_t + \max_a R_t(s_{t+1}, a) - R_t(s_t, a_t)], \tag{2.17}$$

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

$$\rho_{t+1} \leftarrow \rho_t + \beta_t[r(s_t, a_t) - + \max_a R_t(s_{t+1}, a) - \max_a R_t(s_t, a) - \rho_t] \tag{2.18}$$

O ponto chave é que ρ somente é atualizado quando uma ação não aleatória foi tomada, ou seja, $\max_a R_t(s_t, a) = R_t(s_t, a_t)$. O ρ não depende de algum estado particular, ela é uma constante para todo o conjunto de estados. A descrição do algoritmo *R-learning* é apresentada no Algoritmo 4 (Faria, 2000).

Algoritmo 4 $R - learning$

inicialize ρ e $R(s, a)$ arbitrariamente
para sempre **faça**
  $s \leftarrow$estado atual
  $a \leftarrow$ação baseada em uma política de comportamento $\epsilon - gulosa$
  Execute a ação $a \in A(s)$
  receba a recompensa r e o próximo estados'
  $R(s, a) \leftarrow R(s, a) + \alpha[r - \rho + \max_{a'} R(s', a') - R(s, a)]$,
  se $R(s, a) = \max_a R(s, a)$ **então**
    $\rho \leftarrow \rho + \beta[r - \rho + \max_{a'} R(s', a') - \max_a R(s, a)$
  fim se
fim para

2.3.5 Algoritmo DYNA

O termo Dyna (Singh e Sutton, 1996) define uma técnica simples para integrar aprendizado, planejamento e atuação. O agente interage com o ambiente, gerando experiências reais. Estas experiências são utilizadas simultaneamente para construção de um modelo ($\hat{T}e\hat{R}$) e 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 (Kaelbling et al., 1996).

Após cada transição ($s_t, a_t \rightarrow s_{t+1}, r$), 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 (Monteiro e Ribeiro, 2004).

Dyna opera em uma iteração de *loop* com o ambiente. Dado uma tupla $\langle s, a, s', r \rangle$, ele se comporta da sequinte forma:

- Atualiza o modelo, incrementando a estatística da transição do estado s para s' com a ação a e com o recebimento da recompensa r. Os modelos atualizados são $\hat{T}e\hat{R}$.
- Atualiza a política do estado s baseado no modelo atualizado usando a regra:
  $Q(s,a) = \hat{R}(s,a) + \gamma \sum_{s'} \hat{T}(s,a,s') \max_{a'} Q(s',a')$
- Realiza k atualizações adicionais: é escolhido aleatoriamente k pares de estado-ação e os atualiza segundo a mesma equação anterior:
  $Q(s,a) = \hat{R}(s,a) + \gamma \sum_{s'} \hat{T}(s,a,s') \max_{a'} Q(s',a')$
- Escolha uma ação a' a ser realizada no estado s', baseado no valor Q

O algoritmo Dyna requer k vezes a computação do $Q - learning$ para cada exemplo, mas este é tipicamente bem menor que os método baseado em modelo mais simples. Um valor razoável de k pode ser determinado baseado na velocidade da computação e na tomada de ação (Kaelbling et al., 1996).

2.3.6 *H-Learning*

O algoritmo *H-learning* (Tadepalli e Ok, 1994) é um algoritmo que utiliza métodos baseados em modelo e assim como o *R-learning* foi introduzido para otimizar a recompensa média sem a utilização de desconto (Faria, 2000).

Um conjunto de estados é $ergodic$ com respeito a uma política, se a probabilidade de sair de qualquer estado e ir para outro for diferente de zero, sendo que não pode haver transições para fora do conjunto (Faria, 2000).

A seguir, assume-se que as garantias de estretégia de exploração para todo estado pertencente a S é visitado com alguma probabilidade, de forma que o conjunto total de estados é $ergodic$ para toda política durante o treinamento (Faria, 2000).

Sob estas condições, as recompensas finitas iniciais obtidas, indo de um estado s para s', não contribuem em nada à recompensa de média esperada, a longo prazo. Consequentamente, se μ é uma política intermediária usada em treinamento, então $\rho^{\mu}(s) = \rho^{\mu}(s')$. Denota-se apenas por $\rho(\mu)$ e considera-se o problema de encontrar uma política ótima μ^* que maximize $\rho(\mu)$. Este problema é resolvido pelo seguinte teorema e é provado em Bertsekas (1987).

Teorema 1 *Se um escalar ρ e um vetor n-dimencional h satisfazem a relação recorrente*

$$h(i) = \max_{u \in U(o)} \{r(i,u) + \sum_{j=1}^{n} p_{ij}(u)h(j)\} - p,\ i = 1, ..., n \tag{2.19}$$

então ρ é uma recompensa média $\rho(\mu)$ e μ^* atinge o máximo para cada estado i (Faria, 2000).

O $H - learning$ estima as probabilidades $P(s'|s,a)$ e os reforços $R(s,a)$ por contagem direta e atualiza os valores de h utilizando a equação 2.19, como podemos observar no algoritmo 5 a descrição do $H - learning$ (Faria, 2000).

O *H-learning* faz escolhas aleatórias de ações com uma probabilidade fixa, ou seja, também utiliza a política de escolha de ações e-gulosa. O método utilizado para atualizar a recompensa média ρ segue a idéia de Schwartz (1993). Isso pode ser percebido comparando-se a Equação 2.18, que faz a atualização de ρ para o algoritmo *R-learning* utilizando-se $\beta = 1/T$, com a atualização de ρ feita pelo $H - learning$ (Faria, 2000).

O valor $h(s)$ representa a recompensa esperada estando no estado s, sendo equivalente a $\max_{a \in A(s)} R(s,a)$ do algoritmo $R - learning$.

2.4 Generalização

Durante toda discussão realizada sobre os métodos de AR, foi assumido a possibilidade de enumerar os estados e ações e armazená-los em tabelas. Para ambiente muito pequeno isso é possível, mas se torna impraticável pela quantidade de memória necessaria. Isto também torna ineficiente o uso de experiência. Em um ambiente considerado grande e plano espera-se que estados similares tenham ações ótima similares. Portanto, deve existir alguma representação mais compacta que a tabela. O problema de aprendizado em grandes ambientes está relacionado a

Algoritmo 5 H-$learning$

Considere $N(s,a)$ como o número de vezes que a ação a foi executada no estado s, e $N(s,a,s')$ como o número de vezes que $N(s,a)$ resultou no estado s'.$P(s'|s,a)$ é a função de probabilidade de atingir o estado s' estando no estado s e executando a ação a. $r(s,a)$ é a recompensa esperada e r' é a recompensa imediata. $h(s)$ representa a recompensa gerada estando no estado s, $A_{best}(s)$ é o conjunto de ações ótima no estado s e T é o número total de passos que uma ação aparentemente ótima foi executada.
Inicialize as matrizes $P(s'|s,a)$,$r(s,a)$,$h(s)$ e o escalar ρ com $0's$
$A_{best}(s) \leftarrow A(s)$
$T \leftarrow 0$
$s \leftarrow$ valor aleatório do estado corrente
repita
  se a estratégia de exploração sugere uma ação aleatória para s **então**
    $a \leftarrow$ ação aleatória
  senão
    $a \leftarrow A_{best}(s)$
  fim se
  execute a ação a
  receba a recompensa imediata r' e o estado resultante s'
  $N(s,a) \leftarrow N(s,a)+1$
  $N(s,a,s') \leftarrow N(s,a,s')+1$
  para todo $s' \in S$ **faça**
    $P(s'|s,a) \leftarrow N(s,a,s')/N(s,a)$
  fim para
  $r(s,a) \leftarrow r(s,a)+(r'-r(s,a))/N(s,a)$
  se $a \in A_{best}(s)$ **então**
    $T \leftarrow T+1$
    $\rho \leftarrow \rho+(r'-h(s)+h(s')-\rho)/T$
  fim se
  para todo $a \in A(s)$ **faça**
    $H(s,a) \leftarrow r(s,a)+\sum_{s' \in S} P(s'|s,a)h(s')$
  fim para
  $A_{best}(s) \leftarrow \arg\max_{a \in A(s)} H(s,a)$
  $h(s) \leftarrow H(s,a)-\rho$, onde $a \in A_{best}(s)$
  $s \leftarrow s'$
até que atinja convergência ou MAX-STEPS vezes

técnica de generalização, que permite compactar as informações do aprendizado armazenado e transferir o conhecimento entre estados e ações similares (Kaelbling et al., 1996).

Kaelbling et al. (1996) informa a existência de uma larga literatura de técnicas de generalização de indução para aprendizado que pode ser aplicado ao AR, mas sendo necessários certos ajustes de acordo com a especificidade do problema. Generalização sobre a entrada, generalização sobre a saída e métodos de hierarquia são abordados.

A generalização sobre a entrada busca técnicas para generalizar ações ou o aperfeiçoamento da função de avaliação do estado corrente do agente. O primeiro grupo de técnicas é especializado no caso onde a recompensa não é atrasada sendo chamada recompensa imediata (Kaelbling et al., 1996).

A recompensa imediata é quando a ação do agente não influencia na transição de estados, e o problema se transforma na escolha de ações para maximizar a recompensa imediata em função do estado corrente do agente. Os algoritmos dessa técnica estão ligado aos problemas de aprendizado por recompensas imediatas booleanas onde o estado é um vetor de valores e a ação é um vetor booleano. Eles também podem ser generalizados para recompensas de valores reais através do método de recompensa comparada (Sutton e Barto, 1998). Alguns algoritmos desta técnica podem ser encontrados em Kaelbling et al. (1996).

No reforçamento atrasado, o segundo grupo de técnica, a função de aproximação é usada para representar a função de avaliação por um mapeamento da descrição de um estado por um valor. Entre eles, os modelos de resolução adaptativa (*adaptive resolution models*) busca particionar o ambiente em regiões que podem ser consideradas similares para um propósito de aprendizado e gerando ações para tal regiões. As árvores de decisão (*decision trees*) são utilizadas em ambientes que podem ser caracterizados por um conjunto booleano ou discreto, sendo possível armazenar os valores Q nessa estrutura. E o algoritmo VRDP (em inglês, Variable Resolution Dynamic Programming) capacita a programação dinâmica convencional a trabalhar com variáveis reais de estados (Kaelbling et al., 1996).

A generalização sobre ações se torna importante quando as ações se tornam numerosas e sendo descritas de forma combinatorial. Alguns métodos utilizando redes neurais são apresentados em Kaelbling et al. (1996).

Aprendizado hierárquico é geralmente estruturado pela escolha de comportamento (em inglês, *gated behaviors*), como mostra a Figura 2.5. Há uma coleção de comportamento que mapeia os estados do ambiente em ações e a função de escolha decide, baseada no estado do ambiente, qual ação deve ser selecionada (Kaelbling et al., 1996).

Alguns algoritmos desta técnica podem ser encontrados em Kaelbling et al. (1996).

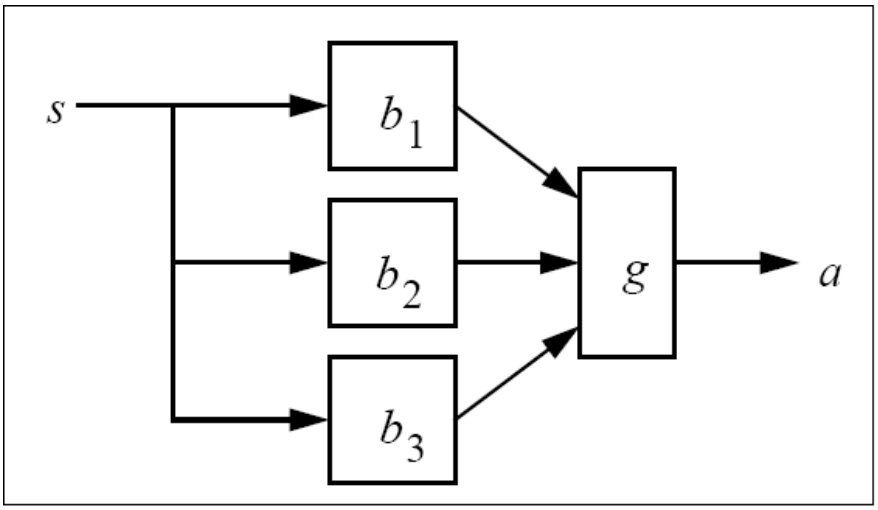

Figura 2.5: A estrutura das escolhas de comportamentos.

2.5 Considerações Finais

O Aprendizado por Reforço é bem aplicado as situações em que os exemplos para aprendizado de um agente não são confiavéis ou mesmo não são disponíveis, e o agente possui um conjunto de ações a ser escolhida, considerando o estado atual do ambiente, a fim de atingir um objetivo maximizando o reforço recebido.

O Capítulo foi introduzido com a definição formal de AR como forma de introduzir os conceitos envolvidos no método de aprendizado. Alguns dos principais métodos foram apresentados mostrando a plausibilidida do desenvolvimento desse modelo de aprendizado. Apesar da aplicavéis, existem técnicas que são utilizadas para melhorar o desempenho desses métodos. No próximo Capítulo são introduzidos os conceitos de aprendizado por reforço relacional.

CAPÍTULO

3

Aprendizagem por Reforço Relacional

Esse capítulo tem como finalidade mostrar os conceitos fundamentais do ARR e o seu funcionamento. Será discutido sobre a importância de uma boa representação do ambiente e como é fundamentada a representação de primeira ordem. Em seguida, será apresentada uma definição formal de Aprendizado por Reforço Relacional. Posteriormente, um embasamento teórico das técnicas que contribuiram para a construção do algoritmo TG e o seu funcionamento serão apresentados. E finalmente, as considerações sobre o Capítulo.

3.1 Introdução

O Aprendizado por Reforço Relacional (ARR) foi projetado buscando resolver o problema em domínios onde os objetos possuem relações entre si. Esses domínios são, na maioria das vezes, muito extensos e as técnicas de AR não são satisfatórias (Driessens, 2004).

O ARR é uma técnica de aprendizado que combina aprendizado por reforço com representação relacional do ambiente (Dabney e McGovern, 2007), ou seja, os estados e as ações são representados através de predicados.

Na construção de um sistema ARR, adotando-se o *Q-learning* como método de aprendizado por reforço, o mecanismo de representação da função Q deve ser analisado, pois esse mecanismo deve ser apropriado para trabalhar com a representação relacional (Driessens, 2004). Na literatura, são três os principais métodos de representação da função Q comumente utilizados. Cada um deles é baseado em diferentes técnicas: representação através de árvores de regressão, representação baseado em exemplo e representação baseada em processos gaussianos com kernel.

Inicialmente, será abordadado, a importância de uma boa representação, demonstrando que a relacional é interessante em vários aspectos. Posteriormente, a definição de ARR e a base do algoritmo. E finalmente, serão abordados técnicas que são base precursora do mecanismo regressivo que será focalizado neste trabalho.

3.2 Representação do conhecimento

Representação do conhecimento é um elemento chave da Inteligência Artificial (IA) (Otterlo, 2005). Poole et al. (1998) definou conhecimento como sendo a informação sobre algum domínio ou área específica, ou sobre como se fazer alguma coisa. Os seres humanos requerem e usam muito conhecimento para realizar até mesmo as mais simples tarefas de senso comum ou tarefas como reconhecimento de faces em uma pintura, diagnóstico médico, entendimento da linguagem natural ou uma argumentação jurídica. Os computadores tem um bom desempenho em tarefas na qual não necessita de muito conhecimento, como cálculos aritméticos e diferenciação simbólica, mas ainda não conseguem executar bem as tarefas citadas que são facilmente realizados pelos seres humanos.

Freqüentemente, a forma adotada para representar o ambiente pode definir muito bem um domínio, mas pode ser inapropriado para outros (Poole et al., 1998).

3.2.1 Importância da representação

Para facilitar o entendimento, vamos adotar a utilização do algoritmo *Q-learning* como método de aprendizado para os exemplos de representação.

Podemos usar como exemplo a locomoção de um robô em um determinado ambiente, como o da Figura 2.2, no qual o agente a partir de uma posição inicial precisa chegar a uma outra posição. Podemos representa-lo com a utilização de uma matriz LxC (onde L é o número de linhas e C o número de colunas), indicando a posição do agente no ambiente. Para cada posição é utilizado um vetor, que pode ter 4 ou 8 posições, indicando as possíveis ações do agente no estado e armazenando o valor de Q associado a uma ação a ser tomada, como na Figura 2.2.

A Figura 3.1 nos mostra essa representação, onde o agente está representado pela letra A, o seu estado inicial por Y e o seu objetivo por X. Para cada posição da matriz existe um vetor que armazena o valor de Q para cada uma das opções (N,S,E,W,NE,NW,SE,SW), segundo a configuração do ambiente.

Existem situações onde a representação utilizada não é suficiente para definir estados e ações. Um exemplo disso a tentativa de representar o ambiente do mundo dos blocos, como o da Figura 3.2, empregando a estrutura da Figura 3.1.

O mundo dos blocos consiste de um número constante de blocos e uma superfície capaz de suportar todos os blocos. Os blocos podem estar sobre a superfície ou pode estar, apenas, sobre um outro bloco. Isto também implica que existirá, pelo menos, um bloco sobre a superfície.

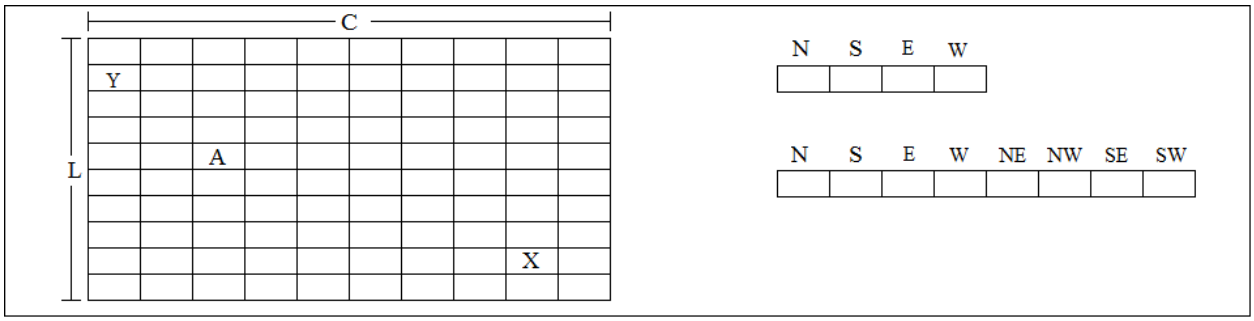

Figura 3.1: Representação do ambiente para o problema da locomoção de um agente.

As ações no mundo dos blocos consistem em movimentar um bloco que esteja no topo de uma torre ou na superfície (Nilsson, 1980; Langley, 1995; Slaney e Thiébaux, 2001). Uma explicação mais detadalha do problema do Mundo dos Blocos é realizada no Capítulo 6.

A Figura 3.2 mostra o mundo dos blocos, com 5 blocos e a seta pontilhada indica a ação que será tomada.

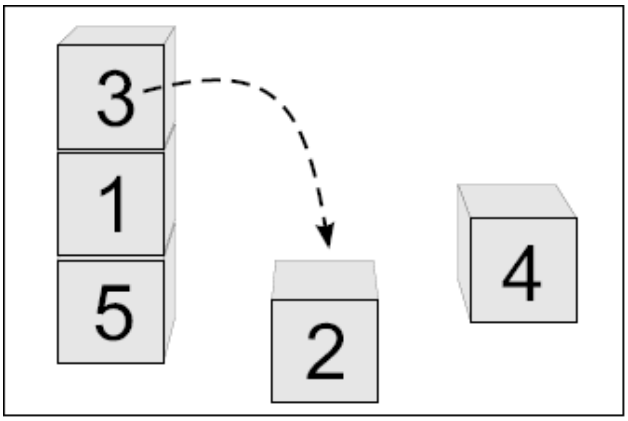

Figura 3.2: O mundo dos blocos que possui 5 blocos

Podemos adotar uma forma de representar o problema do mundo dos blocos de forma semelhante ao da locomoção com a utilização de uma matriz LxC, onde a L(linha) tem o valor do estado do ambiente e C(coluna) a ação a ser tomada. As células guardam o valor de Q.

A Figura 3.3 mostra essa representação. A utilização do alfabeto como forma de representar os estados é utilizada para uma melhor visualização e devem ser visto como: A - todos os blocos sobre a superficie, B - bloco 1 sobre o bloco 2 bloco 3 bloco 4 bloco 5 sobre a superficie, C - bloco 1 sobre o bloco 2 sobre o bloco 3 bloco 4 bloco 5 sobre a superficie, D - bloco 1 sobre o bloco 2 sobre o bloco 3 sobre o bloco 4 bloco 5 sobre a superficie, E - bloco 1 sobre o bloco 2 sobre o bloco 3 sobre o bloco 4 sobre o bloco 5 sobre a superficie, etc.

Essa forma de representar nos traz alguns problemas, sendo o maior deles o de que todos os estados e ações estão presente mesmo que esses não sejam visitados ou utilizados, ou seja, todos os estados e ações possíveis estão presentes desde o início do processo de aprendizagem. Isso pode também inviabilizar a utilização da estrutura, caso o número de blocos seja muito grande.

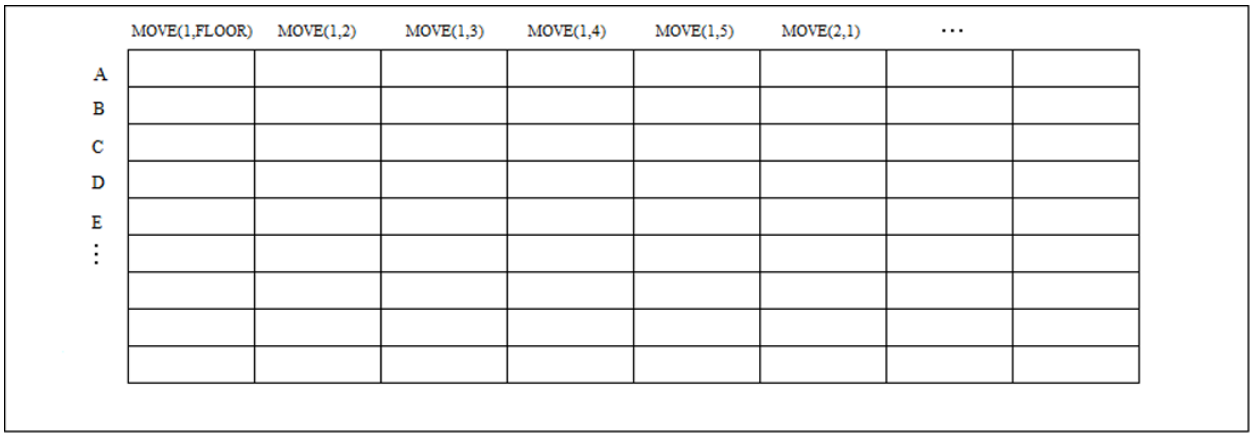

Figura 3.3: Representação do mundo dos blocos com a utilização de Matriz como forma de representação.

Uma das formas utilizadas para resolver este problema é a utilização da lógica de primeira ordem (relacional), como forma de representação. Com a utilização de predicados é possível representar o mesmo ambiente de forma mais clara. A Figura 3.4 mostra como representar o estado da Figura 3.2 e como será o estado após a execução da ação, utilizando-se a representação relacional.

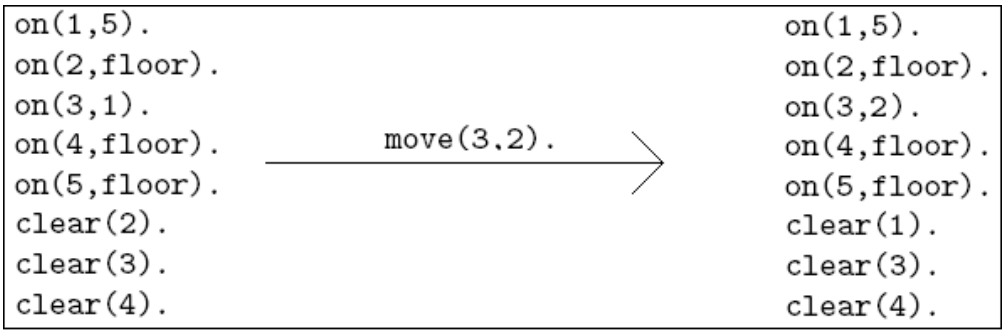

Figura 3.4: Representação do mundo dos blocos com a utilização de predicados como forma de representação.

A ação também é representada por predicados e que através de regras é possível atingir o outro estado. Com isso, somente aqueles estados e ações só apareceram se eles forem necessários.

Para enter melhor o funcionamento dessa representação, que foi a utilizada neste trabalho, a seguir serão abordados conceitos básicos importantes para o seu entendimento.

3.2.2 Representação Relacional

A descrição relacional é baseada em Linguagem de Primeira Ordem (LPO), a qual é uma linguagem de representação de propósito geral que permite a descrição de objetos e relações entre objetos. A LPO fornece ao usuário a liberdade para descrever os objetos da maneira mais apropriada para seu domínio (Russell e Norvig, 2003), como no exemplo a seguir:

"Rei John, o maldoso, reinou na Inglaterra em 1200."

Objetos: John, Inglaterra, 1200.

Relações:reinou.

Propriedade:maldoso, rei.

Nesse exemplo, *rei* é considerado como uma propriedade da pessoa. Porém, se fosse mais apropriado, *rei* poderia ser uma relação entre pessoas e países ou, ainda, entre países e pessoas em um mundo onde cada país tem seu *rei* (Ferro, 2002).

Os sistemas que utilizam esse tipo de linguagem de representação, são chamados de sistema de aprendizado relacional (Ferro, 2002).

A notação e a terminologia está baseada em (Poole et al., 1998; Russell e Norvig, 2003; Morik et al., 1993), e sendo constituido por:

- Uma **linguagem formal** - que especifica as sentenças que podem ser utilizadas para expressar conhecimento.
- Uma **semântica** - que especifica o significado da sentença na linguagem, tanto como entrada ou saída.
- Uma **teoria de raciocínio** - que é uma possível especificação não determinística de como uma resposta pode ser derivada de uma base de conhecimento.

A linguagem formal define os símbolos e como eles devem ser organizados. E esses são divididos em variáveis, termos, átomos, predicados, fatos e regras.

A **variável** é uma sequência de caracteres que indica um **termo**, e que pode ser uma variável ou uma constante. Os **predicados** são palavras que se referem as relações e os **átomos** são encontrados na forma p ou $p(t_1, \cdots, t_n)$ onde p é um predicado e t_i denota um termo. Alguns exemplos de átomos são: Aula(cse,s201), Feliz(João), Ensolarado e JogoBasquete(Assis,Franca) (Poole et al., 1998; Russell e Norvig, 2003).

O **fato** é considerado um átomo como cláusula (Poole et al., 1998).

As **regras** são cláusulas com dois literais que utiliza a forma a→b, em que a é conhecido como o corpo, conhecido também como premissa e b é a cabeça da regra, se referenciando a conclusão (Morik et al., 1993). Por exemplo, temperatura(Pessoa, T)&grausTemperatura(T,37) → febre(Pessoa). A conclusão é acionada se a premissa for verdadeira, ou seja, se a pessoa João (Pessoa=João) está com 37 graus de temperatura (T=37), então, pela regra, ele está com febre.

Além disso, a base de conhecimento deve ser bem planejada, ou seja, a forma de armazenar os dados e de acessá-los, de tal modo que esses processos sejam eficientes (Poole et al., 1998).

Na próxima seção será definido o ARR e a descrição do funcionamento do algoritmo principal que é utilizado por todos os sistemas ARR. O que diferencia cada sistema é a utilização de diferentes mecanismos de regressão. Um desses mecanismos será abordado posteriormente.

3.3 Aprendizado por reforço relacional

A definição do ARR é muito semelhante a definição do AR feito no capítulo anterior. A diferença entre os dois está no fato de que o ARR utiliza a representação relacional como meio de representação dos estados e ações, e do conhecimento prévio do ambiente. Ambos não são contemplados no AR.

Definição 2 *O aprendizado por reforço relacional é definido como:*
Dado

- *um conjunto de possíveis estados S (representado em formato relacional),*
- *um conjunto de possíveis ações A (representado em formato relacional),*
- *uma desconhecida função de transição δ :Conjunto de estados (S) x Conjunto de ações (A)$\rightarrow$ S (essa função pode ser não determinística),*
- *uma função de recompensa r: S x $A\rightarrow \mathbb{R}$.*
- *conhecimento prévio geralmente válido sobre o meio.*

Encontrar

uma política para selecionar ações π^*: $S\rightarrow$ A que maximiza o valor da função $V^{\pi}(s)$ para todo estado $s\in S$ (Driessens, 2004).

O conhecimento prévio pode ser visto como um grande número de diferentes tipos de informação. Possibilidades incluem informação sobre a forma da função valor, conhecimento parcial sobre o efeito das ações, similaridade entre diferentes estados e outros. Como exemplo, no caso do Mundo dos blocos, pode-se utilizar número de blocos como conhecimento prévio (Driessens, 2004).

O cálculo de Q é feita por um algoritmo de regressão que usa as infromações(estado, ação e Q) encontrados durante a exploração do ambiente. O algoritmo de regressão faz uma aproximação do função de avaliação Q, que aproxima as predições dos valores Q para todo par (estado,ação), mesmo que esses nunca sejam encontrados durante a exploração. O uso de regressão para $Q-learning$ não apenas reduz a quantidade de memória e o tempo necessário, mas também permite ao agente fazer predições da qualidade de pares (estado, ação) (Driessens, 2004).

O cálculo de Q pode ser feita por um número diferente de métodos de regressão como redes neurais artificiais, mínimos quadrados, árvores de regressão, métodos baseados em instância, entre outros (Driessens, 2004).

Três diferentes algoritmos de regressão têm sido utilizados na literatura como mecanismo para o cálculo do valor Q: o algoritmo de árvore de regressão chamado *TG* (Driessens, 2001), um

algoritmo baseado em exemplo relacional conhecido como *RIB* (Driessens, 2003) e um algoritmo regressivo baseado em processos Gausianos e kernels para domínios estruturais chamado *KBR* (Gärtner et al., 2003).

Esses diferentes mecanismos são considerados como a segunda parte de todo o sistema de ARR. A primeira parte é feita com a técnica de AR, que utiliza como base o algoritmo *Q-learning*.

A idéia básica da primeira parte é apresentada através do Algoritmo 6. A diferença entre esse algoritmo e o *Q-Learning* apresentado no algoritmo 2, é que esse armazena todos os exemplos para depois calcular o valor de Q de forma reversa, antes de armazenar a experiência. Essa melhoria, já citada por Mitchell (1997), é feita para uma rápida convergência, mas com um gasto maior de memória.

Algoritmo 6 Algoritmo de aprendizado por reforço relacional

inicializa a hipotese $Q - function\ \hat{Q}_0$
$e \leftarrow 0$
repita
  $Exemplos \leftarrow \emptyset$
  geração de um estado inicial s_0
  $i \leftarrow 0$
  repita
    escolha a_i para s_i usando uma politica derivada da hipotese $\hat{Q}_e$
    execute a_i, observe r_i e s_{i+1}
    $i \leftarrow i + 1$
  até que s_i seja terminal
  para $j = i - 1$ to 0 **faça**
    geração de exemplo $x = (s_j, a_j, \hat{q_j})$ onde $\hat{q_j} \leftarrow r_j + \gamma \max_a \hat{Q}_e(s_{j+1}, a)$
    $Exemplos \leftarrow Exemplos \cup \{x\}$
  fim para
  Atualize $\hat{Q}_e$ usando Exemplos e um algoritmo de regressão relacional para produzir $\hat{Q}_{e+1}$
  {Utilizando o algoritmo 7}
  $e \leftarrow e + 1$
até que não exista mais episódios

Inicia-se a função de avaliação Q, pois existe a possibilidade de incluir alguma informação inicial sobre o meio, embora o mecanismo de regressão retorne o mesmo valor padrão para cada par (estado, ação) (Driessens, 2004).

O algoritmo inicia os episódios de aprendizado como um Q-learning padrão (Sutton e Barto, 1998) (Mitchell, 1997) (Kaelbling et al., 1996). Para a estratégia de exploração, o sistema converte a atual aproximação da função de avaliação Q na política usando a estatística de Boltzmann, dada pela equação (2.4) (Kaelbling et al., 1996).

Durante o episódio de aprendizado, todos os estados encontrados e as ações selecionadas são armazenados, junto com as recompensas conectadas a cada par (estado, ação). No fim de cada episódio, quando o sistema encontra um estado terminal, ele utiliza de retro-alimentação das re-

compensas e a aproximação da função de avaliação Q atual para gerar uma aproximação do valor Q para cada par (estado, ação) encontrado.

Então o algoritmo apresenta o conjunto de triplas (estado, ação, valor Q) para o mecanismo de regressão relacional que irá usar esse conjunto de exemplos para atualizar a estimativa da função de avaliação Q. Assim o algoritmo continua executando o próximo episódio.

Este algoritmo é essencialmente idêntico ao tradicional $Q - Learning$, exceto pela forma de atualização do valor da função de avaliação Q, Q_e (representado de acordo com o mecanismo escolhido após um episódio e).

A próxima seção será abordado o mecanismo que utiliza árvore de regressão e que foi utilizado neste trabalho.

3.4 Algoritmo TG

Antes de apresentarmos este algoritmo, faz-se necessario apresentar o embasamento teórico das técnicas que de alguma forma contribuiram para a construção do TG, descrito a seguir. Será apresentado conceitos sobre árvores de decisão e alguns dos seus métodos.

3.4.1 Árvores de decisão

Uma árvore de decisão consiste de uma hierarquia de nós internos e externos que são conectados por ramos. O nó interno, também conhecido como nó decisório ou nó intermediário, é a unidade de tomada de decisão que avalia através de teste lógico qual será o próximo nó descendente ou filho. Em contraste, um nó externo (não tem nó descendente), também conhecido como folha ou nó terminal, está associado a um rótulo (Árvores de Classificação) ou a um valor (Árvores de Regressão) (da Silva, 2005).

Em geral, o procedimento de uma árvore de decisão é o seguinte: apresenta-se um conjunto de dados ao nó inicial (ou nó raiz que também é um nó interno) da árvore; dependendo do resultado do teste lógico usado pelo nó, a árvore ramifica-se para um dos nós filhos e este procedimento é repetido até que um nó terminal é alcançado. A repetição deste procedimento caracteriza a recursividade da árvore de decisão (da Silva, 2005).

A Figura 3.5 ilustra uma árvore de decisão relacionada as condições climáticas para a prática de um esporte em que esse fator é fundamental, o tenis. Vale notar que cada nó interno na árvore especifíca um teste de algum atributo do exemplo e que cada percurso, da raiz à folha, representa uma regra que no caso da figura é uma regra de classificação (Mitchell, 1997).

A construção da árvore depende da forma que os atributos dos exemplos são selecionados, pois é interessante que atributos mais relevantes estejam em nós perto do nó raiz. Aqueles que são considerados ineficientes podem ser eliminados. Se esses fatos acontecerem, regras mais eficientes seram geradas (Castiñeira, 1990).

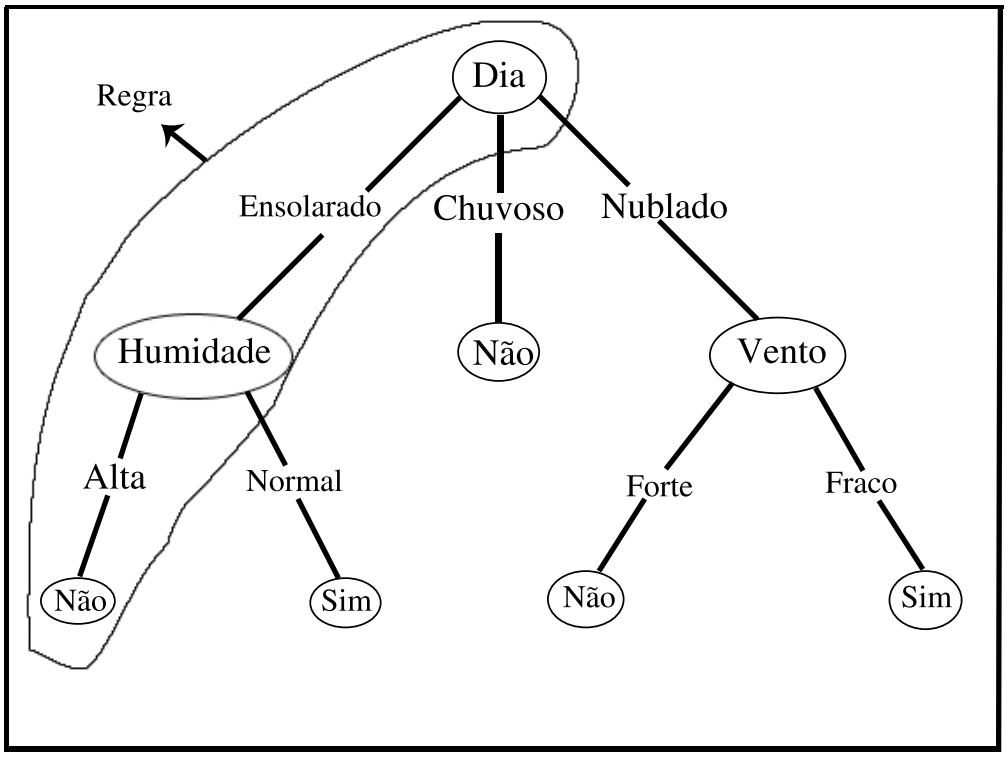

Figura 3.5: Árvore de Decisão

A partir da árvore de decisão, pode ser formulado um conjunto equivalente de regras. Por exemplo, considerando a Figura 3.5, para o caso onde o dia esteja com as seguintes condições:

- Dia = ensolarado.
- Temperatura = baixa.
- Humidade = alta.
- Vento = fraco.

a resposta do sistema que utiliza a árvore será negativa, indicando a impossibilidade de jogar tênis. A regra gerada pode ser especificada como **Se dia ensolarado e Humidade alta então não é possivel jogar tênis**.

Uma árvore completa poderia ser construída representando todas as possíveis perguntas com todas as possíveis respostas. Já o exemplo é considerado uma árvore de decisão parcial, representando o caminho resultante da condição existente , ou seja, a regra criada (Castiñeira, 1990).

No caso das árvores de decisão binária, cada nó intermediário divide-se exatamente em dois nós descendentes: o nó esquerdo e o nó direito. Quando os dados satisfazem o teste lógico do nó intermediário seguem para o nó esquerdo e quando não satisfazem seguem para o nó direito. Logo, uma decisão é sempre interpretada como verdadeira ou falsa. Deve ser mencionado que, restringimos a nossa descrição de divisão para árvores binárias, pois estas serão empregadas nesta

dissertação. Contudo, na literatura há árvores de decisão com várias divisões e sua descrição pode ser encontrada em Zighed (da Silva, 2005).

Além da árvore binária, essa dissertação utiliza árvore de regressão (da Silva, 2005), que assume como forma uma árvore de decisão onde o valor do nó folha é númerico, como mostrado na representação gráfica da Figura 3.6.

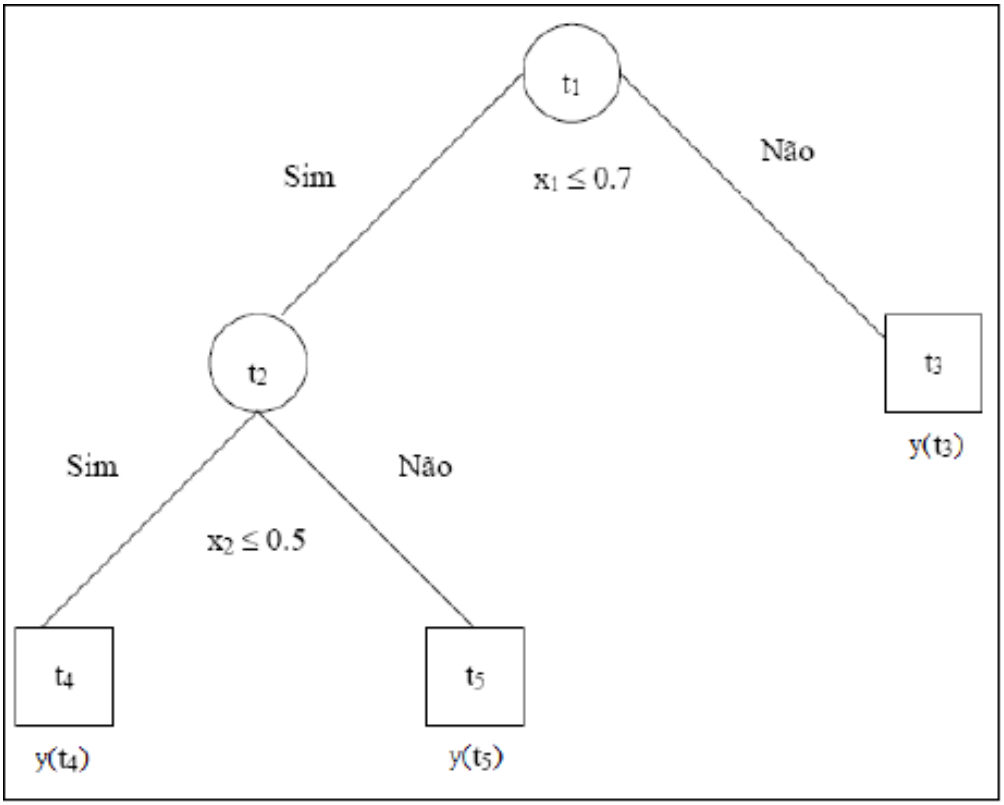

Figura 3.6: Árvore de Regressão

Os círculos representam os nós internos (intermediários ou decisórios); os quadrados representam os nós folhas ou terminais, contém uma constante (geralmente, uma média) ou uma equação para o valor previsto de um determinado conjunto de dados; as linhas representam os ramos que interligam dois nós; e x_1 e x_2 representam as variáveis decisórias. Chama-se de variável decisória a variável de entrada que levará a uma nova divisão da árvore de decisão, em relação a um possível valor.

Breiman et al. (1984) estabeleceu que as árvores modeladas eram desnecessariamente grandes. E como o tamanho da árvore está diretamente relacionado com a complexidade do modelo, árvores grandes implicam em modelos difíceis de interpretar e compreender. Além disso, o modelo pode apresentar um problema conhecido como *overfitting*, ou seja, ajuste demasiado aos dados de treinamento. O modelo precisa apresentar uma margem de generalidade para dados que não fazem parte do conjunto de treinamento (da Silva, 2005).

Para resolver estes dois problemas, (Breiman et al., 1984) define e usa a técnica de poda da árvore. Depois que a árvore é completamente desenvolvida, alguns nós intermediários são trans-

formados em nós terminais (folhas). Isto aumenta a taxa de erro do modelo em relação aos dados de treinamento, mas generaliza a árvore, ou seja, reduz a taxa de erro para outros dados, como os dados de teste. Além disso, esta operação reduz o tamanho da árvore, que significa um modelo final mais interpretável.

A podagem pode ser dada de duas circunstâncias. Na primeira, pode ser usada para interromper o crescimento da árvore mais cedo, chamada de pré-podagem ou poda descendente. Nesse caso verifica-se se a divisão é confiável ou não (da Silva, 2005).

Na segunda, pode acontecer com a árvore já completa, chamada de pós-podagem ou poda ascendente. Na pós-podagem a árvore é gerada no tamanho máximo e então a árvore é podada aplicando-se métodos de evolução confiáveis. A sub-árvore com o melhor desempenho será a escolhida (da Silva, 2005).

A pré-podagem é mais rápida porém menos eficiente que a pós-podagem pelo fato do risco de interromper o crescimento da árvore ao selecionar uma árvore sub-ótima (Breiman et al., 1984).

3.4.2 *Top-down Induction of Decision Trees*

A familia *Top-down Induction of Decision Trees* (TDIDT) é conhecida por gerar árvores do nó raiz ao nó folha, ou seja, as árvores são construídas a partir do no raiz, se extendendo até o nó folha (Castiñeira, 1990).

O algoritmo TDIDT, genérico especificado em Blockeel e Raedt (1998), consiste em duas fases: a de crescimento e a de poda. A fase de crescimento inicia a partir do recebimento do conjunto de exemplos, chamaremos esse conjunto de exemplos de E. Dado E, sobre ele é realizado testes sobre os atributos a fim de encontrar valores para possíveis testes. Por exemplo, se um exemplo possui o atributo cor, onde observou-se os valores verde e azul para este atributo, então é conferido os testes cor = verde e cor = azul.

Assim, os testes são avaliados (divisão ótima) para descobrir qual o melhor atributo para ser inserido em um determinado nó (Blockeel e Raedt, 1998).

Antes de ser inserido, uma verificação (critério de parada) é feita para determinar se esse atributo é bom o bastante para ser inserido como teste em uma sub-árvore. Em caso positivo, o atributo é colocado na sub-árvore, mas o contrário, um nó folha é construído com a informação relevante sobre o exemplo (Blockeel e Raedt, 1998).

Para evitar um alto crescimento da árvore é executado sobre ela uma mecanismo de poda. Esse mecanismo foi apresentado no final da seção 3.4.1.

As operações de divisão ótima, critério de parada, seleção de informação relevante e poda são ajustadas de acordo com o objetivo da aplicação, seja ela classificação, regressão ou agrupamento de dados. A Tabela 3.1 mostra as possíveis funções para cada tipo de aplicação anteriormente apresentado.

Entre os algoritmos da família TDIDT estão o ID3, *Top-down Induction of Logical Decision Trees* (TILDE), algoritmo G e TG, apresentados a seguir.

	divisão ótima	critério de parada	seleção de informação relevante	Poda
classificação	gain(ratio) gini index	x^2-test min. convergence MDL	mode	C4.5 valid. set
regressão	intra-cluster variance of target value	F-test, t-test min. convergence MDL	mean	valid. set
clusterização	intra-cluster variance	F-test, t-test min. convergence MDL	prototype identity	valid. set

Tabela 3.1: Funções de tratamento de árvore. Funções que são utilizadas segundo a sua especialidade em determinados pontos do algoritmo

Algoritmo ID3

O algoritmo básico do ID3 (Mitchell, 1997), constroi a árvore de decisão na forma "*top-down*" (análise descendente), onde cada exemplo é avaliado através de um teste estatístico para verificar sua representatividade. O melhor exemplo é selecionado e utilizado como teste no nó raiz da árvore. Os outros exemplos são inseridos nos nós descendentes, estabelecendo um ramo para este conjunto de exemplos. O processo é então repetido para os próximos conjuntos de exemplos, onde cada exemplo é testado com o seu respectivo nó, caso exista (Mitchell, 1997).

Para selecionar os atributos mais significativos a ser inserido em cada nó da árvore, o algoritmo ID3 utiliza uma avaliação dos atributos de cada exemplo. Este teste é conhecido como ganho de informação (*informution gain*), definido como uma medida da eficácia de um atributo na classificação da formação do exemplo. Mais precisamente, ganho de informação, *Gain(E, X)*, de um atributo X, relativo a coleção de exemplos E é definido como:

$$Gain(E, X) \equiv Entropy(E) - \sum_{v \in Value(X)} \frac{|E_v|}{|E|} Entropy(E_v) \tag{3.1}$$

onde $Value(X)$ é o conjunto de todos os possiveis valores para o atributo X e E_v é o subconjunto de E para todo atributo de X que tenha o valor v(e.g.,$E_v = \{e \in E | X(e) = v\}$) (Mitchell, 1997).

A *Entropia* é definido como uma medida da impureza na formação de uma coleção de exemplos. Dada uma coleção de exemplos E, contendo exemplos positivos e negativos, a entropia relativa de S é dado pelo classificador booleano:

$$Entropy(E) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \tag{3.2}$$

onde $p_{\oplus}$ são os exemplos positivos e $p_{\ominus}$ os exemplos negativos em E.

Para ilustrar, suponha E é uma coleção de 14 exemplos, sendo 9 positivos e 5 exemplos negativos (que adotamos a notação [9+,5-] para resumir essa amostra de dados). Então a entropia de E relativa a esta classificação booleana é:

$$\begin{aligned} Entropy[9+, 5-] &= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) \\ &= 0.940 \end{aligned}$$

O apresentado é valido para distribuições booleanas, onde podemos validar o positivo e negativo. Uma forma mais genérica, se os atributos possuem c valores diferentes, a entropia relativa de E pode ser calculada:

$$Entropy(E) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i \tag{3.3}$$

onde p_i é a proporção de exemplos E pertencentes à classe i (Mitchell, 1997).

Algoritmo TILDE

As árvores de decisão tem colocado o foco na utilização de atributos. O TILDE é um sistema que realiza processo de classificação, regressão ou agrupamento de dados através da geração de uma árvore de decisão de lógica de primeira ordem. Nessa estrutura é utilizada fatos ao invés de atributos. A Figura 3.7 mostra como é organizada a arquitetura do sistema TILDE (Blockeel e Raedt, 1998).

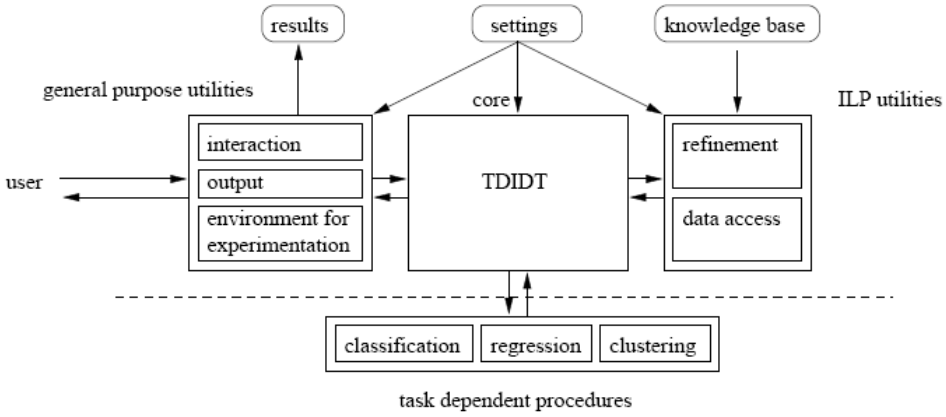

Figura 3.7: Algoritmo TILDE. Mostra a arquitetura geral do algoritmo TILDE onde as setas denotam o fluxo da informação (Blockeel e Raedt, 1998)

O sistema TILDE (Blockeel e Raedt, 1998) utiliza como base um algoritmo genérico TDIDT com três modulos de apoio:

- Modulo utilidade contém o código para aplicar o operador de refinamento a uma cláusula (para gerar um conjunto de testes que será considerado no nó) e todas as funções de acesso aos dados.

- Modulo objetivo que contém toda implementação de três diferentes opções de trabalho. Entre as opções estão: Classificação, Regrassão e Agrupamento de dados.

- Modulo auxiliar que auxilia na execução e avaliação do sistema, e.g. interface com o usuario, produz a árvore em um formato legível, etc.

O sistema recebe dois tipo de entrada: um arquivo de configuração, especificando varios parametros do sistema, e.g., qual modo deve rodar e outros. E a base de conhecimento que é dividido em um conjunto de exemplos e o conhecimento prévio. O sistema então irá gerar um ou dois arquivos de saida como resultado do processo indutivo (Blockeel e Raedt, 1998).

É importante observar que no modulo objetivo cada uma das opções possuem sua função de divisão ótima, critério de parada, armazenamento de informação e poda (Blockeel e Raedt, 1998).

Um novidade no TILDE é a questão da utilização do operador de refinamento. Ele mapea uma cláusula c para um conjunto de cláusulas e este é especificado na forma *rmode(n:cláusulas)*, indicando que cláusulas podem ser inseridas e o número(n) máximo de vezes que a clausula pode ser inserida (Blockeel e Raedt, 1998).

Para permitir a unificação das variáveis entre os testes utilizados em diferentes nós dentro de um caminho da árvore, o conjunto de cláusulas dadas na declaração *rmode* inclui informações de modo a utilizar variáveis. Modos possíveis são '+', '-' e'+-'. Um '+' indica que a variável deve ser utilizada como uma variável de entrada, ou seja, a variável que deve ocorrer em um dos testes sobre o caminho a partir do topo da árvore à folha que será prorrogado. Um '-' relaciona a saída, ou seja, que as variáveis associadas ainda não deveriam ocorrer. '+ -' Significa que ambas as opções são permitidas, ou seja, extensões podem ser gerados tanto com uma já ocorrido ou uma completamente nova variável (Blockeel e Raedt, 1998).

A utilização do operador de refinamento *rmode(3:on(X,Y))*, onde X e Y é uma variavel, indica que o fato *on(X,Y)* pode ser utilizado como testes em até três nós na qual o nó está no caminho.

Algoritmo G

O algoritmo G foi desenvolvido para ser um módulo de um algoritmo de AR, o *Q-learning*.

Chapman e Kaelbling (1991), desenvolveram o algoritmo G motivado pela ineficiência utilização da tabela Q, pois essa necessita ser inicialmente tão grande quanto o ambiente e, considerando que na maior parte dos domínios, grandes regiões da tabela deveram ter entradas idênticas, podemos então colocar um valor Q para uma região, basta descobrir os locais considerados irrelevantes. Assim podemos salvar tanto espaço (para armazenar os valores) e tempo (dado que a experiência com qualquer estado da região pode ser utilizado Q único para atualizar o valor).

O algoritmo G constrói incrementalmente uma árvore estruturada Q. Ele começa supondo que todos os exemplos são irrelevantes, assim ele faz com que toda tabela esteja em um único nó. Ela recolhe os valores Q e os dados estátisticos dos exemplos dentro deste nó. Quando se descobre que um exemplo é relevante, ele divide o espaço em dois subespaços (dois nós) sendo um correspondente ao novo exemplo. Em seguida, ele recolhe estatísticas (ação e relevância) dentro de cada um desses nós. Esses nós, por sua vez, pode ser divididos, dando origem a uma árvore estruturada Q. O sistema atua com base nas estatísticas Q existentes no nó folha que corresponde a uma entrada (Chapman e Kaelbling, 1991).

A técnica de divisão do algoritmo G está relacionada com algoritmos existentes, tais como ID3 (Quinlan, 1986), e CART (Breiman et al., 1984), para indução de árvores de decisão (*inducing decision trees*). A diferença crucial é que o algoritmo de árvore de decisão apresentam com pares de entrada/saída em vez de dados de reforço, no caso o valor Q. Por este motivo, os testes estatísticos usados para fazer a divisão devem ser diferentes. Além disso, nosso trabalho tem enfatizado fazer decisões incrementais com um valor fixo de computação por exemplo, em vez de aprender sem profundidade ou uma árvore menor (Chapman e Kaelbling, 1991).

O método incemental de construção da árvore G pode falhar se o grupo de entrada são coletivamente relevantes mas individualmente irrelevantes. Se considerarmos o sistema perceptual de um agente para ser parte do "ambiente" do seu sistema de aprendizagem, como temos, então esta limitação pode ser colocado nesse sistema, em vez de todo o mundo (Chapman e Kaelbling, 1991).

Um problema similar ocorre na saida, caso o número de ações seja muito grande, onde o agente pode não executar cada ação para cada estado. (Chapman e Kaelbling, 1991) acredita, porém, que o algoritmo G deve ser diretamente aplicável a este problema.

Algoritmo TG

O algoritmo TG foi desenvolvido pela combinação entre o TILDE e o algoritmo G. Ele é considerado um processo incremental que utiliza operador de refinamento utilizado sitema TILDE. Driessens (2004) usa a árvore de regressão relacional na qual os fatos são testes nos nós internos e nos nós folha ele armazena o Q.

Esse é apresentado no Algoritmo 7, e mostra como Driessens (2004) utiliza a árvore de regressão, como mecanismo de regressão do algoritmo ARR, para atualizar as informações de forma incremental para cada exemplo. Uma importante característica é que os exemplos podem ser descartados depois que eles são processados, isto faz com que haja economia de memória.

O algoritmo TG recebe o conjunto de triplas (estado, ação, valor Q). Realiza o teste de divisão ótima para indicar os exemplos úteis. Com o valor do estado ele percorre os nós internos até alcançar um nó folha, onde o valor do Q e as estatísticas são atualizadas. Quando o estado não consegue ser descrito sobre todo o caminho, ele verifica no nó folha a possibilidade de divisão, caso não seja possível somente as estatísticas são atualizadas, caso contrário, ele gera 2 novos nós folhas.

Algoritmo 7 Algoritmo TG

```
inicializa pela criação da árvore com uma folha e com estatísticas vazias
para cada exemplo que se torna útil faça
  aplica o exemplo sobre a árvore usando testes internos até o nó folha
  atualiza as estatísticas do nó folha de acordo com o novo exemplo
  se a estatística na folha indica que uma nova divisão é necessaria então
    gera um nó interno usando o teste indicado
    gera 2 novas folhas com estatísticas vazias
  fim se
fim para
```

As estatísticas para cada nó folha é uma forma de poda na qual Driessens (2004) utiliza. Para isso ele utiliza o Teste-F (Snedecor e Cochran, 1989) com um nível de significância de 0.001 para realizar está decisão, dividindo depois que o número de exemplos é coletado e o teste de significância se torna de alta confiança. Mais detalhes sobre como é a função, pode ser encontrada em Driessens (2004)

Como forma de exemplificar o funcionamento do algoritmo foi elaborada uma sequência de Figuras: 3.8 a 3.10, que mostram os passos do sistema ARR com o mecanismo TG no mundo dos blocos.

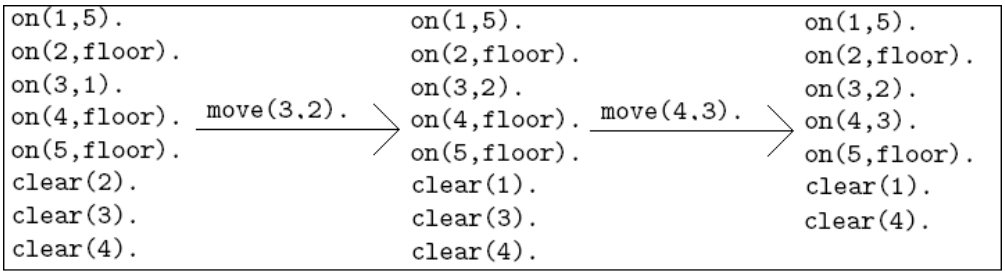

Figura 3.8: Etapa de ação sobre o meio

Na Figura 3.8 pode ser visto que o agente parte do estado inicial e por meio de ações (operação de *move(x,y)*), ele atinge o seu objetivo. Essa etapa é considerada apenas a interação do agente sobre o meio, gerando os exemplos que futuramente será armazenado.

Após ser gerado os exemplos é feito o calculo do valor Q para cada exemplo. Esse calculo é feito de forma inversa ao processo de interação com o ambiente, ou seja, é feito do estado final, sendo este não incluso, para o estado inicial. O processo está representado na Figura 3.9.

Por fim, o mecanismo de regressão é atualizado com os exemplos gerados. Para cada exemplo é realizado testes para verificar a sua utilidade e assim, para os exemplos úteis, é atualizado o mecanismo, no qual se extende o exemplo até o nó folha. O próximo passo é atualizar as estatísticas e depois verificar a necessidade de uma nova divisão.

A Figura 3.10 mostra uma simulação da árvore construída pelos exemplos. O valor Q_0 é definido como um valor padrão, isto é, pode-se utilizar o valor 0.

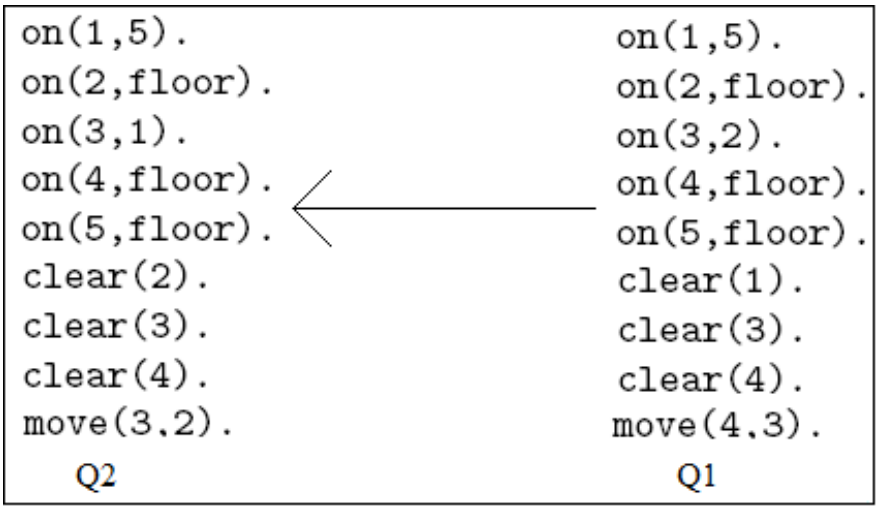

Figura 3.9: Calculo do valor Q

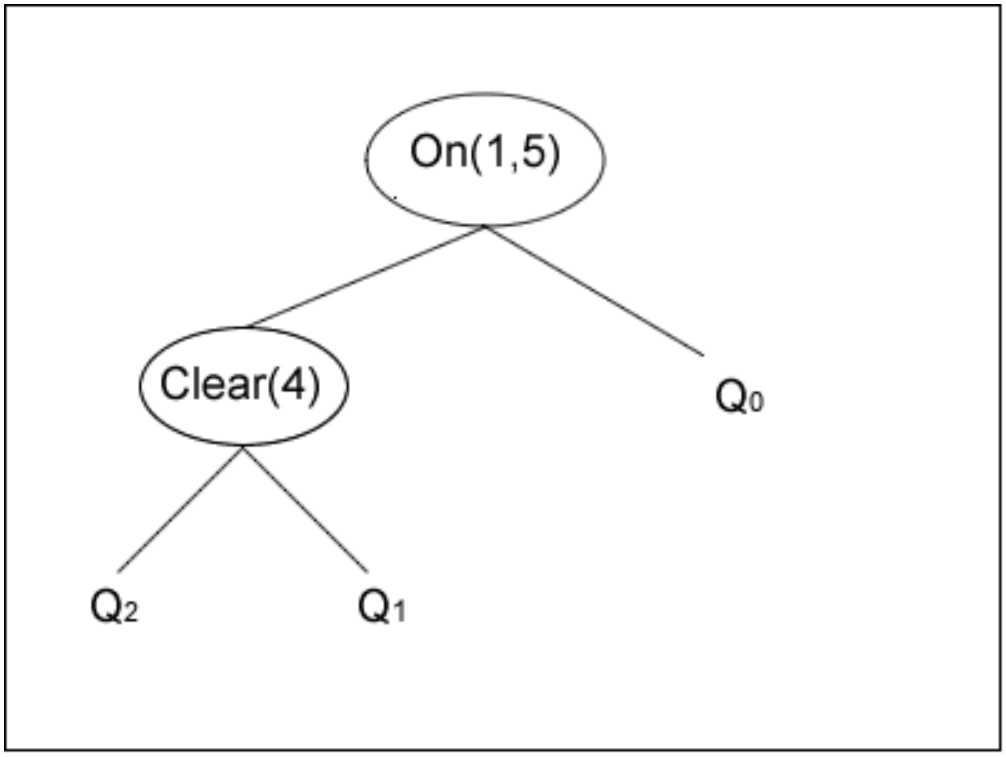

Figura 3.10: Atualização do mecanismo de regressão

3.5 Considerações Finais

A representação relacional foi vista como uma representação econômica que permite a descrição de objetos e relações entre objetos. O Capítulo mostra ainda a ligação entre o AR e essa representação. Finalizando pela apresentação de métodos que antecederam ao desenvolvimento do algoritmo TG, que é utilizado como base nesse projeto.

CAPÍTULO

4

Sistema de Aprendizado Proposto

Este capítulo está organizado da seguinte forma. A primeira seção aborda como foi elaborada a representação que é utilizada durante todo o trabalho e a seguir é a apresentada uma modificação no algoritmo TG proposta neste trabalho.

4.1 Representação utilizada

Na seção 3.2 do capítulo 3 vimos a importância de uma boa representação do conhecimento como um fator da aprendizagem e a representação relacional como uma forma de representar o ambiente de modo econômico.

Esta seção tem como objetivo esclarecer como esta representação é utilizada pelo sistema proposto. Vamos utilizar como exemplo as notações e o ambiente do mundo dos blocos. Os detalhes sobre o problema do Mundo dos blocos podem ser encontrados na seção 6.2 do capítulo 6.

Assim, para que a representação fosse possível, cada fato foi associado a um valor binário. Esse valor deve pertencer à 2^x, onde x é um valor inteiro dentro do intervalo que compreende o valor 0 até o número total de fatos(n) subtraído de uma unidade (n-1). Observe que cada fato foi associado a um valor binário pertencente a uma potência de 2. A Figura 4.1 mostra como essa associação dos fatos é feita para o mundo dos blocos que utiliza o número de blocos igual a 3.

Dessa forma, podemos representar o estado do ambiente que está na Figura 4.2 utilizando a lógica de primeira ordem pela união dos fatos: **on(1,floor),on(2,floor),on(3,1),clear(2),clear(3)**. Utilizando os valores especificado na Figura 4.1, o conjunto de fatos pode ser representado ao valor inteiro 1102 que em binário é representado por 100001001110.

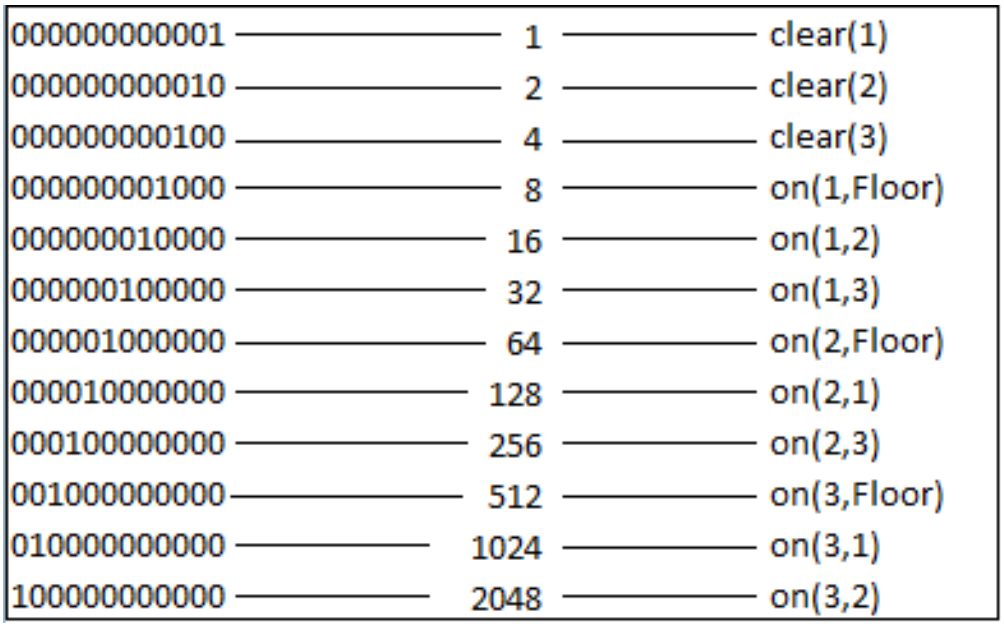

Figura 4.1: Associação de Dados. Mostra como é feita a associação dos dados da forma relacional para a binária no ambiente do Mundo dos blocos com 3 blocos.

Note que o valor do estado do ambiente é especificado pela soma dos códigos binários correspondentes aos fatos. Um ponto a ser notado é a possibilidade de decomposição do estado, que de forma simples, é possivel obter os seus fatos constituintes.

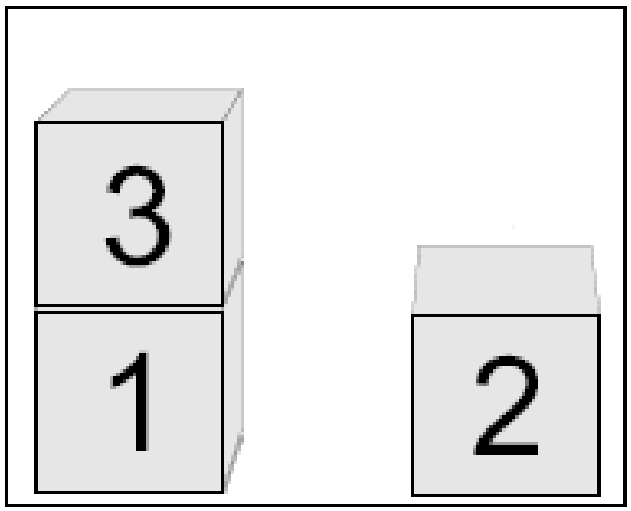

Figura 4.2: O mundo dos blocos

As ações são representadas de forma mais simples, pois estas não necessitam de ser combinadas. A Figura 4.3 mostra as possíveis ações para a configuração acima.

Esse tipo de representação apresentado é utilizado em todos os ambientes, tomados a sua devida proporção, na qual os testes foram realizados, ou seja, nos simuladores do Mundo dos Blocos, da atenção compartilhada e na cabeça robótica.

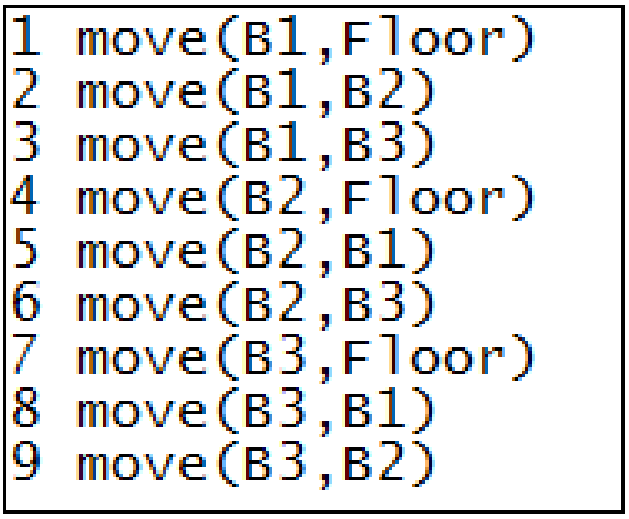

Figura 4.3: As possíveis ações para o mundo dos blocos

A partir do momento em que a representação foi definida foi possível pensar em desenvolver um sistema de aprendizado para ser inserido na arquitetura robótica.

4.2 Algoritmo de Aprendizado Proposto

Dentre as técnicas proposta por Driessens (2004), o algoritmo TG é o que mais satisfaz ao nosso interesse, por se tratar de algoritmo mais simples de implementar, possui uma forma de armazenamento de informação mais clara. Segundo Driessens (2004), o TG é mais rápido e pode armazenar mais exemplos que as outras técnicas. Assim, o sistema RRL com o algoritmo TG é utilizado como base para o desenvolvimento do sistema proposto.

Na análise do sistema ARR feita na seção 3.3 do capítulo 3 nota-se que ele reúne todos os exemplos antes de inserir no árvore de regressão. Essa forma, já citada por Mitchell (1997), é uma maneira rápida de convergência, mas com um custo maior de armazenamento e esse é um custo que queremos evitar, por isso usaremos a forma mais simples do algoritmo *Q-learning*.

Um outro fator que se observa está no agrupamento feito pelo sistema ARR com o TG para que em um determinado conjunto de estados seja definido uma única ação. Isto é feito selecionando-se os exemplos significativos a serem inseridos e usando o teste estatístico. Considerando, por exemplo, a Figura 4.4 como ambiente, em um sistema de locomoção, esse agrupamento é um fator interessante de se aplicar.

Essa Figura usa o mesmo ambiente definido na seção 2.3.2 do Capítulo 2. Ela ainda mostra o agrupamento de estados para uma determinada ação.

A Figura mostra também um número muito grande de estados dependentes de uma ação. Cada bloco está relacionado ao desenho circular em que está inserido e cada desenho tem uma ação corespondente (indicado por uma seta).

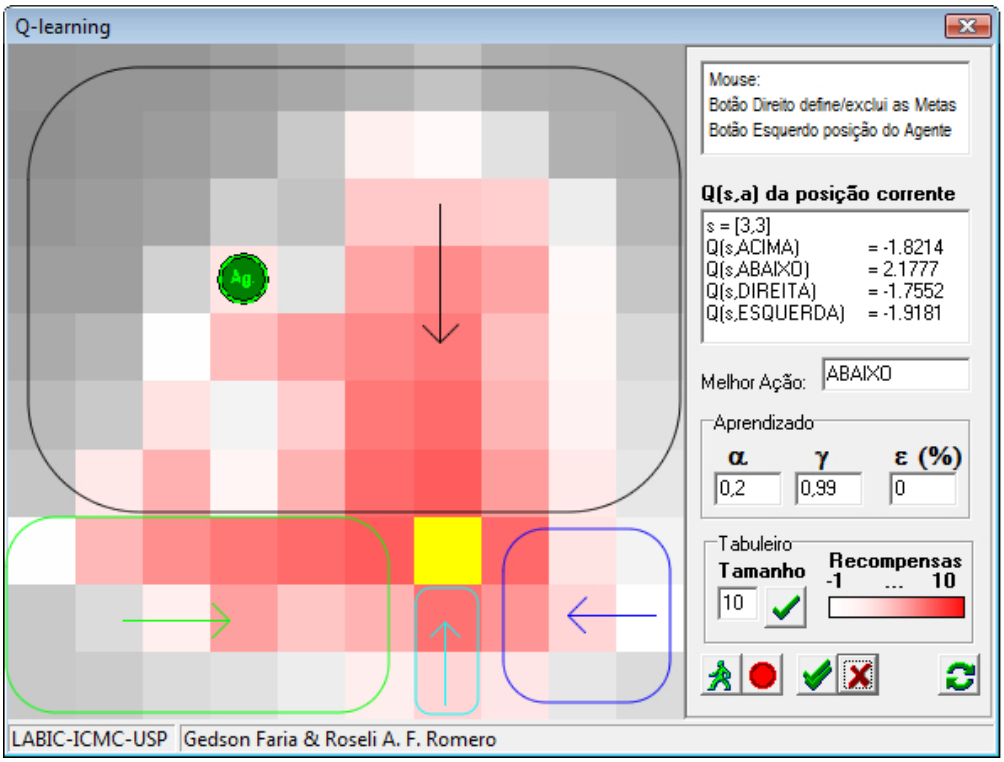

Figura 4.4: Agrupamento de estados associados a uma ação(seta) para o problema da locomoção.

Como o problema da atenção compartilha, detalhado na seção 5.4 do Capítulo 5, possui poucos estados relacionados a uma ação e considerado o fator precisão foi adotado uma simplificação do tratamento dos exemplos.

O algoritmo desenvolvido foi chamado de TG econômico (TGE) e consiste de algumas adaptações do algoritmo proposto em Silva (2008a). Detalhes do algoritmo também podem ser encontradas em Silva (2008b). Essas adaptações foram realizadas devido a algumas diferenças no ambiente onde o algoritmo de aprendizado foi inserido.

Um dos fatos que diferem os ambientes utilizados e que foi necesseraria uma alteração é o formato de execução. No simulador do Mundo dos Blocos, dado um estado inicial o agente interage com o ambiente até que o estado final seja atingido. No simulador de Interações Sociais,,o agente parte de um estado inicial e ao executar ações ele pode atingir o estado desejado e esse pode ser alterado ou continuar o mesmo.

O sistema de aprendizado inicia o seu funcionamento com a inicialização da função Q, que neste caso, vai fazer com que o sistema faça a exploração aleatória do ambiente, e crie uma árvore com um nó vazio.

O algoritmo obtém o estado corrente, no instante i, s_i e a necessidade do sistema n_i. Com essa informação o agente escolhe uma ação a_i segundo a política derivada da função Q e a executa modificando o ambiente s_{i+1} e gerando uma recompensa r_i. Os valores referentes ao estado, ação, Q e necessidade são enviados ao mecanismo de regressão, que se atualiza com as novas

informações. Cada uma dessas etapas são realizadas até que se atinja o número de iterações que foi pré-definido.

Todo esse processo está representado na Figura 4.5.

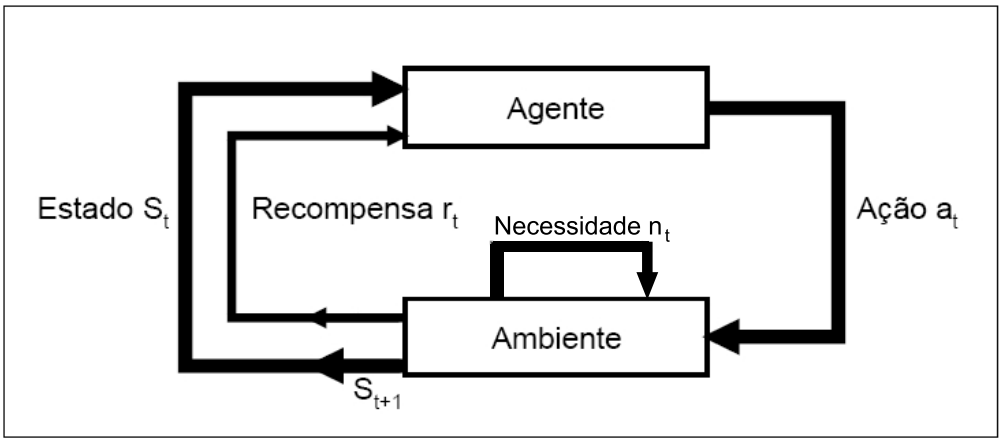

Figura 4.5: Interação entre o agente e o ambiente com a utilização da necessidade do agente.

A descrição do processo está expresso no Algoritmo 8.

Algoritmo 8 Algoritmo TGE

inicialize a função Q ($\hat{Q}_0$) e cria a árvore com um nó vazio
$i \leftarrow 0$
repita
  obtenha o estado s_i
  obtenha a necessidade do sistema n_i
  escolha uma ação a_i para um estado s_i usando uma politica derivada da função Q $\hat{Q}_i$
  execute a ação a_i, observe r_i e s_{i+1}
  Atualize $\hat{Q}_i$ usando $x = (s_i, a_i, \hat{q}_i, n_i)$ onde $\hat{q}_i \leftarrow r_i + \gamma \max_a \hat{Q}_e(s_{j+1}, a)$ e um algoritmo de regressão relacional para produzir $\hat{Q}_{i+1}$ {Utilizando o algoritmo 9}
  $i \leftarrow i + 1$
até que não há mais episodíos

O mecanismo de regressão recebe o conjunto (estado, ação, Q, necessidade) do sistema TGE. É realizado testes no nó interno com o valor do estado para verificar a sua existência. Em caso positivo, o Q existente no nó folha referente a ação é atualizado, caso contrario, o estado é inserido na árvore e a ação com seu Q é inserido no nó folha.

No nó folha podem existir mais de uma ação e para um acesso fácil á melhor ação, estas são ordenadas segundo os seus valores Q. Este processo é realizado quando um exemplo é inserido ou atualizado. Todo o processo do mecanismo de regressão é demonstrado no Algoritmo 9.

É evidente que a árvore não utiliza as estatísticas para escolha do melhor elemento e divisão dos nós para o crescimento da árvore, com isso a árvore construída se torna bem maior. Um outro ponto é o fato de existir mais de uma ação no nó folha, o que não ocorre no TG.

Algoritmo 9 Algoritmo de árvore relacional TGE

repita
distribuir todo exemplo sobre a estrutura da árvore do nó raiz ao nó folha usando os testes nos nós internos
se encontrou um nó folha **então**
atualiza o Valor Q para a ação executada no nó folha
senão se encontrou uma posição vazia **então**
gera um novo nó
fim se
até que o exemplo esteja representado do nó raiz ao nó folha
se necessario **então**
ordenar ações de acordo com o valor Q
fim se

4.3 Considerações Finais

Nesse Capítulo foi apresentado o Algoritmo TGE, que consiste de um algoritmo de aprendizado por reforço com árvore de de regrssão utilizado uma representação relacional do ambiente. Ele se diferencia do algoritmo TGE por não utilizar uma estrutura auxiliar para armazenar os exemplos antes de inserir na árvore de regressão. O algoritmo TGE também não utiliza os cálculos das estatísticas e nem os operadores de refinamento na estrutura da árvore.

CAPÍTULO

5

Arquitetura de Aprendizado para uma Cabeça Robótica

Este capítulo aborda uma arquitetura que foi desenvolvida para o aprendizado e interação com uma cabeça robótica, uma vez que o algoritmo proposto foi testado visando o aprendizado desta cabeça. Essa arquitetura foi projetada para ser aplicada a robótica social, na qual é feito uma breve descrição. Além disso, o Capítulo apresenta o simulador de Interações Sociais e a cabeça robótica. No final são feitas as considerações finais do Capítulo.

5.1 Introdução

O desenvolvimento de técnicas de aprendizado busca criar sistemas para solucionar problemas com diversidades de aspectos, e ao mesmo tempo adquirir conhecimento da interação com o ambiente. Entre os vários estudos na área da robótica um deles visa a construção de robôs, que possam interagir com os seres humanos estabelecendo uma interação social e que esses possam apreender com isso (Policastro, 2008).

Um robô que co-existe diariamente com pessoas deve ser capaz de aprender e se adaptar a novas experiências. Idealmente, as pessoas poderam ensinar ao robô como executar novas tarefas ou detalhes de como executar uma determinada tarefa. Conseqüentemente, um desafio fundamental é projetar robôs que possam ser ensinados da mesma maneira que outra pessoa ou da forma mais semelhante possível (Dautenhahn, 1995; Breazeal, 2002, 2004; Policastro, 2008).

Entretanto, existem algumas questões a serem consideradas durante o projeto dos mecanismo de aprendizagem. Frente a grande complexidade e número de estímulos que chegam ao sistema

de percepção de um robô, este precisa decidir quais estímulos são relevantes para o seu aprendizado. A determinação de quais estímulos são relevantes ao processo de aprendizado pode ser entendido como uma questão de saliência (Itti et al., 1998; Breazeal, 2002, 2004). A saliência dos estímulos pode ser determinada internamente, por meio do processamento das propriedades dos estímulos (cor, tamanho, orientação, proximidade, entre outras), ou externamente, apontadas por um professor (Policastro, 2008).

Uma vez que os estímulos salientes foram determinados e identificados, o robô deve ser capaz de determinar qual a ação a ser tomada no presente contexto do ambiente (Breazeal, 2002, 2004). Conforme aumentam as funcionalidades do robô, aumenta seu repertório de possíveis ações. Este fato também contribui para um aumento no espaço de busca por ações apropriadas. A determinação de qual ação deve ser tomada pode ser efetuada de diversas maneiras. O robô pode selecionar ações e experimentá-las em um processo de aprendizado por contingência ou o robô pode selecionar suas ações baseado em suas experiências prévias. Na literatura foram propostos diversas abordagens de aprendizagem para robôs sociáveis (Marom e Hayes, 2001; Nagai et al., 2003; Lockerd e Breazeal, 2004; Gold e Scassellati, 2007; Policastro, 2008).

Lockerd e Breazeal (Lockerd e Breazeal, 2004), por exemplo, apresentam um mecanismo de aprendizagem, implementado em um robô humanóide, para demonstrar que um diálogo colaborativo pode permitir a um robô aprender uma tarefa por meio da tutelagem de um ser humano. O robô possui sistemas de visão e reconhecimento de fala para permitir a interação multimodal com o mesmo. O sistema cognitivo recebe dados continuamente destes sistemas de visão e fala, e os integra para formar diversas convicções sobre objetos do mundo, assim como sobre gestos e fala dos seres humanos (Policastro, 2008).

5.2 Controle de Estímulos em Robôs Sociais

A robótica social é uma subárea da robótica na qual são exploradas abordagens e métodos para o desenvolvimento de robôs amigáveis aos seres humanos, denominados robôs sociais (Dautenhahn, 1998; Breazeal, 2002, 2003). Robôs sociais devem ser capazes de interagir, comunicar, entender e se relacionar com seres humanos de uma maneira natural (Breazeal, 2002). Esses robôs devem ser capazes de reconhecer os seres humanos e realizar interações sociais, além de possuir percepções e interpretar o ambiente no qual estão inseridos. Robôs sociais devem ser capazes de aprender a partir das interações com os seres humanos, adquirindo novos conhecimentos e adaptando seus comportamentos em resposta aos estímulos e ao contexto do ambiente (Dautenhahn, 1995).

Existem motivações científicas e práticas para o desenvolvimento de robôs sociais (Breazeal, 2002). Esses robôs podem ser utilizados para uma variedade de propósitos como: plataforma de pesquisas, educação e entretenimento. De uma perspectiva científica, pode-se aprender muito sobre a natureza humana com o processo de desenvolvimento de robôs sociais. Nesse contexto, os

robôs sociais podem ser utilizados como plataforma de teste, na qual modelos computacionais de habilidades sociais podem ser codificados, testados e analisados por meio do robô inserido em um ambiente social controlado. Esses modelos são meios pelos quais as hipóteses podem ser testadas para adequação e suficiência para explicar um conjunto de dados e realizar predições (Breazeal, 2002; Webb, 2000).

A incorporação de mecanismos evidenciados pela Análise do Comportamento pode conduzir o desenvolvimento de robôs capazes de interagir em ambientes sociais e de exibir comportamentos adequados que podem ocorrer a partir da interação com o ambiente social. A utilização de teorias como a Análise do Comportamento, mais especificamente, o controle de estímulos (Serio et al., 2004), pode conduzir ao desenvolvimento de robôs capazes de aprender a responder diferencialmente na presença de determinados estímulos do ambiente. Este responder diferenciado pode surgir a partir da interação social, caracterizando o aprendizado de robôs por meio de interações. Uma arquitetura robótica inspirada na Análise do Comportamento pode beneficiar a construção de robôs sociais interativos para diversas aplicações. Esta arquitetura formará uma ferramenta que possibilitará o projeto de robôs individuais altamente específicos, para diferentes tarefas, capazes de interagir socialmente com outros indivíduos (Policastro, 2008).

5.3 Arquitetura Inspirada na Análise do Comportamento

A arquitetura robótica na qual será inserido o algoritmo por nós proposto foi projetada por Policastro (2008). Ela é baseada nas teorias do Controle de Estímulos (Serio et al., 2004) que envolve o aprendizado em emitir respostas (ações) diferenciadas na presença de determinados contextos do ambiente (estímulos discriminativos) e será descrita a seguir. Existem três grandes módulos que podem ser identificados nesta arquitetura: módulo de percepção de estímulos, módulo de emissão de respostas e módulo de controle de consequências, conforme é representada na Figura 5.1.

Estes três módulos são evidenciados na relação fundamental da análise comportamental e do controle de estímulos ($S^d \rightarrow R \rightarrow S^r$), na qual S^d é o conjunto de estímulos discriminativos presente no ambiente no momento que um indivíduo emite uma resposta R e recebe com conseqüência um estímulo reforçador S^r. Esta relação fundamental é a base para a descrição de todos os comportamentos operantes emitidos pelos seres humanos e outros seres vivos (Policastro, 2008).

O módulo percepção de estímulos é composto por algoritmos de aquisição de dados a partir de sensores do ambiente e por um sistema de visão computacional. Esse módulo é responsável por codificar os estímulos percebidos no ambiente e por fornecer o conhecimento necessário sobre esses estímulos, na representação apropriada, de forma que este conhecimento possa ser utilizado pelo módulo de emissão de respostas e pelo módulo de controle de conseqüências. A arquitetura é embasada na técnicas de representação do conhecimento conhecida como representação relacional de primeira ordem, na qual suas vantagens já foram citadas no capítulo 3. Todos os estímulos definidos possuem diversas propriedades como forma, cor, tamanho, posicionamento em relação

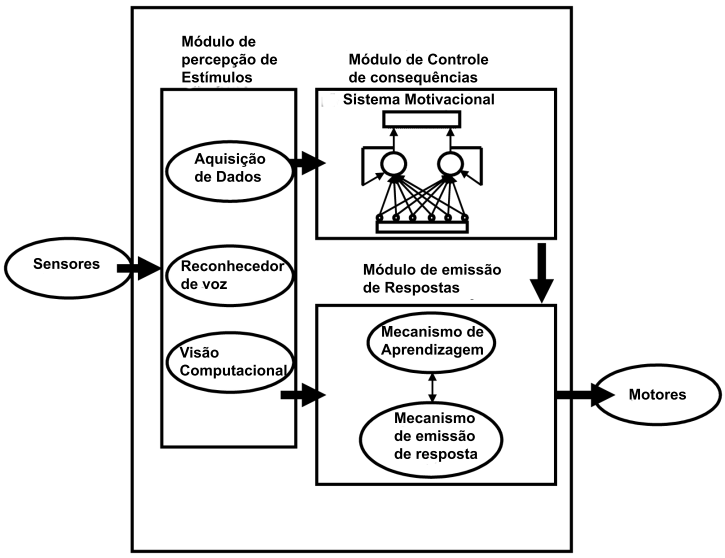

Figura 5.1: Organização geral da arquitetura. As setas indicam o fluxo de informações entre os três módulos da arquitetura. Os círculos indicam os métodos e estruturas componente dos módulos. O módulo de percepção de estímulos codifica os estímulos detectados no ambiente. Esses estímulos são então utilizados pelos módulos de controle de conseqüências e de emissão de respostas para aprender a exibir comportamentos apropriados.

ao foco do robô e, no caso de seres humano, a pose da cabeça. Os estímulos percebidos no ambiente deverão ser codificados em átomos no tipo: see(X), smell(X), hear(X), at(X), nos quais X representa um fato sobre um estímulo percebido no ambiente, como por exemplo *see(frontal(face))*, no qual *see()* é um átomo, *face* é um estímulo percebido e *frontal()* é um fato sobre *face*, definido de acordo com as propriedades deste estímulo (Policastro, 2008).

O módulo de emissão de respostas é composto por algoritmos de aprendizado e seleção de ações (respostas), que devem ser emitidas pelo agente, na presença de um conjunto estímulos. Esse módulo será composto por métodos e estruturas evidenciados na Análise do Comportamento (Serio et al., 2004), empregando um algoritmo não determinístico de aprendizado por reforço (Sutton e Barto, 1998), conhecido por aprendizado de contingência, empregando uma representação relacional de primeira ordem para o conhecimento gerado durante as interações socials (Driessens, 2004; Otterlo, 2005). O funcionamento desse módulo é descrito a seguir.

Inicialmente, o sistema de emissão de resposta emite respostas, estocasticamente, na presença dos estímulos pecebidos no ambiente. Quando uma determinada resposta, emitida na presença de um conjunto de estímulos apresenta efeito de reforço positivo no sistema de controle de conseqüência, é criada uma regra, empregando-se a representação relacional de primeira ordem, na qual a parte antecedente representa os estímulos presentes no ambiente e a parte conseqüente representa a resposta emitida pelo agente. Esta regra recebe um valor de aptidão, que é transformado na probabilidade desta ser selecionada na presença dos estímulos existentes no ambiente. Este valor de aptidão é criado em conformidade com a potência do estímulo reforçador percebido como conse-

qüência da resposta emitida pelo agente e é incrementado sempre que a regra é selecionada e sua execução resulta em um novo reforçamento.

A seleção de ações a serem executadas (respostas a serem emitidas) pelo sistema de emissão de respostas segue uma política estocástica. Esta política foi adotada devido aos mecanismos evidenciados no Controle de Estímulo, no qual um estímulo reforçador aumenta a probabilidade de uma resposta ser emitida na presença de um estímulo discriminativo e uma punição diminui a probabilidade da emissão de uma resposta, constituindo-se, portanto, uma política estocástica de seleção de ações. Adicionalmente, o sistema de emissão de respostas deve realizar um balanço entre a seleção de relações já existentes e a exploração de novas respostas que podem levar ao surgimento de novas regras entre estímulos, respostas e conseqüências. Após um número de interações com o ambiente, um processo de generalização deverá ser empregado para tentar generalizar o conjunto de regras empregado, por meio de algoritmos de regressão relacional (Driessens, 2004). Esse processo é importante para a generalização do conhecimento adquirido durante o processo de aprendizado.

Os detalhes do funcionamento desse algoritmo, que é conhecido como aprendizagem de contingência é detalhado em Policastro (2008).

O módulo de controle de conseqüências é composto por um mecanismo que simula as necessidades de um indivíduo, baseado no sistema apresentado proposto em Breazeal (2002). Esse mecanismo é constituído de unidades de necessidade que são implementadas de maneira similar a um *perceptron* simples com uma conexão recorrente (Haykin, 1999). A função de ativação da unidade de necessidade é dada por: $y = \sum w_j.i_j$, na qual i_j são sinais de entrada que indicam a presença ou não das propriedades dos estímulos percebidos no ambiente e w_j são os pesos (que podem ser positivos e negativos) pré-definidos de acordo com a unidade de processamento. Essas unidades de necessidade simulam a homeostase dos organismos vivos. Elas possuem um ponto de equilíbrio e variam entre um valor máximo e mínimo, empiricamente determinados. Um valor positivo em uma unidade de necessidade, acima do ponto de equilíbrio, indica a privação do robô a um determinado estímulo reforçador. Por exemplo, na ausência de um estímulo no ambiente que indica a presença de uma pessoa (cor de pele), uma unidade que simule a necessidade de interagir com pessoas por ter seu valor aumentado no tempo, levando o robô a precisar estabelecer contato e interagir com um ser humano. Uma vez que o robô estabeleça contado, essa unidade tem seu valor de ativação reduzido gradativamente até o ponto de equilíbrio.

Quando um estímulo é percebido pelo módulo de percepção de estímulos, este é decomposto e suas propriedades são utilizadas pelo módulo de seleção de respostas e pelo módulo de controle de conseqüências. Nesse último, as propriedades são empregadas para o cálculo do valor de ativação das unidades de necessidade. Quando uma unidade tem seu valor de ativação diminuído em direção ao ponto de equilíbrio, as propriedades empregadas no cálculo da ativação são consideradas como reforçadores para o robô e, então, uma regra de comportamento é criada, ou uma regra existente tem sua probabilidade de seleção aumentada, pelo mecanismo de aprendizado. A magnitude da variação do valor de ativação da unidade de necessidade é utilizada pelo módulo de controle de conseqüências para definir a potência do reforço ou da punição associada a uma resposta emitida.

Dessa forma, a arquitetura fornece mecanismos para simular estados de privação e satisfação de necessidades, permitindo que uma resposta emitida tenha uma conseqüência reforçadora ou punitiva e permitindo que o algoritmo de aprendizado possa receber informações sobre as conseqüências das respostas por ele selecionadas na presença de determinados estímulos discriminativos.

A troca de informação entre os três módulos principais da arquitetura é feita por uma memória de trabalho. Esta memória é usada para manter informações sobre estímulos, últimas respostas emitidas, necessidades internas ativas e reforços detectados. Cada elemento inserido na memória de trabalho possui um contador que confere a noção de tempo. Quando um novo elemento é inserido na memória, seu contador de tempo é inicializado com valor igual a zero. Adicionalmente, este contador é incrementado de 1 sempre que novos elementos são inseridos subseqüentemente na memória. Assim, os elementos persistem por vários passos de tempo na memória.

Esta arquitetura robótica foi testada em um simulador de interação social que é explicado neste Capítulo, projetada sobre uma importante habilidade social: a atenção compartilhada (Dube et al., 2004). Na área da robótica social, a atenção compartilhada é um dos grandes desafios a ser solucionados pelos pesquisadores. Esta habilidade é associada a uma situação, na qual dois agentes encontram-se olhando para um mesmo objeto ou se olhando mutuamente. Ela possibilita o aprendizado do que é importante no ambiente. Diversas teorias enfatizam o papel da atenção compartilhada como fundamental para a interação e aprendizado social como pode ser visto na próxima seção.

5.4 Atenção compartilhada

A atenção compartilhada é uma habilidade fundamental na interação e na cognição sociais humanas. É definido como atenção re-orientada e re-alocada a um alvo porque é o objeto da atenção de uma outra pessoa (Deák et al., 2001). Ela pode ser definida também como a capacidade de usar gestos e contatos visuais com atenção coordenada com outro agente para dividir experiências de objetos ou eventos interessantes no ambiente (Dube et al., 2004; Kaplan e Hafner, 2004). A atenção visual compartilhada é associada à situação na qual dois agentes estão olhando um para o outro, um agente direciona o seu olhar para um objeto presente no ambiente e, o segundo agente segue o olhar ao objeto correto. Esta habilidade torna possível o aprendizado do que é importante no ambiente (Deák e Triesch, 2005b).

A atenção compartilhada é mais do que o olhar que segue em frente. É o uso de sugestões sociais para guiar a atenção ao ambiente, e a monitoração de outra para determinar se estão compartilhando de uma experiência e o que capturou seu interesse (Deák et al., 2001).

Dube e seus colegas (Dube et al., 2004) apresentaram uma análise para explicar as origens de atenção compartilhada. Esta análise recorre a situações nas quais uma criança inicia a seção de atenção compartilhada, direcionando a atenção de um adulto para um objeto no ambiente e obtendo a atenção deste adulto durante a interação com o objeto. Porém, esta análise pode ser

estendida para explicar sessões de atenção compartilhada iniciadas também por um adulto. A análise caracteriza o início de um evento interessante em um contexto do ambiente que inclui a presença de um adulto familiar como Operador Motivacional.

Um Operador Motivacional é um evento que muda o estado do ambiente e reforça uma resposta de um indivíduo. Um comportamento torna-se mais freqüente se o Operador Motivacional estabelece um reforço, ou fica menos freqüente se o Operador Motivacional inibe um reforço ou estabelece um castigo. O efeito da mudança de comportamento de um Operador Motivacional pode ser visto como a mudança positiva ou negativa da freqüência do comportamento relevante aos eventos conseqüentes e pode depender da presença de estímulos discriminativos apropriados no ambiente.

Em uma sessão de atenção compartilhada, um evento interessante estabelece no ambiente a capacidade de reforço de uma classe de estímulos denominados estímulos de atenção de um adulto. Estes estímulos são percepções visuais e auditivas indicativas da atenção dos adultos a um objeto ou evento de interesse. As respostas de um adulto tornam-se efetivas como reforçador condicional após uma história de reforçamento: os estímulos discriminam que o adulto reagirá a um evento interessante e a reação do adulto é relacionada a um aumento da probabilidade de acesso a reforçadores. Então, o comportamento da criança de seguir o olhar do adulto é resultado de uma história de reforçamento na qual a atenção do adulto torna-se o estímulo reforçador. Ainda, a atenção dos adultos age como uma ponte para uma cadeia de comportamentos (Dube et al., 2004; Policastro, 2008).

5.5 Simulador de Interações Sociais

Antes de inserir a arquitetura na cabeça robótica foi desenvolvido um simulador de interações sociais, capaz de simular os movimentos necessários ao robô e ao ser humano, além de fornecer estímulos do ambiente apropriados ao contexto da atenção compartilhada. Esse simulador foi baseado em trabalhos de Triesch e colegas (Triesch et al., 2006). Os detalhes sobre o simulador são apresentados na próxima seção.

5.5.1 Simulador de Interações Sociais

O simulador de interações sociais desenvolvido para os experimentos da atenção compartilhada é capaz de simular uma interação entre um robô e um ser humano em um ambiente social controlado.

Na Figura 5.2 é apresentada a interface do simulador. No lado esquerdo da interface está o painel de controle que habilita as simulações interativas ou automáticas. O ser humano é fixo na porção superior da interface e é capaz de girar a sua cabeça em um ângulo de $\pm$ 90 graus. O robô é fixo na porção inferior da interface e também é capaz de girar a sua cabeça em um ângulo de $\pm$ 90 graus.

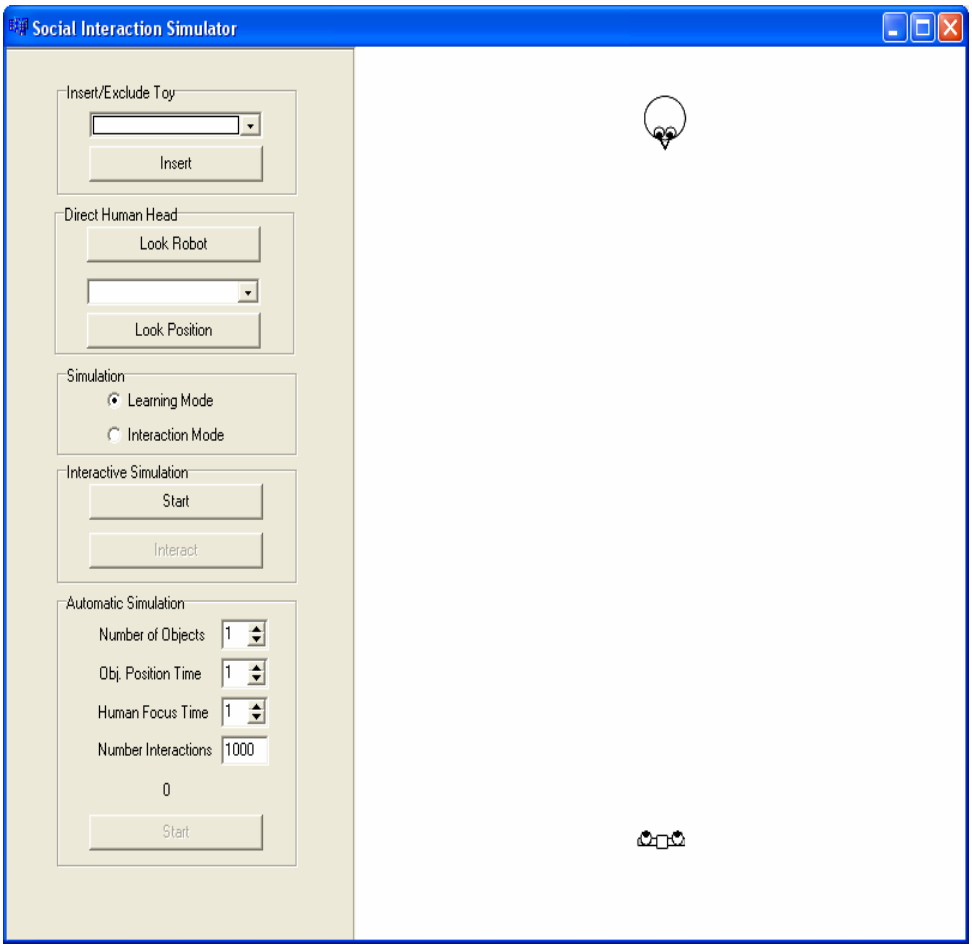

Figura 5.2: Interface do simulador de interações sociais.

Para simular a atenção compartilhada, foram definidas três entidades que podem ser manipuladas por funções do simulador, um humano, um robô e um brinquedo. Neste simulador, o ser humano e o robô foram posicionados frente a frente a uma distância fixa. Além do ser humano e do robô, o simulador possibilita posicionar até dois brinquedos simultaneamente no ambiente social. Esta funcionalidade é útil para se simular um objeto distrator enquanto se posiciona um objeto que é foco de atenção do ser humano, permitindo verificar se o robô olha para o brinquedo correto, mesmo na presença de outros objetos no ambiente. Um brinquedo pode ser posicionado em qualquer lugar vazio do ambiente social, a qualquer momento durante uma simulação.

O ambiente social foi modelado da seguinte maneira. Tanto o robô como o ser humano podem girar suas cabeças para a esquerda ou para a direita, em um ângulo de até $90°$. O robô tem seu foco central em $0°$ e tem seu campo visual limitado por um parâmetro de fóvea $\vartheta°$ permitindo que o robô visualize objetos em um campo visual formado por um cone de abertura dada por: [-$\vartheta°$, +$\vartheta°$], com centro em $0°$. Na Figura 5.3 é ilustrada esta modelagem do campo visual do robô.

A posição da cabeça do robô é determinada por θ_r, que pode assumir valores entre $[-90°, +90°]$. A posição da cabeça do ser humano é determinada por θ_a, que também pode assumir valores entre $[-90°, +90°]$. Quando um objeto i é posicionado no ambiente social, o simulador traça o ân-

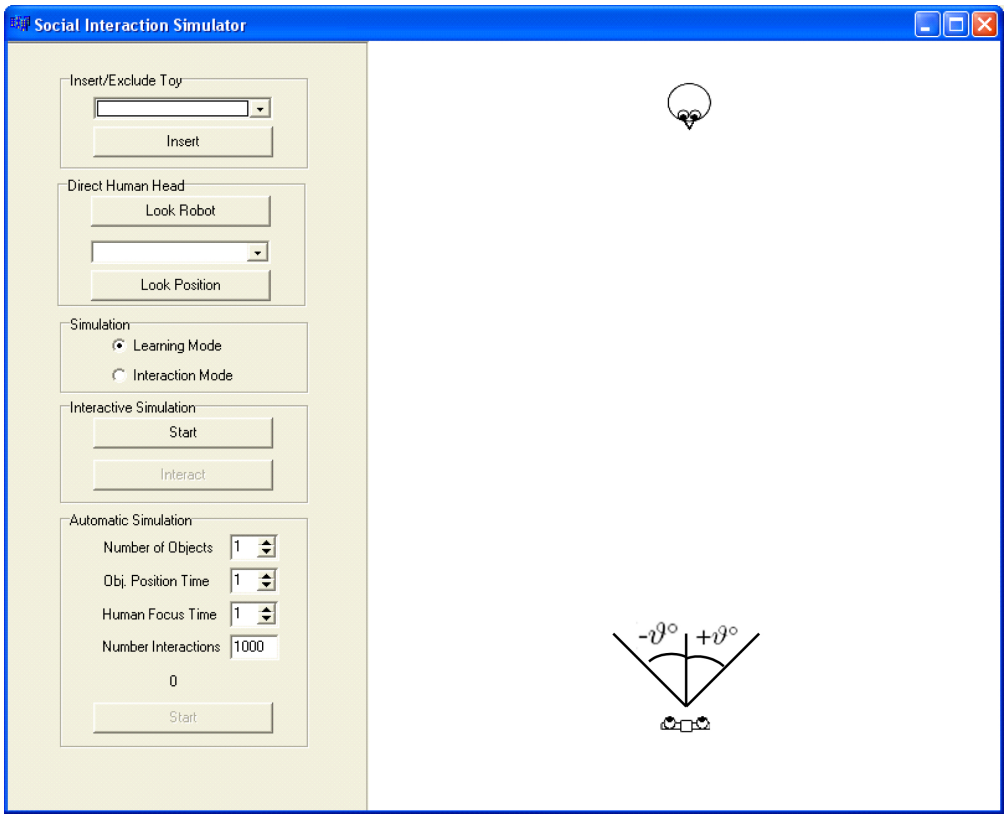

Figura 5.3: Campo visual do robô. As linhas representam os limites do campo visual do robô, com abertura dada por: [-$\vartheta°$, +$\vartheta°$], com centro em $0°$.

gulo entre este objeto e o foco do robô, ou seja, ele determina o deslocamento da cabeça do robô necessário para focalizar o objeto posicionado no ambiente. Esta mapeamento é dado por θ_{oi}, que pode assumir valores entre $[-90°, +90°]$. Desta maneira, se um objeto é posicionado no ambiente, o simulador verifica se o mesmo está dentro do campo visual do robô, comparando sua posição em relação ao foco da visão do robô, considerando o campo de visão do mesmo.

Na Figura 5.4 são ilustrados os parâmetros de posicionamento dos objetos e do ser humano em relação ao campo visual do robô. As linhas representam as distâncias entre o foco do robô e os objetos posicionados no ambiente social, assim como a posição da cabeça do robô e da cabeça do ser humano. Nesta figura é mostrada uma interação na qual o ser humano está olhando para um objeto posicionado no ambiente e o robô acompanha seu olhar para o objeto correto, apesar da existência de um objeto distrator.

Adicionalmente aos estímulos visuais da face do ser humano e dos brinquedos, o simulador provê um estímulo auditivo que simula a atenção do ser humano para com o robô. O simulador provê este estímulo quando o humano e o robô estão mantendo contato ocular e quando o robô segue o olhar do ser humano até um objeto correto, que é foco de atenção deste último. Este mecanismo foi incorporado no simulador para simular os resultados da análise comportamental apresentada por Dube e seus colegas (Dube et al., 2004), na qual eles argumentam que os adultos agem como operadores motivacionais no contexto da aprendizagem de atenção compartilhada,

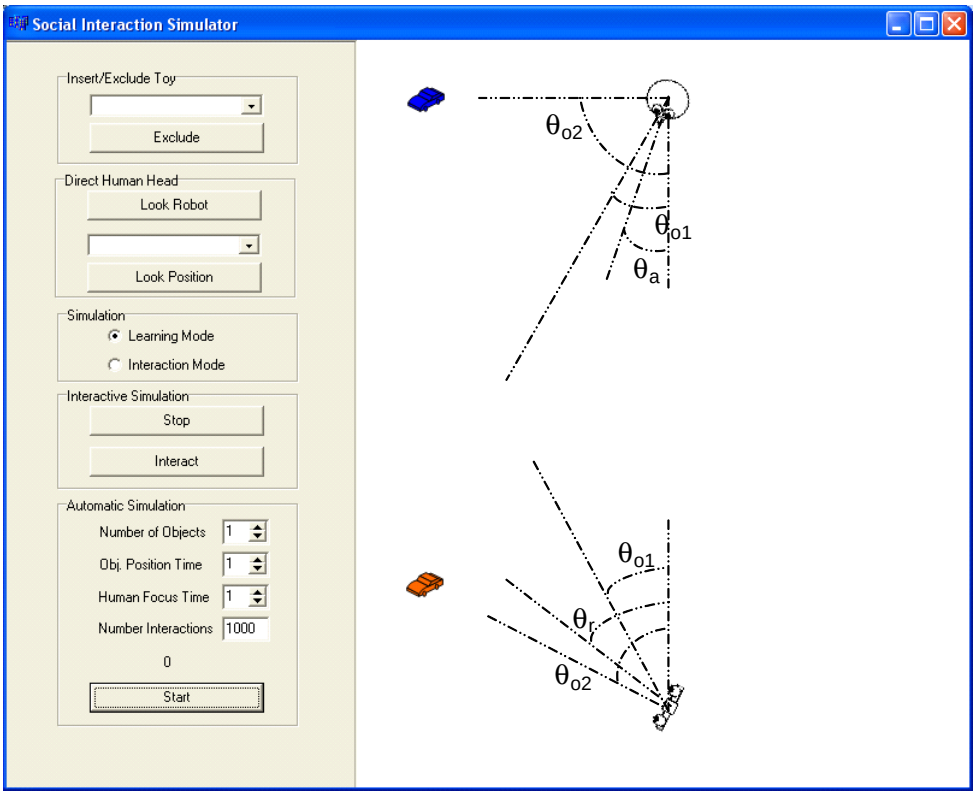

Figura 5.4: Controle de posicionamento.

fornecendo de uma classes de estímulos reforçadores denominados de atenção do ser humano, como apresentado na Seção 5.4 do Capítulo 5.

Durante uma simulação, o simulador é capaz de executar interações continuamente e cada interação toma aproximadamente 1 segundo. O simulador pode posicionar até dois objetos simultâneos no ambiente social, em posições estocasticamente selecionadas com probabilidade ρ_o. Estes objetos são posicionados nos respectivos lugares durante um tempo determinado pelo usuário (determinado em segundos ou cíclos de interações no painel de controle). Adicionalmente, o simulador pode direcionar a cabeça do ser humano para focar um objeto presente no ambiente ou para focar o robô. O objeto que recebe o foco do olhar do ser humano é estocasticamente selecionado com probabilidade ρ_a. Após focar um objeto, o humano permanece com o seu foco no mesmo durante um tempo determinado pelo usuário (determinado em segundos ou cíclos de interações no painel de controle), antes de direcionar a sua cabeça para focar outro objeto ou para focar o robô.

5.6 Cabeça Robótica Interativa

Os experimentos de interações sociais reais da arquitetura com a utilização do algoritmo proposto neste trabalho foram realizados empregando-se uma cabeça robótica interativa conectada por cabo a um computador. Esta cabeça robótica á composta por 5 servo motores, uma câmera digital colorida, um módulo multimídia e um módulo controlador e é apresentada na Figura 5.5.

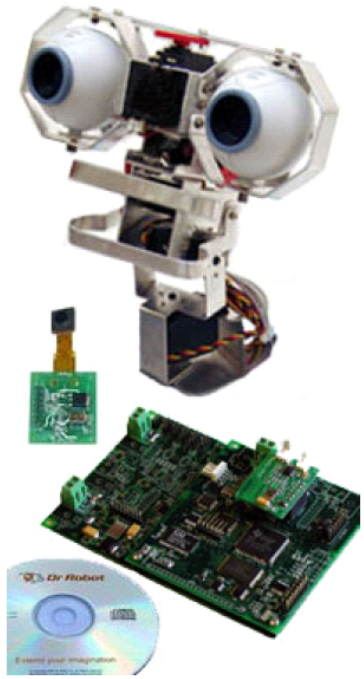

Figura 5.5: WHA8030 da Dr. Robot.

Esta cabeça robótica possui um kit de desenvolvimento de software (SDK) composto por um controle Active X contendo uma série de funções da interface de programação API do controlador do robô, que pode ser adicionado em programas desenvolvidos em Visual C++. Para o desenvolvimento de um sistema de controle, deve-se então adicionar este Active X em um projeto de software, no ambiente de programação IDE do Visual C++ e acessar as rotinas de controle disponibilizadas por este componente de controle.

Existem diversas funções de controle disponibilizadas no SDK do robô e estas funções são agrupadas nas categorias de Sensores Periféricos, Controle Motor, Controle Multimídia e Eventos. Entretanto, devido a versão do robô somente as funções de controle dos servo motores foram empregadas neste projeto de pesquisa. A seguir são apresentadas as funções de controle empregadas no desenvolvimento do sistema de controle do robô.

- EnableServo(short channel) - liga o canal (servo motor) especificado de controle.
- DisableServo(short channel) - desliga o canal (servo motor) especificado de controle.
- ServoTimeCtr(short channel, short cmdValue, short timePeriod) - envia um comando de controle de movimento ao canal do controle do servo motor especificado (channel). Este comando especifica que o servo motor deve atingir uma determinada posição (cmdValue) em um determinado tempo (timePeriod).
- void ServoNonTimeCtr(short channel, short cmdValue) - envia um comando de controle de movimento ao canal do controle servo do motor especificado (channel). Este comando

especifica que o servo motor deve atingir uma determinada posição (cmdValue), sem determinar tempo de ação.

- ServoTimeCtrAll(short cmd1, short cmd2, short cmd3, short cmd4, short cmd5, short cmd6, short timePeriod) - envia simultaneamente um comando de controle de movimento para todos os canais do controle dos servo motores. Este comando especifica que os servo motores devem atingir determinadas posições (cmd1...cmd6) em um determinado tempo (timePeriod).

- ServoNonTimeCtrAll (short cmd1, short cmd2, short cmd3, short cmd4, short cmd5, short cmd6) - envia simultaneamente um comando de controle de movimento para todos os canais do controle dos servo motores. Este comando especifica que os servo motores devem atingir determinadas posições (cmd1...cmd6), sem determinar tempo de ação.

As funções acima descritas, disponibilizadas no SDK do robô, foram empregadas para o desenvolvimento de diversas rotinas motoras que foram utilizadas pelo sistema de controle e pela arquitetura proposta durante os experimentos de interação com o robô em um ambiente social real e controlado.

Para a execução destes experimentos com a cabeça robótica interativa, foi desenvolvido um sistema capaz de controlar as interações entre o robô e um ser humano em um ambiente social controlado. Este sistema integra um sistema de visão computacional, um sistema de reconhecimento de fala, um sistema motor desenvolvido para a cabeça robótica interativa e a arquitetura robótica proposta neste trabalho. Adicionalmente, para demonstrar o valor do aprendizado da atenção compartilhada como precursor do aprendizado por meio de interações sociais, foi desenvolvido um mecanismo capaz de simular o aprendizado de conceitos sobre objetos do mundo real apresentados ao robô, por meio da tutelagem de um ser humano. Na Figura 5.6, é apresentada a arquitetura geral do sistema de controle do robô, ilustrando as interações entre todos os módulos. A seguir, são apresentados os módulos deste sistema.

5.6.1 Sistema de Visão Computacional

O sistema de visão é composto por um módulo de reconhecimento de face capaz de estimar a pose da cabeça de um ser humano, baseado em modelos adaptativos de visão computacional baseados em aparência (Morency et al., 2003). O sistema de visão também é composto por um módulo de detecção de objetos, baseado em modelos de saliência e atenção visual (Itti et al., 1998).

O sistema de visão é baseado no trabalho apresentado por Breazeal e Scassellati (1999) e é capaz de simular algumas preferências visuais de crianças entre os 6 e os 18 meses de idade, como cores salientes e faces humanas. A implementação do sistema de visão é baseado em mapas de características processados para cada percepção (cores e faces). O resultado do processo deste sistema consiste em um mapa de ativação que pode ser usado pelos outros módulos da arquitetura de controle para controlar o comportamento do robô.

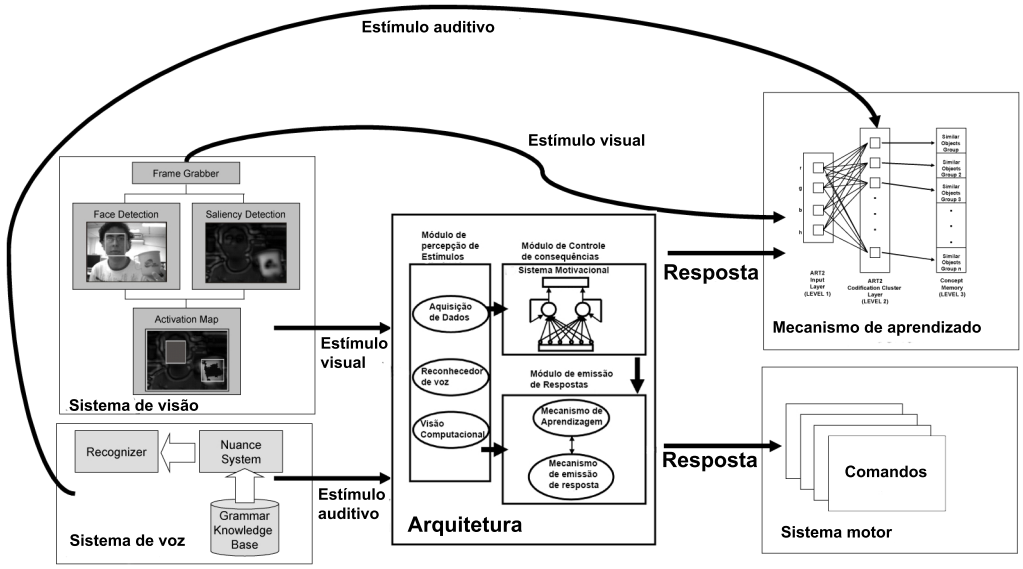

Figura 5.6: Arquitetura geral do sistema de controle do robô. As setas indicam o fluxo de informações entre os módulo do sistema de controle. Este sistema emprega os diversos módulos desenvolvidos durante este projeto de pesquisa: sistema de visão, sistema de voz, mecanismo de aprendizagem por tutelagem, sistema motor e arquitetura robótica.

O mapa de cores é baseado no trabalho de busca e atenção visual apresentado por Itti et al. (1998). Este processo utiliza um mecanismo biologicamente inspirado de atenção visual para criar um mapa de características que representa a *saliência visual* de uma cena. Esta saliência visual é formada pela composição de vários mapas de características da imagem. Este mapa de saliência foi desenvolvido utilizando-se as funções de saliência da Lti-Lib (Lib, 2003).

O mapa de faces é baseado nos trabalhos apresentados por Morency et al. (2003); Morency (2007), sobre face e detecção de poses. A detecção das faces é executada empregando-se uma abordagem de modelos adaptativos baseados em aparência (Morency et al., 2003; Morency, 2007). Este mapa de face foi desenvolvido empregando-se as funções de detecção de face e pose da Watson (Morency, 2007).

Depois de processar estes mapas de características (saliência e face), estes são utilizados para construir um mapa de ativação que representa os estímulos detectados pelo sistema de visão. O mapa de saliência é processado empregando-se um limiar de saliência e um limiar de raio mínimo de área para selecionar uma região de interesse a partir das saliências apontadas no mapa. Este processo é baseado no trabalho apresentado por Rodrigues e Gomes (2002). Então, um processamento baseado em histograma de cores é executado para se obter os valores mais freqüentes dos canais r, g, b (do espaço de cores RGB) e h (do espaço de cores HSI), dentro da região de interesse. O mapa de face é processado, empregando-se as funções da biblioteca Watson, para se obter a posição (x e y) das faces detectadas e a pose destas faces (ângulos pan e tilt). Posteriormente, os atributos de cores de objetos salientes e pose das faces são codificados e propagados para outros módulos da

arquitetura para controlar o comportamento do robô. Na Figura 5.7, é ilustrada a arquitetura geral do sistema de visão computacional desenvolvido.

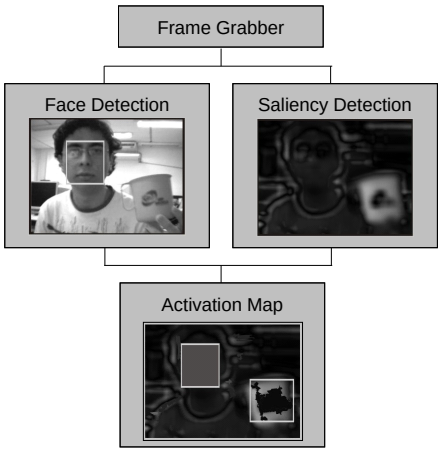

Figura 5.7: Sistema de visão computacional. *Face Detection* ilustra o mapa de faces e *Saliency Detection* ilustra o mapa de cores. *Activation Map* ilustra o mapa de ativação resultante da combinação dos mapas processados pelo sistema de visão.

5.6.2 Sistema de Voz

O sistema de voz é composto por um sistema de reconhecimento de fala e por um sistema de vocalização. O sistema de reconhecimento de fala é capaz de reconhecer a linguagem natural falada no idioma Português Brasileiro e é baseado no Sistema Nuance (Nuance, 2001).

Este sistema contém uma máquina de reconhecimento de fala e uma base de conhecimento gramatical. Adicionalmente, existe um módulo reconhecedor implementado em linguagem Java, por permitir um fácil interfaceamento com o sistema $Nuance^{TM}$. Este módulo reconhecedor recebe a codificação da fala, relativa à gramática configurada na base de conhecimento e a envia para o sistema de controle da cabeça interativa por meio de uma porta *socket*.

O sistema de vocalização, nesta versão do sistema de voz, utiliza diversos arquivos *.wav* gravados com palavras individuais previstas no vocabulário o robô. Desta forma, o sistema de voz pode montar uma frase desejada para que o robô a vocalize, pela união destas palavras individuais. Este sistema habilita sessões de conversações curtas com o robô, suficientes para os propósitos dos experimentos realizados neste trabalho de pesquisa. Na Figura 5.8, é ilustrada a arquitetura geral do sistema de reconhecimento de fala.

5.6.3 Sistema Motor

O sistema motor da cabeça robótica interativa foi implementado empregando-se as funções disponibilizadas no SDK do robô. Este sistema motor possui scripts com diversos comandos temporizados de motor, que permitem ao robô emitir seis comportamentos diferentes:

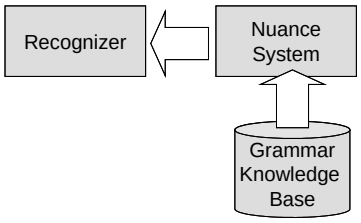

Figura 5.8: Sistema de reconhecimento de fala. O sistema $Nuance^{TM}$ (*Nuance System*) utiliza a base de conhecimento gramatical (*Grammar Knowledge Base*) para reconhecer uma fala detectada no ambiente. Este módulo é responsável por enviar a codificação de uma fala reconhecida ao módulo reconhecedor (*Recognizer*) implementado em $Java^{TM}$, que por sua vez disponibiliza esta codificação para o sistema de controle da cabeça robótica por meio de uma porta de *socket*.

1. Procurar por um ser humano, no qual o robô parte da sua posição atual e posiciona a sua cabeça de forma centralizada e ereta, para que este enxergue uma área imediatamente à sua frente, na linha do horizonte.

2. Procurar por um objeto a frente e abaixo, no qual o robô parte da sua posição atual e posiciona a sua cabeça de forma centralizada e abaixada, para que este enxergue uma área imediatamente a sua frente, mas abaixo do horizonte.

3. Procurar por um objeto a esquerda, no qual o robô parte da sua posição atual e posiciona a sua cabeça de forma ereta, mas virada para a sua esquerda, para que este enxergue uma área à esquerda, na linha do horizonte.

4. Procurar por um objeto a direita, no qual o robô parte da sua posição atual e posiciona a sua cabeça de forma ereta, mas virada para a sua direita, para que este enxergue uma área à direita, na linha do horizonte.

5. Procurar por um objeto a esquerda e abaixo, no qual o robô parte da sua posição atual e posiciona a sua cabeça virada para a sua esquerda e abaixada, para que este enxergue uma área à esquerda e abaixo do horizonte.

6. Procurar por um objeto a direita e abaixo, no qual o robô parte da sua posição atual e posiciona a sua cabeça virada para a sua direita e abaixada, para que este enxergue uma área à direita e abaixo do horizonte.

Estes seis scripts de comandos motores permitem a emissão dos comportamentos necessários aos experimentos reais de interação no contexto do aprendizado da atenção compartilhada e do aprendizado por tutelagem.

5.6.4 Sistema de Controle Integrado

O sistema de controle da cabeça robótica emprega todos os sistemas e módulos componentes apresentados anteriormente para executar um ciclo de controle que possibilita a interação do robô com o ambiente.

Na Figura 5.9 é apresentado o algoritmo de controle integrado. Algoritmo principal do sistema de controle da cabeça robótica interativa. Este algoritmo integra os sistemas de visão e voz para obter os estímulos visuais (s_v) e auditivos (s_a), a arquitetura de controle proposta neste trabalho de pesquisa e o sistema motor do robô. Após obter estímulos presentes no ambiente, o sistema de controle os envia para a arquitetura robótica, que indica qual resposta (a) deve ser emitida pelo robô. Então, o sistema de controle chama a rotina motora apropriada e emite a resposta designada pela arquitetura.

```
function RobotMainControl ()
 do forever
    Get  the visual environment state s_v  from the vision system
    Get  the auditory environment state s_a from the auditory system
    Send the environment state s_v e s_a to tue robotic architecture
    Get the response a, to be emitted by the robot, from the architecture
    Emit the response a, calling the corresponding motor script
end do
end function
```

Figura 5.9: Algoritmo principal do sistema de controle da cabeça robótica interativa.

Em cada iteração do ciclo de controle, o sistema codifica os estímulos visuais e auditivos dos sistemas de visão e reconhecimento de fala, respectivamente. Então, o sistema de controle envia os estímulos codificados para a arquitetura robótica que, por sua vez, executa uma iteração de aprendizado e indica uma resposta a ser emitida como conseqüência desta iteração. Posteriormente, o sistema de controle chama a rotina motora apropriada e emite a resposta designada pela arquitetura robótica, completando um ciclo de iteração com o ambiente.

5.6.5 Mecanismo de Aprendizagem por Tutelagem

O mecanismo de aprendizagem por tutelagem proposto neste trabalho de pesquisa é capaz de associar estímulos visuais e auditivos para simular a aprendizagem de conceitos sobre objeto do mundo real por meio da tutelagem de um ser humano.

Este mecanismo utiliza o sistema de visão para extrair as características visuais de um determinado objeto, composto pelos valores mais freqüentes dos canais r, g, e b (do espaço de cores RGB), e do canal h (do espaço de cores HSI). O mecanismo de aprendizagem também utiliza o sistema de voz para adquirir o nome falado do objeto apresentado ao robô. Estas características são

então aprendidas e organizadas empregando uma rede neural do tipo ART2 (Carpenter e Grossberg, 1987) e uma memória plana que armazena as características visuais juntamente com o nome falado do objeto para formar novos conceitos. O mecanismo de aprendizagem contém três níveis de organização de memória, como ilustrado na Figura 5.10:

1. O primeiro nível (LEVEL 1 na Figura 5.10) é composto pela camada de entrada da rede neural ART2. Esta camada contém quatro nós de entrada, um para cada canal de cor (r, g, b, e h);

2. O segundo nível (LEVEL 2 na Figura 5.10) é composto pela camada de saída da rede neural ART2. Esta camada cria e/ou indica agrupamentos de objetos com características semelhantes, possibilitando o reconhecimento e aprendizagem de conceito e a generalização do conhecimento;

3. O terceiro nível (LEVEL 3 na Figura 5.10) é composto por uma memória plana que armazena as características visuais e auditivas dos objetos.

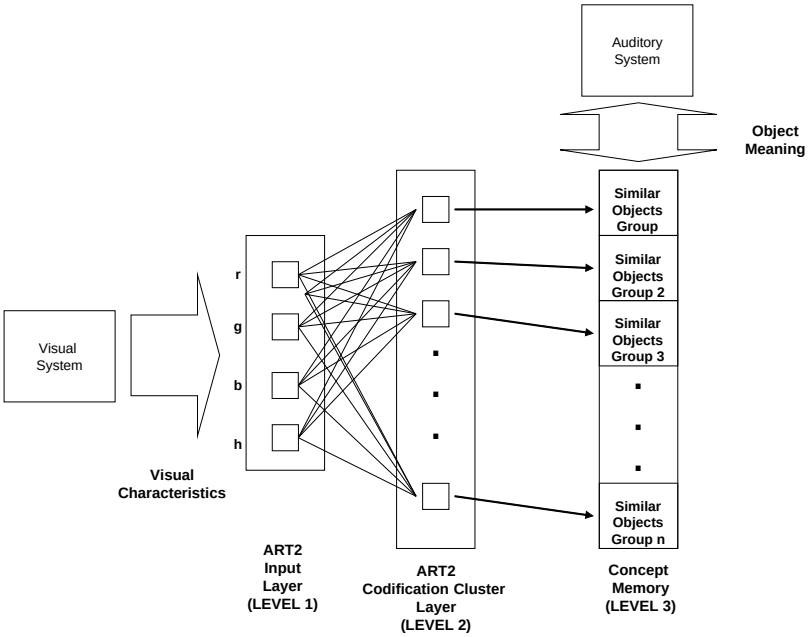

Figura 5.10: Arquitetura geral do mecanismo de aprendizagem. A camada de entrada (input layer) da rede neural ART2 recebe as características de cor de um objeto (r, g, b, e h) do sistema de visão e indica a codificação de *cluster* do objeto. Então, para um objeto desconhecido, o mecanismo de aprendizagem obtém o significado do mesmo, do sistema de voz, e integra estas informações visuais e auditivas para formar um conceito novo na memória de conceitos.

O mecanismo de aprendizagem trabalha da seguinte maneira. Inicialmente, a memória de conceito está vazia. Quando um objeto é apresentado ao robô, o sistema de visão codifica este objeto por seus valores mais freqüentes dos canais r, g, b, e h. A camada de entrada da rede neural ART2

recebe as características de cor deste objeto (r, g, b, e h) do sistema de visão e indica que não há nenhum *cluster* ativo em sua camada de saída. Então, o mecanismo de aprendizagem entra em um estado de *desconhecimento* e ativa o sistema de voz para vocalizar que o objeto é desconhecido. Depois, o mecanismo aguarda o correto significado do objeto, por meio do sistema de reconhecimento de fala. Posteriormente, a rede neural ART2 é treinada no modo rápido (Carpenter e Grossberg, 1987) no qual os pesos das sinapses são atualizados durante o processo de ressonância da rede neural, tomando somente o tempo de uma apresentação do padrão de entrada. Como resultado, o novo conceito aprendido é armazenado na memória de conceitos.

A partir disso, quando são apresentados novos objetos ao robô, a camada de entrada da rede neural ART2 recebe as características de cor do objeto, por meio do sistema de visão, e indica o cluster de codificação, se há um, para o novo objeto. Então, o algoritmo de busca procura por objetos, dentro do cluster indicado, empregando uma métrica dado por: $m = ||h_n - h_r||$, na qual h_n é o valor do h do novo objeto, e h_r é o valor do h de um objeto armazenado na memória de conceitos. Se o algoritmo de busca encontrar algum objeto com métrica de similaridade abaixo de um limiar de confiança ϕ_c, o mecanismo de aprendizagem entra em um estado de *certeza* e ativa o sistema de voz para vocalizar o nome do objeto.

Se o algoritmo de busca encontrar somente objetos acima um limiar de conhecimento ϕ_k, o mecanismo de aprendizagem entra em um estado de *desconhecimento* e ativa o sistema de voz para vocalizar que o objeto é desconhecido. Depois, o mecanismo aguarda o significado correto do objeto, por meio do sistema de reconhecimento de fala. Então, a rede neural ART2 é treinada no modo rápido (Carpenter e Grossberg, 1987) no qual os pesos das sinapses são atualizados durante o processo de ressonância da rede neural, tomando somente o tempo de uma apresentação do padrão de entrada. Como resultado, o novo conceito aprendido é armazenado na memória de conceitos.

Se o algoritmo de busca encontrar somente objetos entre o limiar de confiança e o limiar de conhecimento, o mecanismo de aprendizagem entra em um estado de *incerteza* sobre o objeto e ativa o sistema de voz para vocalizar que o objeto é supostamente o objeto mais similar encontrado na memória de conceitos. Depois, o mecanismo aguarda a confirmação do significado ou o significado correto do objeto, por meio do sistema de reconhecimento de fala, e armazena o novo conceito aprendido na memória de conceitos caso o mecanismo tenha errado o seu *chute* sobre o nome do objeto.

5.7 Considerações Finais

Uma arquitetura robótica inspirada na Análise do Comportamento pode beneficiar a construção de robôs sociais interativos para diversas aplicações. O capítulo faz uma breve descrição sobre a arquitetura que foi projetada para ser aplicada a robótica social, no contexto do problema da atenção compartilhada. A cabeça robótica e o simulador de interação sociais que foram utilizados também são apresentados.

CAPÍTULO

6

Resultados

O capítulo apresenta uma breve introdução sobre as características do problema do mundo dos blocos, um dos ambientes utilizado na realização dos experimentos, e os resultados obtidos. Posteiromente, são apresentadas os resultados dos experimentos do simulador de Interações Sociais e da interação com a cabeção robótca. E finalmente, na seção 6.5 são feitas as considerações finais do capítulo.

6.1 Introdução

Após as pesquisas bibliográficas necessárias ao embasamento teórico, foram realizadas algumas atividades de planejamento e definição do algoritmo proposto neste trabalho. Após estas atividades, a primeira versão do mecanismo de aprendizado proposto foi modelado. Esta versão foi implementada e testada em uma simulador do pronlema do Mundo dos Blocos (Nilsson, 1980; Langley, 1995; Slaney e Thiébaux, 2001).

Após esta validação, foi realizada algumas pequenas adaptações do algoritmo TGE para ser inserido como mecanismo de aprendizado na arquitetura. Posteriormente, foram realizados diversos experimentos para a validação da arquitetura em um problema de aplicação real e não trivial, o aprendizado da atenção compartilhada (Dube et al., 2004; Kaplan e Hafner, 2004). Para tanto, foi utilizado um simulador de interações sociais (detalhado no capítulo anterior) desenvolvido por Policastro (2008), capaz de simular os movimentos necessários ao robô e ao ser humano, além de fornecer estímulos apropriados do ambiente. Após este estudo, outros mecanismos foram incorporados à arquitetura para possibilitar um novo estudo para a comparação do desempenho da arquitetura original (com a utilização do aprendizado por contingência) com a versão que utiliza

um algoritmo Q-Learning clássico (forma tabular), além da comparação com o algoritmo de aprendizado TGE, proposto no presente trabalho.

6.2 Simulação no Mundo dos Blocos

O mundo dos blocos consiste em um número pré-estabelecido de blocos, e cada um possui uma identificação única e uma superfície. Os blocos podem estar sobre a superfície ou sobre um outro bloco. Assim um bloco B pode estar sobre um único bloco C e nenhum outro bloco pode estar sobre o bloco C. A situação onde há uma sobreposição de blocos é conhecido como torre. Um outro fator que deve ser colocado é que pelo menos um bloco obrigatóriamente esteja sobre a superfície.

As ações no mundo dos blocos são feitas a partir da movimentação dos blocos livres para uma outra posição, mudando assim a configuração do ambiente. Os blocos livres são aqueles que não possuem nenhum bloco sobre ele, estes podem ser blocos que estão no topo de uma torre ou sobre a superfície que não estejam em nenhuma torre.

A Figura 6.1 mostra um possível estado do problema do mundo dos blocos (que contém 5 blocos) sendo modificado por uma determinada ação. A ação está sendo representada pela seta pontilhada e está movendo o bloco 3 e colocando-o sobre o bloco 2. Nesse caso, cada bloco é identificado com um número, facilitando a identificação.

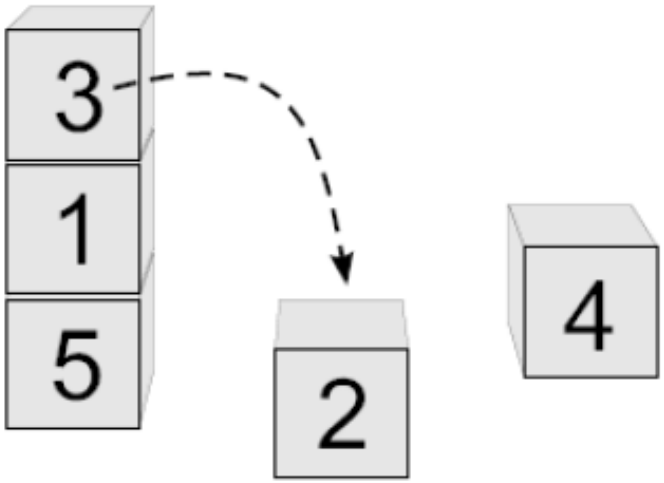

Figura 6.1: Um exemplo de Mundo dos Blocos com 5 blocos. A ação é indicada pela seta pontilhada (Driessens, 2004).

Assim, a seguinte representação é utilizada:

- clear(A): verdadeiro se o bloco A não tem algum bloco sobre ele
- on(A,B): verdadeiro se o bloco A está sobre o B. O bloco B pode ser um bloco ou a superfície.

para cada predicado a associação ao valor binário, segundo seção 4.1 do capitulo 4, é realizada.

6.2.1 Configuração do ambiente

Em uma primeira fase, foi realizado três experimentos com número diferentes de blocos para validar a proposta do algoritmo de aprendizagem TGE. Os experimentos foram conduzidos com 3, 4 e 5 blocos sendo realizado um total de 50 turnos com 500 episódios para cada um. Para cada turno foi calculado a média do valor Q obtido pelo agente, sendo este o critério de avaliação definido como recompensa média, de acordo com Driessens (2004).

O ambiente foi configurado para que o reforço positivo aplicado ao agente fosse o valor inteiro 1 e como reforço negativo fosse valor inteiro 0. Adotou-se que uma ação aleatória será escolhida com uma probabilidade de 30%, isto é, foi utilizada 30% como taxa de exploração do ambiente.

Um outro fator a ser considerado em ambos os testes é o fato do aprendizado contínuo. O agente aprende de acordo com a interação, não existindo uma separação entre fase de aprendizado e de validação. As vantagens de utilizar essa técnica está na não dependencia de um conjunto de fatos e pelo fato de estar mais próximo da realidade humana.

Além disso, os objetivos foram selecionados de forma randômica entre três diferentes opções. A primeira e mais simples é a colocação de todos os blocos sobre a superfície. A segunda, parte da construção de uma única e especifíca torre, isto é, a partir de um estado inicial para um mundo dos blocos com 3 blocos, construir a torre onde o bloco 3 está sobre o bloco 1 e este esteja sobre o bloco 2 que por fim esteja sobre a superfície. A última opção parte da hipótese que todos estão sobre a superfície exceto um, isto é, atingir o estado onde o bloco 3 esteja sobre o bloco 1, sendo este e o bloco 2 estejam sobre a superfície.

Em uma segunda fase, foi desenvolvido uma adaptação do algoritmo TGE que utilizou como base o Algoritmo 6, especifícado do capítulo 3 para compararmos o aproveitamento de ambos e termos parametros para uma boa escolha de um algoritmo de aprendizado. O mecanismo de regressão utilizado por essa adaptação é o mesmo do TGE e foi descrito no capítulo 4, através do algoritmo 9.

A adaptação possui um vetor auxiliar para armazenar os exemplos antes de inserir no mecanismo de regressão, e para isso foi utilizado uma matriz com 1.000.000 linhas e 4 colunas. As colunas armazenam em sua posição o Q, o estado, a ação e a recompensa recebida.

Utilizando a mesma configuração da primeira fase, alterando-se somente a probabilidade de exploração que foi ajustada em 10%, 30% e 50%. Os algoritmos foram avaliados pela média do reforço obtido pelo agente e o tempo gasto para resolver os 25000 episódios executados.

Em ambas as etapas o equipamento e o funcionamento eram iguais. Para realizar os dois experimentos foi utilizado um notebook HP pavilion dv6000 com processador Intel Core Duo 2.00 GHz e um gigabyte de memória (*RAM*), e 140 gigabytes de memória rigída e foi utilizado o sistema operacional Windows Vista Home Premium.

O funcionamento do simulador é feito em etapas. O episódio se inicia pela escolha, de froma aleatória, do estado inicial e o objetivo do agente. O agente recebe o estado e realiza ação sobre o ambiente, modificando-o, a fim de encontrar o objetivo. Esse fato se realiza até que o objetivo seja encontrado. Todo o processo se repete até que o número de épocas determinadas pelo usúario seja atingido.

6.2.2 Resultados experimentais

A adoção da métrica para a primeira fase é importante pois nos mostra, na média, a recompensa necessária em cada episódio e traço da evolução do aprendizado sobre os turnos, que dessa forma nos mostra o aprendizado realizado.

Na Figura 6.2, é mostrado os resultados dos testes realizados onde pode ser observado a curva de reforço crescendo até um limiar, indicando a aquisição de conhecimento do agente. No momento que a curva atinge o limiar a curva permanece em um nível, mostrando que o agente encontrou uma política ótima.

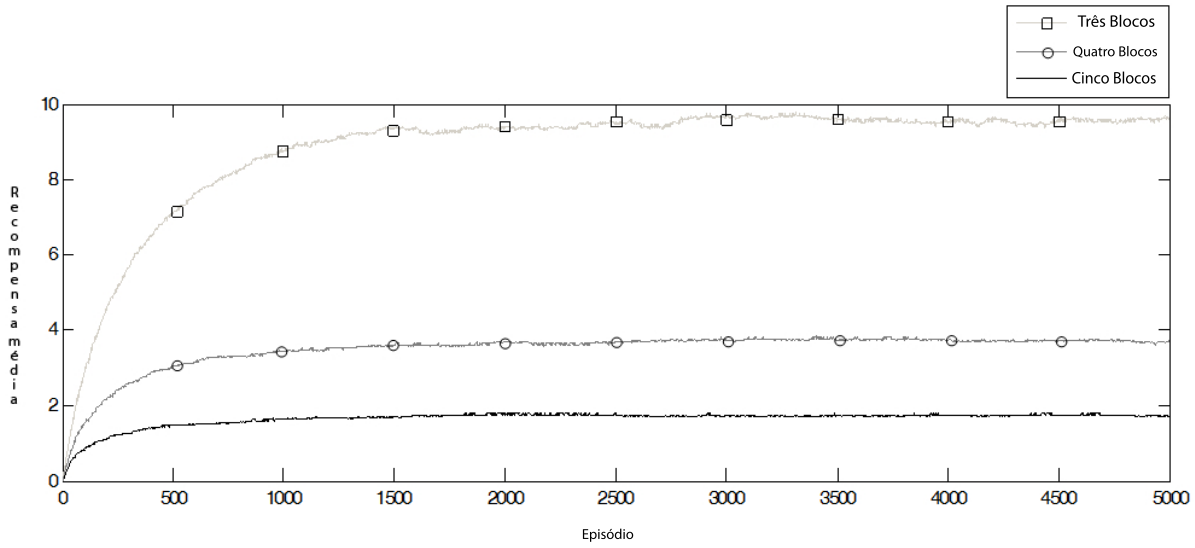

Figura 6.2: Curva de aprendizado. Curva de aprendizado do algoritmo TGE utilizando a média do valor Q para o mundo dos Blocos.

Pode-se observar também que o agente inicia explorando o ambiente e a medida que encontra o objetivo, reforço positivo, ele vai construindo uma política no sentido de reduzir os estados intermediários. À medida que o número de iterações vai aumentando, a curva de aprendizado vai crescendo até encontrar um limiar, ou seja, até atingir um ponto no qual o agente alcançou uma política ótima.

A fim de realizar a avaliação do tempo gasto para resolver os episódios foi utilizado dois marcadores de tempo da biblioteca *time.h*. Um marcador para identificar o inicio do processo

e o outro para marcar o fim de todo o processo. A diferença entre ambos foi armazenada em segundos e é mostrada na tabela 6.1, para números diferentes de blocos.

	TGE			Adaptação do TGE		
	10%	30%	50%	10%	30%	50%
3 Blocos	69	67	106	708	247	136
4 Blocos	134	132	91	-	1536	296
5 Blocos	2425	1326	439	-	-	3083

Tabela 6.1: Avaliação de Tempo de execução, em segundos, dos algoritmos TGE e sua adaptação no Mundo dos Blocos

A tabela revela que o algoritmo TGE é mais rápido que a sua adaptação em qualquer uma das configurações adotada. A adaptação do TGE ainda encontra um outro problema que é o caso onde não existe mais espaço na sua estrutura auxiliar e o programa é encerrado sem completar a processo. Esse caso é verificado na tabela pela falta de valor, no que foi representado por um traço (-).

Dentro da abordagem do algoritmo TGE podemos ver como, para a realização dos teste para 3 blocos, o aumento do fator exploração do ambiente reduziu a eficiência do algoritmo. Um dos fatores que pode ter levado a essa redução é a alta exploração quando o agente já encontrou a política ótima.

As Figuras 6.3, 6.4 e 6.5 mostram a curva de aprendizado para dois sistemas TGE e sua adaptação para o mundo dos Blocos com 3, 4 e 5 blocos, respectivamente. Para uma melhor visualização do aprendizado do algoritmo TGE adaptado, foi realizada uma redução do número total de iterações consideradas em cada gráfico. O valor máximo em cada um é de 50, pois após esse valor não existe nenhuma alteração, ou seja, as linhas permanecem constantes.

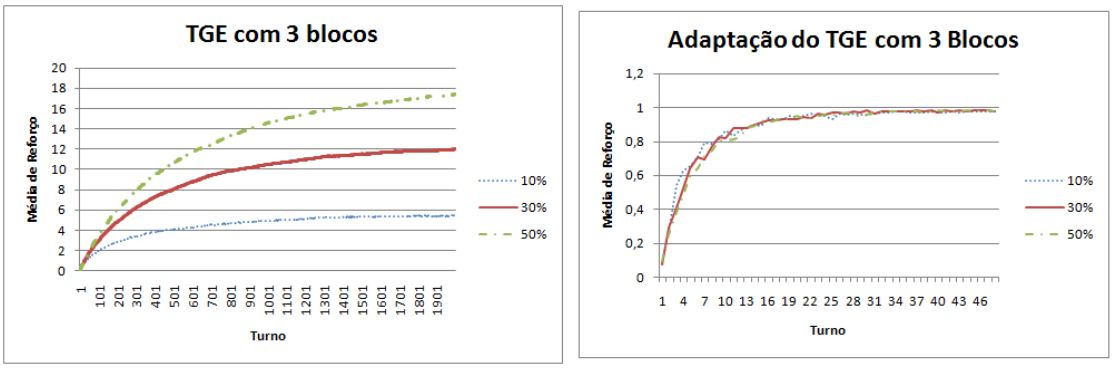

Figura 6.3: Curva de aprendizado no mundo com 3 Blocos. A esquerda, é mostrada a evolução do algoritmo TGE sobre diferentes valores do parâmetro de exploração e a direita a adaptação do algoritmo TGE sobre as mesmas condições.

Ao analisar cada um desses gráficos, nas Figuras 6.3, 6.4 e 6.5, pode-se ver que em cada ambiente a adaptação do algoritmo TGE realiza uma rápida convergência. Esse fato mostra um aprendizado mais rápido do que o algoritmo TGE.

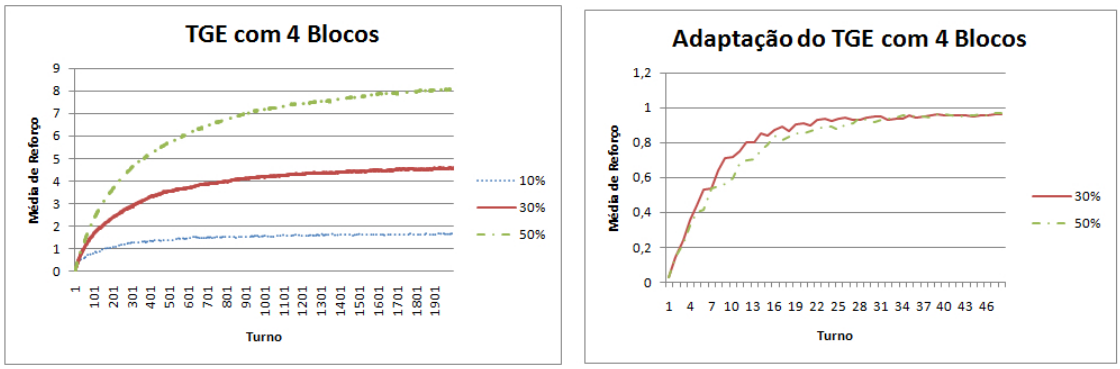

Figura 6.4: Curva de aprendizado no mundo com 4 Blocos. A esquerda, é mostrada a evolução do algoritmo TGE sobre diferentes valores do parâmetro de exploração e a direita a adaptação do algoritmo TGE sobre as mesmas condições.

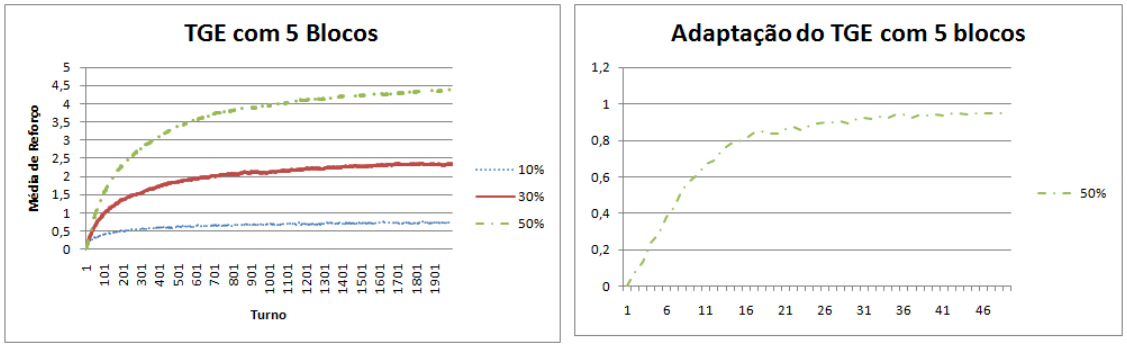

Figura 6.5: Curva de aprendizado no mundo com 5 Blocos. A esquerda, é mostrada a evolução do algoritmo TGE sobre diferentes valores do parâmetro de exploração e a direita a adaptação do algoritmo TGE sobre as mesmas condições.

6.3 Simulação da Atenção Compartilhada

Após os primeiros experimentos do mecanismo de aprendizado com o simulador para resolver o problema do mundo dos blocos, foram realizadas algumas adaptações ao sistema de aprendizado proposto e então foram executados diversos experimentos para a validação da arquitetura em um problema de aplicação real e não trivial: o aprendizado da atenção compartilhada (Dube et al., 2004; Kaplan e Hafner, 2004). Para tanto, foi utilizado um simulador de interações sociais, baseado em trabalhos de Triesch e colegas (Triesch et al., 2006), capaz de simular os movimentos necessários ao robô e ao ser humano, além de fornecer estímulos do ambiente apropriados ao contexto da atenção compartilhada. Os detalhes sobre o simulador são apresentados na próxima seção.

6.3.1 Análise dos Resultados

A análise do desempenho da arquitetura foi baseada na quantificação do aprendizado do domínio de aplicação que é a Atenção Compartilhada. Os experimentos foram compostos por uma fase de aprendizado com duração diferente para cada estudo realizado durante o desenvolvimento deste trabalho. Para os experimentos simulados da atenção compartilhada, a fase de aprendizado foi executada com duração de 10.000 unidades de tempo.

Para os experimentos sobre a atenção compartilhada, a capacidade de aprendizado da arquitetura foi analisada pela observação da interação do robô com o humano e o ambiente, e pela computação de uma métrica denominada *correct gaze index ou índice de olhar correto* (CGI). Esta métrica é baseada nos trabalhos apresentado por Whalen e Schreibman (2003) e é definida como a freqüência de direcionamento da atenção para o local correto, o qual o humano está olhando. Esta métrica é dada pela Equação:

$$CGI = \frac{\#shifts\ from\ the\ human\ to\ correct\ location}{\#shifts\ from\ the\ human\ to\ any\ location} \tag{6.1}$$

Para quantificar o aprendizado da arquitetura durante os experimentos, a fase de aprendizado foi interrompida em pontos específicos, a cada 500 unidades de tempo. Então, uma fase de validação do conhecimento adquirido foi iniciada para avaliar o comportamento da arquitetura. Esta avaliação foi executada por 20 execuções de 500 unidades de tempo ou 500 ciclos de interação. Para cada execução, o valor do CGI foi computado, dado pela Equação 6.1. Depois das 20 execuções, a média e desvio padrão das 20 medidas foram calculados, de acordo com as Equações (6.2) e (6.3), respectivamente:

$$\overline{CGI} = \frac{1}{k}\sum_{i=1}^{k} CGI_i \tag{6.2}$$

$$d = \frac{1}{k}\left[\frac{1}{k-1}\sum_{i=1}^{k}(CGI_i - \overline{CGI})^2\right] \tag{6.3}$$

Depois das 20 execuções da fase de avaliação, a fase de aprendizado foi retomada do ponto no qual esta havia sido interrompida.

6.3.2 Principais Resultados

Nesta Seção, são apresentados os principais resultados dos experimentos executados para a avaliação da arquitetura no contexto do aprendizado da atenção compartilhada, empregando-se o simulador de interações sociais apresentado anteriormente. O propósito destes experimentos foi a determinação da capacidade de exibição de comportamentos apropriados e de aprendizagem da arquitetura com o algoritmo TGE durante o controle do robô simulado em um ambiente social controlado. Foi realizado também uma avaliação comparativa entre três técnicas de aprendizado: aprendizado por contingência, *Q-learning* tradicional (utilizando uma matriz) e TGE.

Para este conjunto de experimentos, o conhecimento de arquitetura foi configurado da seguinte maneira. Quatro estímulos foram declarados: *face*, *object*, *attention* e *environment*, no qual *attention* é um estímulo reforçador gerado com a atenção do ser humano. Foram declarados dois fatos para definir que objetos vermelhos e azuis são brinquedos. Também foram declarados treze fatos para diferenciar a pose da cabeça do ser humano como frontal, além de seis poses de perfil esquerdo e seis poses de perfil direito. Adicionalmente, foram declarados mais fatos para definir quando o robô está focalizando o ser humano ou um brinquedo.

O módulo de emissão de respostas foi configurado como a seguir. A constante de aprendizado (parâmetro α) foi determinada com o valor $\alpha = 0.2$. O reforço positivo foi configurado com valor 10 e o reforço negativo com valor -1. A constante de exploração foi configurado com o valor de 5% ou 0.05. O valor do fator de desconto (parâmetro γ) foi colocada somo sendo igual a 0.1. Foram definidas quatorze respostas de forma que o robô pudesse olhar para o ser humano ou procurar brinquedos em seis regiões definidas ao girar sua cabeça para a esquerda e seis regiões definidas ao girar a sua cabeça para a direita. Isto foi feito para discretizar o ambiente em regiões de interesse que tornaram possível ao robô aprender a seguir o olhar do ser humano para locais corretos, mesmo na presença de objetos distratores. Vale ressaltar que esta discretização foi empregada em diversos trabalhos apresentados por Triesch e seus colegas (Fasel et al., 2002; Carlson e Triesch, 2003; Lau e Triesch, 2004; Deák e Triesch, 2005b).

O sistema motivacional foi configurado como segue. Foi criada uma unidade de necessidade: socialize. O limiar de ativação do sistema motivacional foi fixado em 0.70. A inclinação da função sigmóide das unidades de necessidade (parâmetro δ) foi configurada com valor igual a 0.20. Para a unidade socialize, o bias foi configurado com valor igual a 1.00 e o peso de sua conexão foi configurado com valor igual a 0.5. O peso da conexão recorrente foi configurado com valor igual a 1.00. Os pesos das conexões das unidades de entrada $(hear(attention)$, $see(frontal(face))$,

$see(toy(object))$, $see(looking_{t}oy(object)))$ foram configurados, respectivamente, com valores iguais a -1.00, 0.05, 0.05 e 0.00. Durante o processo de configuração da arquitetura, verificou-se empiricamente que estes valores produziram os melhores resultados.

Conforme a metodologia apresentada anteriormente, estes experimentos foram compostos por uma fase de aprendizagem de 10.000 unidades de tempo (10.000 segundos no simulador). Durante a fase de aprendizagem, o ser humano mantinha o foco inicialmente no robô até que este estabelecesse o contato ocular com aquele, definido como 3 unidades de tempo mantendo-se o contato ocular. Então, dois objetos eram posicionados no ambiente e o ser humano direcionava o seu olhar para um destes objetos, obedecendo as probabilidades definidas no simulador de interações sociais. O humano mantinha o seu olhar no objeto selecionado por 5 unidades de tempo. Depois, os objetos eram removidos do ambiente social e o ser humano voltava a olhar para o robô, aguardando que este estabelecesse contato ocular novamente. Este procedimento foi executado para simular uma interação social na qual dois agentes mantêm contato ocular e então um deles direciona o olhar para um evento ou objeto interessante no ambiente.

Nas primeiras 100 unidades de tempo da fase aprendizagem, nenhum objeto foi posicionado no ambiente e o ser humano manteve o seu foco no robô ao longo deste período. Nestas primeiras 100 unidades de tempo, o robô aprendeu que manter contato ocular com ser humano, quando ele está olhando para o robô, produz alguns estímulos reforçadores de atenção do ser humano, satisfazendo sua necessidade de socialização (unidade de necessidade socialize configurada no sistema motivacional). Este procedimento foi feito para modelar o comportamento do robô de procurar por um ser humano e manter contato ocular sempre que sente necessidades de socialização. Depois das 100 primeiras unidades de tempo, a fase de aprendizagem prosseguiu empregando dois objetos, como declarado anteriormente. A partir desta etapa da aprendizagem, o robô sempre olhava para o ser humano quando queria interagir socialmente (unidade de necessidade socialize estava ativa). Entretanto, quando um objeto foi posicionado no ambiente e o ser humano direcionava seu olhar para este objeto, o robô perdeu a atenção do humano e começou a buscar qualquer estímulo no ambiente que pudesse satisfazer seus estados internos.

Adicionalmente, se o robô procurasse por um objeto que também era o foco de atenção do ser humano, este voltava a dar atenção ao robô, pela simulação de verbalizações do ser humano em relação ao objeto, o qual ambos estavam focando, representado no simulador pelo estimulo *attention*. Deste modo, após de uma história de reforçamento, o robô aprendeu seguir o olhar do ser humano para receber a atenção dele e satisfazer suas necessidades de socializar.

A capacidade de aprendizagem da arquitetura foi analisada pela computação da métrica CGI, dada pela Equação (6.1). Ela é definida como a freqüência de alternância do olhar, a partir do contato ocular com o humano, para um objeto que é foco de atenção do ser humano.

Conforme a metodologia apresentada anteriormente, para quantificar o aprendizado da arquitetura durante os experimentos, a fase de aprendizagem foi interrompida em pontos específicos (a cada 500 unidades de tempo) e uma fase de validação do conhecimento adquirido foi iniciada para avaliar o comportamento da arquitetura. Esta avaliação foi executada por 20 execuções de 500

unidades de tempo (500 ciclos de interação). Para cada execução, o valor do CGI era computado e, depois das 20 execuções, a média e desvio padrão das 20 medidas eram calculados, dados respectivamente pelas Equações (6.2) e (6.3). Depois das 20 execuções da fase de avaliação, a fase de aprendizado era retomada do ponto no qual esta havia sido interrompida.

Na Figura 6.6, é apresentada a curva de aprendizado que demonstra o progresso da aprendizagem durante os experimentos de simulação. Esta figura mostra um gráfico que apresenta o valor da média do CGI para cada fase de avaliação, em pontos específicos durante o processo de aprendizagem. O resultado obtido mostra que o valor do CGI é crescente no decorrer da fase de aprendizagem, demonstrando a capacidade de aprendizagem da arquitetura com o algoritmo TGE.

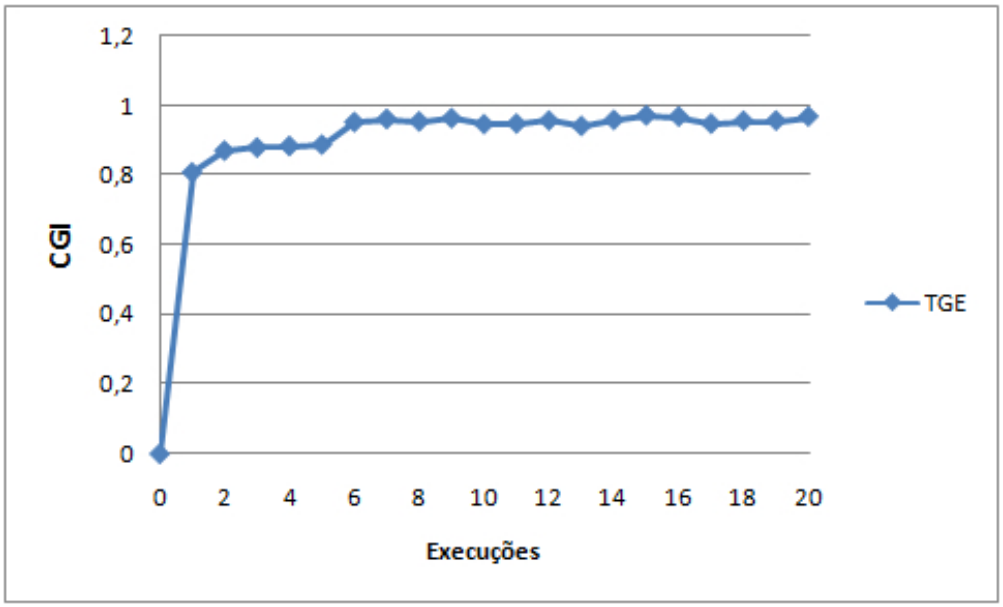

Figura 6.6: Evolução do aprendizado no Simulador, utilizando o algoritmo TGE na arquitetura robótica

Na Figura 6.7, é apresentada a curva da atenção obtida pelo agente do robô, ou seja, quando o agente consegue manter contato ocular com o ser humano, para três mecanismo de aprendizado (aprendizado de contingência, *Q-learning* tradicional e ETG). Ela contém também o valor do desvio padrão do intervalo. Esse fator nos ajuda na avaliação de cada sistema de aprendizado, pois não basta ter uma alta média de CGI se o contato ocular foi baixo, ou seja, isso significa que o agente conseguiu realizar bem o foco no qual o ser humano está observando, mas não conseguiu manter contato ocular.

E neste ponto o algoritmo de aprendizado por contingência foi bem melhor que os outros dois algoritmos.

O comparativo entre os três mecanimos de aprendizado é representado na Figura 6.8. Esta figura contém também o valor do desvio padrão do intervalo. As curvas de aprendizado demonstram o progresso da aprendizagem durante os experimentos de simulação para cada mecanismo. Esta figura mostra um gráfico que apresenta o valor da média do CGI para cada fase de avaliação, em pontos específicos durante o processo de aprendizagem. O resultado obtido mostra que o TGE

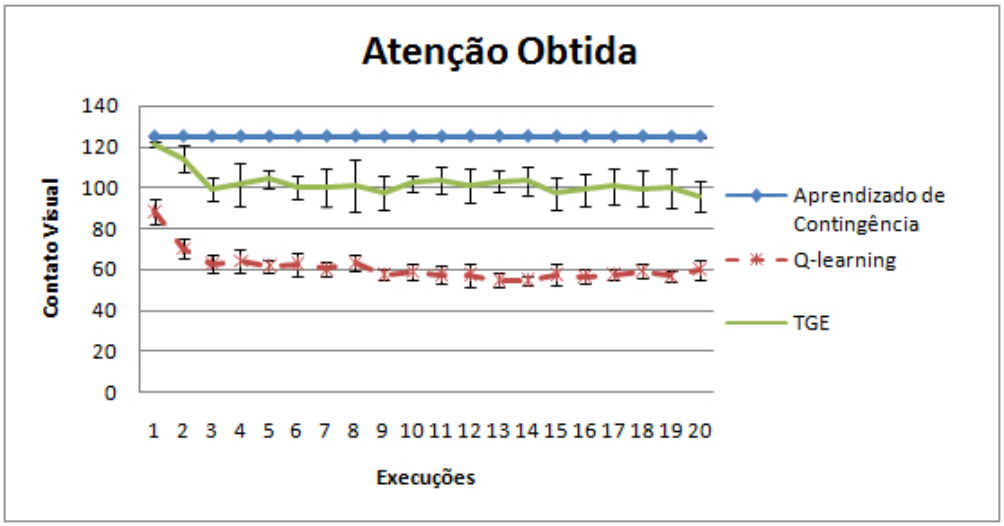

Figura 6.7: Quantidade de contato visual que o robô manteve com o ser humano

tem desempenho similar ao *Q-learning* tradicional, apesar do fato que em alguns momentos o TGE seja superior. A figura mostrou ainda que ambos são melhores que o aprendizado por contingência.

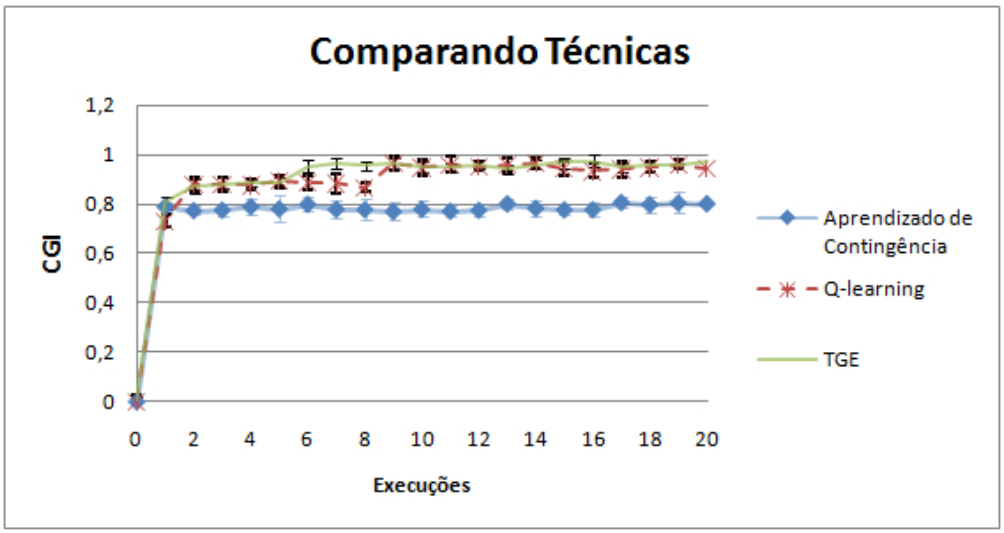

Figura 6.8: Comparação entre as técnicas. Comparativo da evolução do aprendizado entre o TGE, AR e Aprendizado de Contingência

Este resultado também foi confirmado por um análise do intervalo de confiança utilizando uma confiança de utilizando a confiança de 99,95%. Os resultados são mostrados na Tabela 6.2 e no gráfico 6.9.

Uma análise mais profunda pode ser feita considerando as Figuras 6.7 e 6.8. Considerando os fatores de aprendizado e contato ocular, pode-se afirmar que o algoritmo TGE consegue melhores resultados. Essa afirmação é feita pois o algoritmo obteve o melhor resultado relativo ao aprendizado (junto com o algoritmo *Q-learning*), conseguindo manter um bom contato visual. Os outros métodos tiveram um resultado baixo em um dos testes.

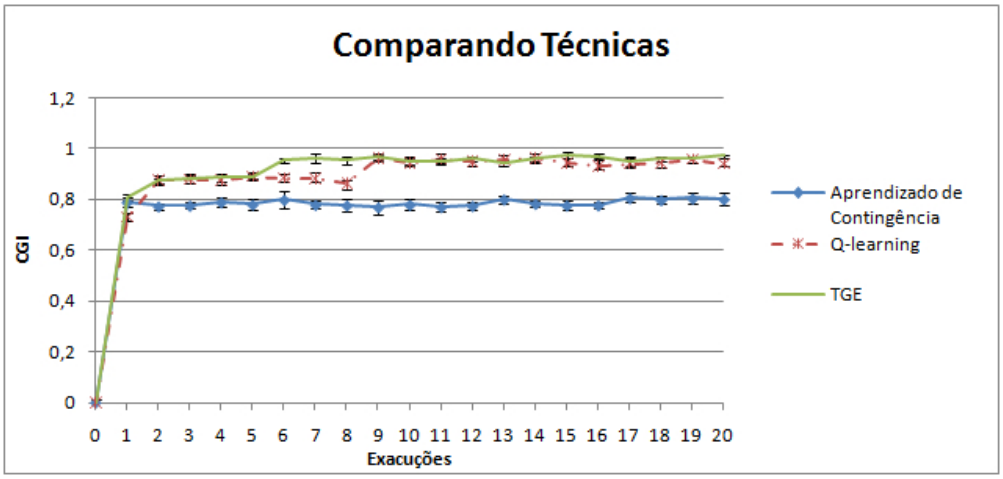

Figura 6.9: Comparação entre as técnicas utilizando o intervalo de confiança

Tabela 6.2: Resultados do intervalo de confiança para o processo de aprendizado.

Modelos Comparados	Conclusão
Aprendizado por Contingência e *Q-Learning* tradicional	A versão com *Q-Learning* tradicional é superior
Aprendizado por Contingência e TGE	A versão com o TGE é superior
TGE e *Q-Learning* tradicional	Desempenho Similar

De fato, para estes experimentos, o *Q-Learning* empregou como mecanismo de representação do conhecimento uma *Tabela Q* de 3150 posições. Já o TGE utilizou 68 nós, sendo que 29 são nós folhas que contribuem com 328 posições para o armazenamento das ações.

6.4 Experimentos sobre a Cabeça Robótica Interativa

Após a realização dos experimentos sobre o aprendizado da atenção compartilhada, empregando o simulador de interações sociais, foram realizados os últimos experimentos e estudos previstos neste projeto de pesquisa, empregando a cabeça robótica interativa apresentada na Seção 5.6 do Capítulo 5. Estes experimentos foram realizados no contexto do aprendizado da atenção compartilhada com o propósito de se avaliar a capacidade de aprendizagem da arquitetura proposta em um ambiente social real e controlado.

6.4.1 Experimentos de Aprendizado da Atenção Compartilhada

Nesta seção, são apresentados os principais resultados dos experimentos executados para a avaliação da arquitetura proposta no domínio de aplicação da aprendizagem da atenção compartilhada, empregando-se a cabeça robótica interativa. O propósito destes experimentos foi a determinação da capacidade de exibição de comportamentos apropriados e de aprendizagem da arquitetura durante o controle do robô em um ambiente social real e controlado.

Para este novo conjunto de experimentos, o conhecimento de arquitetura foi configurado da seguinte maneira. Quatro estímulos foram declarados: $face$, $object$, $attention$ e $environment$. O estímulo $attention$ é um estímulo reforçador gerado com a atenção do ser humano. Foram declarados quatro fatos para definir que objetos vermelhos, amarelos, laranja e verdes são frutas. Também foram declarados seis fatos para diferenciar a pose da cabeça do ser humano. Adicionalmente, foram definidos alguns fatos para definir quando o robô está focalizando o ser humano ou uma fruta. Considerando o valor associado aos estímulus e fatos, essa representação atinge o valor máximo de 2097152 (01000000000000000000000).

O módulo de emissão de respostas foi configurado como a seguir. A constante de aprendizado (parâmetro α) foi determinada com o valor : $\alpha = \ 0.2$. O reforço positivo foi configurado com valor 10 e o reforço negativo com valor -1. A constante de exploração foi configurado com o valor de 5% ou 0.05. O valor do fator de desconto (parâmetro γ) foi configurada com o valor de de 0.1. Foram definidas sete respostas de forma que o robô pudesse olhar para o ser humano ou procurar frutas em cinco regiões definidas ao girar sua cabeça para a esquerda ou para a direita. Isto foi feito para discretizar o ambiente em regiões de interesse que tornaram possível ao robô aprender a seguir o olhar do ser humano para locais corretos, como nos experimentos empregando o simulador de interações sociais.

O sistema motivacional foi configurado como descrito a seguir. Foram criadas duas unidades de necessidade: *socialize*. O limiar de ativação do sistema motivacional foi fixado em 0.50. A inclinação da função sigmóide das unidades de necessidade (parâmetro δ) foi configurada com valor igual a 0.20. Para a unidade *socialize*, o *bias* foi configurado com valor igual a 1.00 e o peso de sua conexão foi configurado com valor igual a 0.5. O peso da conexão recorrente foi configurado com valor igual a 1.00. Os pesos das conexões das unidades de entrada ($hear(attention)$, $see(frontal(face))$, $see(looking\_frontal(face))$, $see(looking\_fruit(object))$) foram configurados, respectivamente, com os valores -1.50, 0.95, -1.50 e 0.50. Durante o processo de configuração da arquitetura, verificou-se empiricamente que estes valores produziram os melhores resultados.

Os experimentos foram compostos por uma fase de aprendizagem de 500 ciclos de interação. Durante a fase de aprendizagem, o ser humano mantinha o foco inicialmente no robô até que este estalecesse o contato ocular. Então, uma fruta era posicionada no ambiente e o ser humano direcionava o seu olhar a mesma. O ser humano mantinha o seu olhar no objeto selecionado até o robô emitir uma resposta (executação uma ação motora). Depois, a fruta era removida e o ser humano voltava a olhar para o robô, aguardando que este estabelece novamente o contato ocular. Este procedimento foi executado para simular uma interação social na qual dois agentes estão mantendo contato ocular e então um deles direciona o olhar para um evento ou objeto interessante no ambiente.

Nas primeiras 30 unidades de tempo da fase aprendizagem, nenhum objeto foi posicionado no ambiente e o ser humano manteve o seu foco no robô ao longo de todo este período de tempo. Nestas primeiras 30 unidades, o robô aprendeu que estabelecer o contato ocular com ser humano

produz alguns estímulos reforçadores de atenção do mesmo, satisfazendo sua necessidade de socialização (unidade de necessidade *socialize* configurada no sistema motivacional). Este procedimento foi executado para modelar o comportamento do robô de procurar por um ser humano e manter contato ocular sempre que sente necessidades de socialização. Depois das 30 primeiras unidades de tempo, a fase de aprendizagem prosseguiu empregando uma fruta como declarado acima. A partir deste ponto da aprendizagem, o robô sempre olhava para o ser humano quando queria interagir socialmente (unidade de necessidade *socialize* estava ativa). Entretanto, quando uma fruta era posicionada no ambiente e o ser humano direcionava seu olhar para esta, o robô perdia a atenção do humano e começava a procurar por qualquer estímulo no ambiente que pudesse satisfazer seus estados internos (unidades de necessidade *socialize*). Quando o robô olhava para um objeto definido como uma fruta na base de conhecimento da arquitetura, este satisfazia suas necessidades. Adicionalmente, se o robô focasse uma fruta que também era o foco de atenção do ser humano, este último voltava a dar atenção ao robô, em relação a fruta a qual ambos estavam focando.

Como as frutas eram posicionadas no ambiente somente quando o ser humano direcionava seu foco para as mesmas, depois de uma história de reforçamento o robô aprendeu seguir o olhar do ser humano para receber a atenção dele e satisfazer sua necessidade de socializar.

A capacidade de aprendizagem da arquitetura foi analisada observando-se a interação do robô com o ser humano e com o ambiente e pela computação da métrica (CGI), dada pela Equação (6.1), apresentada nessa Seção. Para quantificar o aprendizado da arquitetura durante os experimentos, a fase de aprendizagem era interrompida em pontos específicos (a cada 100 unidades de tempo) e uma fase de validação do conhecimento adquirido era iniciada para avaliar o comportamento da arquitetura. Esta avaliação era executada por 5 execuções de 100 unidades de tempo (100 ciclos de interação). Para cada corrida, o valor do CGI era computado e, depois das 5 execuções, a média e desvio padrão das 5 medidas eram calculados, dados respectivamente pelas Equações (6.2) e (6.3). Depois das 5 execuções da fase de avaliação, a fase de aprendizado era retomada do ponto no qual esta havia sido interrompida.

Na Figura 6.10, é apresentada a curva de aprendizado que demonstra o progresso da aprendizagem durante os experimentos. Esta figura mostra um gráfico que apresenta o valor da média do CGI para cada fase de avaliação, em pontos específicos durante a fase de aprendizagem. Os resultados obtidos mostram que o valor do CGI é crescente no decorrer da fase de aprendizagem, demonstrando as capacidades de aprendizagem da arquitetura.

Estes resultados mostram que a arquitetura é capaz de exibir comportamentos apropriados durante uma interação social real e controlada. Adicionalmente, os resultados confirmam os resultados dos experimentos anteriores, mostrando que a arquitetura pode aprender a partir de uma interação social. Os resultados mostram ainda que a arquitetura suporta a modelagem de comportamentos, isso é, ela possibilita que o comportamento do robô seja modelado por meio de aproximações sucessivas, por meio do reforço e encadeamento de comportamentos inatos para a implantação de um comportamento mais complexo. Nestes experimentos, isto foi realizado pela

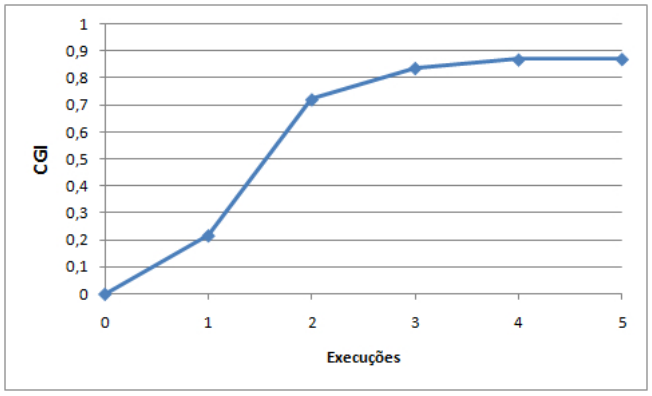

Figura 6.10: Evolução do aprendizado durante os experimentos.

modelagem do comportamento de procurar por um ser humano e, após isso, pela modelagem do comportamento de seguir o olhar deste.

6.4.2 Experimentos de Aprendizado por Tutelagem

Nesta seção são apresentados e discutidos os principais resultados obtidos com os experimentos executados para se avaliar a interface multimodal e o mecanismo de aprendizagem por tutelagem proposto nesta pesquisa. No cenário experimental, um ser humano direcionava a atenção do robô e lhe apresentava diversos objetos com o objetivo de ensinar os nomes dos mesmos ao robô. Como apontado anteriormente, este experimento demonstra o valor do aprendizado da atenção compartilhada como precursor do aprendizado social. Os objetos empregados nos experimentos foram 3 tipos de frutas: uma *maçã vermelha*, um *limão amarelo*, e uma *laranja*. O propósito dos experimentos foi a avaliação da capacidade do mecanismo de aprendizagem em exibir comportamento social apropriado, em aprender a partir da interação social e generalizar os conceitos aprendidos sobre os objetos.

Adicionalmente, a arquitetura foi configurada com o conhecimento prévio que a permitiu ativar o mecanismo de aprendizagem por tutelagem quando o ser humano solicitava ao robô o reconhecimento do objeto apresentado. Este conhecimento prévio foi configurado na forma de estímulos auditivos que codificavam os diálogos previstos durante os experimentos. Adicionalmente, o conhecimento foi configurado na forma de regras de comportamento como a regra a seguir:

$$\ldots$$
$$\text{see}(\text{looking\_fruit}(\text{object}))\&\text{hear}(\text{speech\_identify}) \underset{socialize}{\overset{1.00}{\rightarrow}} \text{recognizeobject}()$$
$$\ldots$$

Para avaliar o mecanismo de aprendizagem proposto, foram calculadas 5 métricas durante os experimentos: *taxa de desconhecimento*, *taxa de suposições corretas*, *taxa de suposições incorretas*, *taxa de erros* e *taxa de acertos*. A *taxa de desconhecimento* é a freqüência na qual o mecan-

ismo de aprendizagem entrou no modo de *desconhecimento*. A *taxa de suposições incorretas* é a freqüência na qual o mecanismo de aprendizagem entrou no modo de *incerteza* e supôs incorretamente o nome da fruta. A *taxa de suposições corretas* é a freqüência na qual o mecanismo de aprendizagem entrou no modo de *incerteza* e supôs corretamente o nome da fruta. A *taxa de erros* é a freqüência na qual o mecanismo de aprendizagem entrou no modo de *conhecimento*, mas apontou incorretamente o nome da fruta. A *taxa de acertos* é a freqüência na qual o mecanismo de aprendizagem entrou no modo de *conhecimento* e apontou corretamente o nome da fruta.

Os experimentos para realização deste proposito foram realizadas junto com os experimentos da atenção compartilhada. A fase de aprendizagem era interrompida em pontos específicos (a cada 100 unidades de tempo) e uma fase de validação do conhecimento adquirido era iniciada para avaliar o comportamento da arquitetura. Esta avaliação era executada por 5 execuções de 100 unidades de tempo (100 ciclos de interação). O conhecimento prévio foi acionado a partir do momento em que uma fruta foi notada pelo robô, na qual já estava sendo foco do ser humano. Então, ao final cada fase de avaliação, a média e o desvio padrão das métricas foram calculados.

Na Tabela 6.3 são mostrados os valores das médias e desvios padrão das 5 métricas para todo o experimento.

Tabela 6.3: Resultados obtidos após as todo processo.

Métrica	Média (%)
Taxa de desconhecimento	7.23 ± 0.58
Taxa de suposições corretas	17.47 ± 1.28
Taxa de suposições incorretas	1.8 ± 0.44
Taxa de erros	0.6 ± 0.2
Taxa de acertos	72.89 ± 2.19

Os resultados mostram que o mecanismo de aprendizagem é capaz de exibir comportamento apropriado e aprender a partir de interações sociais. Os resultados mostram também que durante os experimentos o mecanismo de aprendizagem foi capaz de exibir *conhecimento*, *incerteza*, e *certeza* sobre os nomes das frutas, durante a fase de apresentação, permitindo um processo de aprendizagem socialmente direcionado e de uma maneira mais natural. O uso de certeza e incerteza sobre um objeto permitiu ao ser humano determinar a compreensão exata do robô sobre o conceito aprendido. Os exemplos seguintes ilustram como um ser humano interage com o robô de acordo com suas respostas:

[Ser humano apresenta uma maçã vermelha]
h −− Robô, o que é isso?
r −− Eu não sei!
h −− Isto é uma maçã.
r −− Certo.
h −− Isto mesmo!

...
[Ser humano apresenta uma maçã vermelha]
h −− Robô, o que é isso?
r −− Parece uma maçã.
h −− Muito bom!
...
[Ser humano apresenta uma maçã vermelha]
h −− Robô, o que é isso?
r −− Isto é uma maçã
h −− Muito bom!
...

Os experimentos demonstraram como é possível transformar um problema de aprendizado de máquina em um problema de colaboração entre robôs e seres humanos, empregando as habilidades sociais naturais dos seres humanos para ensinar um robô.

Finalmente, um exame dos resultados obtidos mostram que a arquitetura com o ARR é capaz controlar um robô sociável em uma interação controlada, ainda que simples. Adicionalmente, os resultados evidenciam que a atenção compartilhada é uma habilidade social fundamental para o desenvolvimento social. Os resultados mostram também como um diálogo colaborativo pode permitir que um robô aprenda conceitos sobre objetos ou eventos importantes do ambiente.

Portanto, pode-se concluir que o mecanismo de aprendizagem por tutelagem, assim como a arquitetura robótica, constituem uma ferramenta promissora para controlar robôs sociáveis durante interações em um ambiente social real e controlado.

6.5 Considerações Finais

Este Capítulo apresentou os ambientes na qual o algoritmo de aprendizado proposto foi avaliado. O algoritmo de aprendizado proposto foi avaliada em diversos experimentos executados em diferentes domínios de aplicação, demonstrando a portabilidade.

Os resultados obtidos com os experimentos demonstram o bom funcionamento de sistemas que empregam o mecanismo de aprendizado proposto com a representação apresentada.

A proposta possui vantagens quando se coloca a questão tempo e consumo de recursos como requisitos na resolução de um problema.

Tais resultados também mostram que a potencialidade do algoritmo como mecanismo de aprendizado da arquitetura. Os resultados obtidos mostram que a arquitetura é capaz de exibir comportamentos apropriados e é capaz de aprender a partir de interações sociais.

No próximo Capítulo é apresentada uma discussão geral sobre esta pesquisa. Adicionalmente, são feitas as considerações finais deste trabalho.

CAPÍTULO

7

Conclusões e trabalhos futuros

O principal objetivo que norteou o desenvolvimento do presente trabalho foi o de construir um sistema interativo para tornar uma máquina, seja ela qual for, no caso uma cabeça robótica, com possibilidades de aprender através da interação com um humano. Buscando dar um passo nesta direção, este trabalho foi focalizado no estudo e desenvolvimento de um mecanismo de aprendizado por reforço relacional para ser inserido em uma arquitetura robótica. O aprendizado por reforço relacional representa na forma relacional, usando lógica de primeira ordem, o aprendizado obtido através da interação direta com o ambiente. É uma técnica bem interessante no campo de robótica, pois, em geral, não se dispõe do modelo do ambiente e se requer econômia de recursos utilizados.

A utilização do aprendizado por reforço relacional como objeto de estudo em pesquisas ainda é muito pequeno. No entanto, essas pesquisas têm crescido em número e abrangência nos últimos anos. Do ponto de vista de processos de desenvolvimento, ainda há muito o que ser explorado para desenvolver técnicas de aprendizado com esse tipo de representação em todo o campo de Interação Humano-Robô (IHR).

O estudo realizado sobre o tema "Aprendizado por Reforço Relacional", fundamental para a realização de todas as atividades envolvidas, foi importante para realização de uma proposta de alteração no algoritmo TG, que culminou no algoritmo TGE, o qual foi incorporado em uma arquitetura de controle de uma cabeça robótica. A arquitetura foi avaliada no contexto de um problema real não trivial: o aprendizado da atenção compartilhada. O algoritmo TGE apresenta como vantagens rápida convergência, não utilização de memória auxiliar, rapidez na busca por solução e adaptação.

Embora simples, o TGE mostrou-se como uma boa proposta para o módulo de aprendizado da arquitetura robótica, o que foi observado nos resultados obtidos nos testes realizados com a cabeça

robótica. A eficiência do método TGE proposto foi comparada com outros métodos de aprendizado por reforço, tais como, o aprendizado por contingência e o algoritmo Q-learning. Os resultados comparativos, realizados com o simulador de interações sociais, mostraram que o método TGE teve um desempenho superior.

Entretanto, devido a forma na qual o sistema foi construído, existem algumas limitações apontadas a seguir. A primeira delas, refere-se ao sistema de visão computacional da arquitetura robótica, que foi utilizado na cabeça robótica. Este sistema apresenta diversas deficiências relativas à biblioteca responsável por detectar faces humanas e estimar a posição da cabeça humana (biblioteca Watson). O problema com a biblioteca Watson está no fato que os algoritmos disponíveis para detecção de faces e estimação da posição da cabeça necessitam de aprimoramentos pois possuem limitações. O problema da luminosidade está no fato da não formação do modelo (da não deteccção da face humana) quando o ambiente possui muita luz. Esse fato fez com que os testes se prolongassem por meses.

A segunda limitação, refere-se à forma como foi utilizada a representação do ambiente. A operação de soma dos valores binários é realizada sobre valores inteiros positivos e este fato possui uma limitação com relação ao maior valor inteiro possível de ser representado.

Um terceira e última limitação que foi observado durante o desenvolvimento deste trabalho, refere-se à questão do tamanho da árvore utilizada no algoritmo TGE. A medida que os elementos do ambiente aumentam ela fica maior e o custo de acesso a essa estrutura fica alto. Este fato aponta para a necessidade de se utilizar métodos de *prunning* para tentar diminuir o tamanho da árvore.

Finalmente, vale ressaltar novamente que até o momento, segundo o levantamento bibliográfico feito pelo pesquisador e pelo grupo de pesquisa, não existem trabalhos similares dentro da robótica sociável, sendo desenvolvidos no Brasil e que os autores acreditam que apesar de estarmos muito longe de alcançarmos o objetivo principal que norteou este trabalho, eles contribuiram de alguma forma para que este objetivo venha a ser alcançado algum dia dentro da comunidade científica da área de robótica.

7.1 Trabalhos futuros

A realização deste trabalho criou diversas perspectivas para a realização de trabalhos futuros como continuação desta pesquisa. Esses trabalhos envolvem:

- Inserção de mecanismo de pré e pós poda para reduzir o crescimento da árvore.
- Estudo de formas de representação relacional que possam ser facilmente adaptados a arquitetura robótica.
- Após realizar os testes com o simulador de Interações Sociais, notou-se certos comportamentos nos quais os algoritmos de Aprendizado por Reforço tiveram dificuldades para construir

uma boa política. Isso é devido ao fato do ambiente, ao qual ele foi exposto, ser conhecido não como um problema MDP (Markov Decision Process), mas por ser do tipo POMDP (Partially Observable Markov Decision Problem). Um problema POMDP é conhecido como um problema MDP, mas difere pelo fato do agente não ter conhecimento total do ambiente. Esse fato abre novas perspectivas de investigações, tais como, fatores como adaptação do método POMDP à arquitetura proposta e a sua forma de representação de conhecimento.

- Devido as limitações das técnicas existentes na biblioteca Watson é necessário aprimorar a técnica empregada para a detecção da face e estimativa da posição da face humana.
- Estudos sobre aprendizado por transferência, no qual o conhecimento armazenado para um problema mais simples seja usado pelo agente em um problema mais complexo mas correlato.

7.2 Produção científica

Nesta seção, são citados os trabalhos que foram gerados durante a realização do presente trabalho.

Artigos completos submetidos a periódicos

1. Silva, R. R., Policastro, C. A., Romero, R. A. F. "Comparison of Techniques for Learning in a Robotics Architecture for Sociable Robots Using Relational Representation". *RAS-Springer*. Submited.
2. Silva, R. R., Policastro, C. A., Pais, G. D., Munhoz, V. R., Romero, R. A. F., Zuliani, G., Pizzolato, E. "Learning by Tutelage and Multimodal Interface in Sociable Robots". *Learning and Nonlinear Models (L&NLM)*. Submited.
3. Silva, R. R., Yokoyama, R. S., Romero, R. A. F. "Avaliação do Desempenho da Utilização de Threads em user level em Linux". *RITA*. Submited.

Artigos completos em anais de eventos

1. Silva, R. R., Policastro, C. A., Romero, R. A. F. "An Enhancement of Relational Reinforcement Learning" . In: *Proceedings of 2008 International Joint Conference on Neural* (WCCI'08'). (Hong Kong, June 1 - 6, 2008). IEEE, Piscataway, NJ, p. 2056-2061.
2. Policastro,C.A., Zuliani,G., Silva,R.R., Munhoz,V.R., Romero,R.A.F. "Hybrid Knowledge Representation Applied to the Learning of the Shared Attention". *IEEE World Congress on Computational Intelligence*(WCCI'08'). (Hong Kong, June 1 - 6, 2008). IEEE, Piscataway, NJ, p. 1580-1585.

Artigos resumidos em anais de eventos

1. Silva, R. R., Policastro, C. A., Romero, R. A. F. “Uma extensão do aprendizado por reforço relacional”. In: *Proceedings of The XIII Brazilian Symposium on Multimedia and the Web*. Palmas, Tocantins, TO. II Mostra Cientifica, 2008. p. .

Artigos completos submetidos a anais de eventos

1. Silva, R. R., Policastro, C. A., Romero, R. A. F. “Relational Reinforcement Learning applied to Shared Attention” . In: *Proceedings of 2009 International Joint Conference on Neural* (IJCNN’09’). (Georgia, June 1 - 6, 2009). IEEE, Piscataway, NJ. Submitted.

Referências Bibliográficas

BELLMAN, R. E. *An introduction to artificial intelligence: Can computers think?* Boyd & Fraser Pub. Co, 1978.

BERTSEKAS, D. P. *Dynamic programming: deterministic and stochastic models*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1987.

BJÖRNE, P.; BALKENIUS, C. A model of attentional impairments in autism: first steps toward a computational theory. *Cognitive Systems Research*, v. 6, n. 3, p. 193−−204, 2005.

BLOCKEEL, H.; RAEDT, L. D. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, v. 101, n. 1-2, p. 285−−297, 1998.

BORSATO, F.; FIGUEIREDO, M. Aprendizagem por reforço em redes neurais multicamadas aplicadas em controle autônomo. In: *Simpósio Brasileiro de Automação Inteligente*, 2007.

BOSA, C. Atenção compartilhada e identificação precoce do autismo. *Psicologia: Reflexão e Crítica*, v. 15, n. 1, p. 77−−88, 2002.

BREAZEAL, C. *Designing sociable robots*. MIT Press, 2002.

BREAZEAL, C. Toward sociable robots. *Robotics and Autonomous Systems*, v. 42, n. 3−4, p. 167−−175, 2003.

BREAZEAL, C. Social interactions in hri: the robot view. *IEEE Transactions on Man and Cybernetics , Part C: Applications and Reviews*, v. 34, n. 2, p. 181−−186, 2004.

BREAZEAL, C.; SCASSELLATI, B. A context-dependent attention system for a social robot. In: *International Joint Conference on Artificial Intelligence*, 1999, p. 1146−−1153.

BREIMAN, L.; FRIEDMAN, J.; STONE, C. J.; OLSHEN, R. *Classification and regression trees*. Wadsworth International Group, Belmont, California, USA: Chapman & Hall/CRC, 1984.

CARLSON, E.; TRIESCH, J. A computational model of the emergence of gaze following. In: *8th Neural Computation Workshop (NCPW8)*, 2003.

CARPENTER, G. A.; GROSSBERG, S. Art 2: self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, v. 26, p. 4919−−4930, 1987.

CASTIÑEIRA, M. I. *Aprendizado de máquina por exemplo usando Árvore de decisão*. Tese de Doutoramento, Universidade de São Paulo, 1990.

CELIBERTO, L. A.; RIBEIRO, C. H. C.; COSTA, A. H. R.; BIANCHI, R. A. C. Heuristic reinforcement learning applied to robocup simulation agents. *RoboCup 2007: Robot Soccer World Cup XI*, v. 5001, p. 220–227, 2008.

CHAPMAN, D.; KAELBLING, L. P. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-91)*, 1991, p. 726−−731.

CHARNIAK, E.; MCDERMOTT, D. *Introduction to artificial intelligence*. Addison Wesley Publishing Company, 1991.

CRITES, R. H.; BARTO, A. G. Improving elevator performance using reinforcement learning. In: *Advances in Neural Information Processing Systems 8*, MIT Press, 1996, p. 1017−−1023.

DABNEY, W.; MCGOVERN, A. Utile distinctions for relational reinforcement learning. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007, p. 738−−743.

DAUTENHAHN, K. Getting to know each other - artificial social intelligence for autonomous robots. *Robotics and Autonomous Systems*, v. 16, p. 333−−356, 1995.

DAUTENHAHN, K. I could be you - the phenomenological dimension of social understanding. *The Cybernetics and Systems Journal*, v. 28, n. 5, 1997.

DAUTENHAHN, K. The art of designing socially intelligent agents - science, fiction and the human in the loop. *Applied Artificial Intelligence Journal*, v. 12, n. 7−8, p. 573−−617, 1998.

DAUTENHAHN, K.; BILLARD, A. Bringing up robots or−the psychology of socially intelligent robots: From theory to implementation. In: *Autonomous Agents*, 1999.

DEÁK, G.; FASEL, I.; MOVELLAN, J. The emergence of shared attention: Using robots to test developmental theories. In: BALKENIUS, C.; ZLATEV, J.; KOZIMA, H.; DAUTENHAHN, K.; BREAZEAL, C., eds. *First International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, Lund: Lund University, 2001, p. 95−−104.

DEÁK, G.; TRIESCH, J. Origins of shared attention in human infants. In: FUZITA, K.; ITAKURA, S., eds. *Diversity of Cognition (In Press)*, Kyoto University Press, 2005a, p. 67−−74.

DEÁK, G.; TRIESCH, J. Origins of shared attention in human infants. In: FUZITA, K.; ITAKURA, S., eds. *Diversity of Cognition (In Press)*, Kyoto University Press, 2005b, p. 67−−74.

DRIESSENS, K. Relational reinforcement learning. In: *EASSS '01: Selected Tutorial Papers from the 9th EC-CAI Advanced Course ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School on Multi-Agent Systems and Applications*, London, UK: Springer-Verlag, 2001, p. 271–280.

DRIESSENS, K. Relational instance based regression for relational reinforcement learning. In: *In Proceedings of the 20th International Conference on Machine Learning*, AAAI Press, 2003, p. 123−−130.

DRIESSENS, K. *Relational reinforcement learning*. Tese de Doutoramento, Katholieke Universiteit Leuven, 2004.

DRIESSENS, K.; DŽEROSKI, S. Integrating guidance into relational reinforcement learning. *Machine Learning*, v. 57, n. 3, p. 271−−304, 2004.

DUBE, W.; MCDONALD, R.; MANSFIELD, R.; HOLCOMB, W.; AHEARN, W. Toward a behavioral analisys of joint attention. *The Behavior Analyst*, v. 27, n. 2, p. 197−−207, 2004.

DUNG, L. T.; KOMEDA, T.; TAKAGI, M. Reinforcement learning for pomdp using state classification. *Applied Artificial Intelligence*, v. 22, p. 761–779, 2008.

FARIA, G. *Explorando o potencial de algoritmos de aprendizado com reforço em robôs móveis*. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação - USP, 2000.

FASEL, I.; DEAK, G.; TRIESCH, J.; MOVELLAN, J. Combining embodied models and empirical research for understanding the development of shared attention. In: *2nd International Conference on Development and Learning*, 2002, p. 21−−27.

FERRO, M. *Aquisição de conhecimento de conjuntos de exemplos no formato atributo valor utilizando aprendizado de máquina relacional*. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação - USP, 2002.

GÄRTNER, T.; DRIESSENS, K.; RAMON, J. Graph kernels and gaussian processes for relational reinforcement learning. In: *Machine Learning*, Springer, 2003, p. 146−−163.

GOCKLEY, R.; FORLIZZI, J.; SIMMONS, R. Natural person-following behavior for social robots. In: *Proceeding of the ACM/IEEE international conference on Human-robot interaction, Arlington, Virginia, USA*, ACM Press, 2007, p. 17−−24.

GOLD, K.; SCASSELLATI, B. A bayesian robot that distinguishes "self" from "other". In: *To appear in the Proceedings of the 29th Annual Meeting of the Cognitive Science Society (CogSci'07)*, 2007.

GOSAVI, A. A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis. *Mach. Learn.*, v. 55, n. 1, p. 5–29, 2004.

GREENWALD, A. Reinforcement learning. Lecture note number 12 on Spring 2005 of Artificial Intelligence, 2005.

HAUGELAND, J. *Artificial intelligence: the very idea*. Cambridge, MA, USA: Massachusetts Institute of Technology, 1985.

HAYKIN, S. *Neural networks - a comprehensive foundation*. Prentice Hall, 1999.

ITTI, L.; KOCH, C.; NIEBUR, E. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, n. 11, p. 1254−−1259, 1998.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. P. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, v. 4, p. 237−−285, 1996.

KANDA, T.; SATA, R.; SAIWAKI, N.; ISHIGURO, H. Friendly social robot that understands human's friendly relationships. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2004)*, 2004, p. 2215−−2222.

KAPLAN, F. Taming robots with clicker training: A solution for teaching complex behaviors. In: *the European Workshop on Learning Robots*, 2001.

KAPLAN, F.; HAFNER, V. The challenges of joint attention. In: BERTHOUZE, L.; KOZIMA, H.; PRINCE, C.; SANDINI, G.; STOJANOV, G.; METTA, G.; BALKENIUS, C., eds. *4th International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic System*, 2004, p. 67−−74.

KURZWEIL, R. *The age of intelligent machines*. The MIT Press, 1990.

LANGLEY, P. *Elements of machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995.

LAU, B.; TRIESCH, J. Learning gaze following in space: a computational model. In: *3rd International Conference for Development and Learning, ICDL'04*, 2004.

LIB, L. *Lti image processing library - developerś guide*. Relatório Técnico, Lehrstuhl fuer Technische Informatik - Aachen University of Technology, 2003.

LOCKERD, A.; BREAZEAL, C. Tutelage and socially guided robot learning. In: *International Conference on Intelligent Robots and Systems (IROS'2004)*, 2004, p. 3475−−3480.

MAHADEVAN, S.; MARCHALLECK, N.; DAS, T. K.; GOSAVI, A. Self-improving factory simulation using continuous-time average-reward reinforcement learning. In: *Proceedings of the Fourteenth International Conference on Machine Learning*, Morgan Kaufmann, 1997, p. 202–210.

MAROM, Y.; HAYES, G. *Attention and social situatedness for skill acquisition*. Relatório Técnico EDI-INF-RR-0069, University of Edinburgh, 2001.

MICHAUD, F.; CARON, S. Roball-an autonomous toy-rolling robot. In: *Workshop on Interactive Robot Entertainment*, 2000.

MITCHELL, T. *Machine learning*. McGraw Hill, 1997.

MONTEIRO, S. T.; RIBEIRO, C. H. C. Desempenho de algoritmos de aprendizagem por reforço sob condições de ambiguidade sensorial em robótica móvel. *Sba: Controle & Automação Sociedade Brasileira de Automatica*, v. 15, p. 320 – 338, 2004.
Disponível em: `http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-17592004000300008&nrm=iso`

MORENCY, L.-P. *User guide - head tracking and gesture recognition library*. Relatório Técnico, MIT Department of Electrical Engineering and Computer Science, 2007.

MORENCY, L.-P.; SUNDBERG, P.; DARREL, T. Pose estimation using 3d view-based eigenspaces. In: *Analysis and Modeling of Faces and Gestures (AMFG'03)*, IEEE, 2003, p. 45−−52.

MORIK, K.; WROBEL, S.; KIETZ, J.-U.; EMDE, W. *Knowledge acquisition and machine learning: Theory, methods, and applications*. San Francisco, CA, USA: Academic Press, 1993.

NAGAI, Y.; HOSODA, A.; ASADA, M. A constructive model for the development of joint attention. *Connection Science*, v. 15, n. 4, p. 211−−229, 2003.

NILSSON, N. J. *Principles of artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1980.

NILSSON, N. J. *Artificial intelligence: a new synthesis*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

NUANCE *Introduction to the nuance system*. Relatório Técnico, Nuance Communications, Inc., 2001.

OTTERLO, V. A survey of reinforcement learning in relational domains. *CTIT Technical Report, TR-CTIT-05-31*, 2005.

POLICASTRO, C. A. *Arquitetura robótica inspirada na análise do comportamento*. Tese de Doutoramento, Universidade de São Paulo, 2008.

POOLE, D.; MACKWORTH, A.; GOEBEL, R. *Computational intelligence: A logical approach*. Oxford University Press, 1998.

QUINLAN, J. R. Induction of decision trees. *Mach. Learn.*, v. 1, n. 1, p. 81–106, 1986.

RIBEIRO, C. H. C. Aprendizado por reforço. V Escola de Redes Neurais, Promoção: Conselho Nacional de Redes Neurais, 1999.

RICH, E.; KNIGHT, K. *Artificial intelligence*. McGraw-Hill Science/Engineering/Math - 2 Sub edition, 1991.

ROBINS, B.; DICKERSON, P.; STRIBLING, P.; DAUTENHAHN, K. Robot-mediated joint attention in children with autism: A case study in robot-human interaction. *Interaction Studies*, v. 5, n. 2, p. 161−−198, 2004.

RODRIGUES, F.; GOMES, H. Applying a visual attention mechanism to the problem of traffic sign recognition. In: *SIBGRAPI*, 2002, p. 415.

RUMMERY, G. A.; NIRANJAN, M. *On-line q-learning using connectionist systems*. Relatório Técnico, 1994.

RUSSELL, S.; NORVIG, P. *Artificial intelligence - a modern approach, second edition*. Prentice-Hall, 2003.

SALICHS, M.; BARBER, R.; KHAMIS, A.; MALFAZ, M.; GOROSTIZA, J.; PACHECO, R.; RIVAS, R.; CORRALES, A.; DELGADO, E. Maggie: A robotic platform for human-robot social interaction. In: *IEEE International Conference on Robotics, Automation and Mechatronics (RAM 2006). Bangkok. Thailand*, 2006.

SCASSELLATI, B. Investigating models of social development using a humanoid robot. In: WEBB, B.; CONSI, T., eds. *Biorobotics, MIT Press*, 2000.

SCASSELLATI, B. Using social robots to study abnormal social development. In: BERTHOUZE, L.; KAPLAN, F.; KOZIMA, H.; YANO, H.; KONCZAK, J.; METTA, G.; NADEL, J.; SANDINI, G.; STOJANOV, G.; BALKENIUS, C., eds. *Fifth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Nara, Japan*, 2001, p. 11−−14.

SCHEEFF, M. Experiences with sparky: A social robot. In: *Workshop on Interactive Robot Entertainment*, 2000.

SCHWARTZ, A. A reinforcement learning method for maximizing undiscounted rewards. In: *Machine Learning: Proceedings of the Tenth International Conference*, SanMateo, CA, USA: Morgan Kaufmann, 1993, p. 298−−305.

SERIO, T.; ANDERY, M.; GIOIA, P.; MICHELETO, N. *Controle de estímulos e comportamento operante: uma (nova) introdução*. EDUC - Editora da PUC - SP, 2004.

SILVA, R. R. ; POLICASTRO, C. A. . R. R. A. F. An enhancement of relational reinforcement learning. In: *Proceedings of 2008 International Joint Conference on Neural*, Hong Kong: IEEE, 2008a, p. 2056–2061.

SILVA, R. R. ; POLICASTRO, C. A. . R. R. A. F. Relational reinforcement learning applied to shared attention. In: *Proceedings of 2009 International Joint Conference on Neural*, Atlanta, Georgia, USA: IEEE, 2008b, p. Submited.

SILVA, L. M. O. *Uma aplicação de Árvores de decisão, redes neurais e knn para a identificação de modelos arma não-sazonais e sazonais*. Tese de Doutoramento, Pontifícia Universidade Católica do Rio de Janeiro, 2005.

SINGH, S.; BERTSEKAS, D. Reinforcement learning for dynamic channel allocation in cellular telephone systems. In: *In Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference*, MIT Press, 1997, p. 974–980.

SINGH, S. P.; SUTTON, R. S. Reinforcement learning with replacing eligibility traces. *Mach. Learn.*, v. 22, n. 1-3, p. 123–158, 1996.

SLANEY, J.; THIÉBAUX, S. Blocks world revisited. *Journal of Artificial Intelligence*, v. 125, p. 119–153, 2001.

SMITH, L.; ULVUND, S. The role of joint attention in later development among preterm children: Linkages between early and middle childhood. *Social Development*, v. 12, n. 2, 2003.

SNEDECOR, G.; COCHRAN, W. *Statistical methods*. Iowa University Press, 1989.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. MIT Press, 1998.
Disponível em: `http://www.cs.ualberta.ca/%7Esutton/book/ebook/the-book.html`

SZEPESVÁRI, C.; LITTMAN, M. L. *Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms*. Relatório Técnico, Providence, RI, USA, 1996.

TADEPALLI, P.; OK, D. *H learning: A reinforcement learning method to optimize undiscounted average reward.* Relatório Técnico, 1994.

TESAURO, G. Temporal difference learning and td-gammon. *Commun. ACM*, v. 38, n. 3, p. 58–68, 1995.

TRIESCH, J.; TEUSCHER, C.; DEÁK, G.; CARLSON, E. Gaze following: why (not) learn it? *Developmental Science*, v. 2, n. 9, p. 125−−157, 2006.

TURING, A. *Computing machinery and intelligence*. Mind, 1950.

WATKINS, C. J. C. H. *Learning from delayed rewards*. Relatório Técnico, 1989.

WEBB, B. What does robotics offer animal behaviour? *Animal Behaviour*, v. 60, p. 545−−558, 2000.

WHALEN, C.; SCHREIBMAN, L. Joint attention training for children with autism using behavior modification procedures. *Journal of Child Psychology and Psychiatry*, v. 44, n. 3, p. 456−−468, 2003.

WINSTON, P. H. *Artificial intelligence*. Addison Wesley, 1998.

ZHANG, W.; DIETTERICH, T. G. A reinforcement learning approach to job-shop scheduling. In: *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 1995, p. 1114–1120.