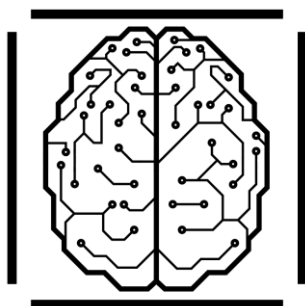




INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

**Área Departamental de Engenharia de Electrónica
e Telecomunicações e de Computadores**



Aprendizagem por Reforço com Recursos Limitados

José Miguel Carvalho Ramada
(Licenciado)

Trabalho Final de Mestrado para obtenção do grau de
Mestre em Engenharia de Redes de Comunicação e Multimédia

Orientador:

Professor Doutor Luís Filipe Graça Morgado

Júri:

Presidente: Professor Doutor Paulo Manuel Trigo Cândido da Silva

Vogal: Professor Doutor Arnaldo Joaquim Castro Abrantes

Janeiro de 2017

Resumo

A aprendizagem por reforço define um paradigma de aprendizagem comportamental onde o processo de aquisição de conhecimento por parte de um sistema é realizado de forma autónoma. A partir da interacção com o ambiente, são usadas estratégias de selecção de acção para maximizar uma recompensa cumulativa ao longo do tempo. Nesse sentido, este paradigma detém um grande potencial e relevância em múltiplas áreas de aplicação, como agentes inteligentes, aplicações sociais, financeiras, jogos, multimédia, robótica, veículos autónomos, entre outras.

Contudo, a possibilidade de aplicação geral deste método de aprendizagem é limitada pela elevada complexidade computacional, que se traduz na utilização extensiva de recursos. Esta complexidade inerente ao processo de aprendizagem, dificulta uma aprendizagem eficiente, sobretudo em tempo real, pelo que este trabalho pretende contribuir com um levantamento de propostas e abordagens que se aproximem de uma solução viável para este problema em diferentes cenários de operação.

É objectivo desta dissertação o estudo de métodos de aprendizagem por reforço passíveis de operação em contextos de restrição de recursos computacionais, bem como, a definição das condições em que essa operação é viável. Neste sentido, propõe-se a selecção de alguns dos métodos mais promissores no contexto de operação com recursos limitados e efectuar uma implementação de cada um destes, primeiramente sobre uma plataforma de simulação e, posteriormente, numa plataforma física.

A concretização da plataforma física será realizada sob a forma de um agente robótico simples para teste e obtenção de resultados práticos. Em contexto de operação sobre condições específicas, aspectos de eficiência e eficácia serão analisados e comparados entre os diferentes métodos.

Palavras-chave: agentes inteligentes, inteligência artificial, aprendizagem por reforço, aprendizagem com recursos limitados, processos de decisão de Markov, modelos de agentes híbridos, política comportamental.

Abstract

Reinforcement learning defines a paradigm of behavioural learning where the process of knowledge acquisition, by a system, is carried out independently. From the interaction with the environment, action selection strategies are used to maximize a cumulative reward over time. In this sense, this paradigm has great potential and relevance in multiple application areas such as intelligent agents, social applications, games and multimedia applications, robotics, autonomous vehicles or financial investments, among others.

However, the possibility of general application of this learning method expresses its main feature: high computational complexity; which results in extensive use of computational resources. This inherent complexity in the learning process hinders an effective learning, particularly in real time. This work aims to contribute to a survey of proposals and approaches to reach a viable solution in different scenarios of operation.

This thesis aims at the study of reinforcement learning methods capable of operation in computational resource constrained contexts, as well as defining the conditions under which the operation is feasible. Thus, this dissertation proposes a selection of some of the most promising methods with a implementation of each; first on a simulation platform and later in physical platform.

The embodiment of the physical platform will be made through a simple robot. In the context of specific operating conditions, efficiency and efficacy issues will be analysed and compared between the different methods.

Keywords: intelligent agents, artificial intelligence, reinforcement learning, learning with limited resources, Markov decision processes, hybrid agent models, behavioural policy.

Agradecimentos

Primeiramente, agradeço ao Professor Doutor Luís Filipe Morgado. Não existem palavras suficientes que descrevam o apoio, dedicação, disponibilidade e, acima de tudo, a modéstia com que partilhou o seu conhecimento e se prontificou a orientar-me nesta minha jornada. O meu obrigado por tantas tardes passadas a indicar-me o caminho.

Quero também prestar os meus agradecimentos ao Instituto Superior de Engenharia de Lisboa e a todos os docentes que de alguma forma contribuíram com o seu conhecimento e me forneceram algumas das ferramentas para o futuro.

Um abraço aos meus colegas de curso, por os momentos e conversas partilhadas durante estes anos. Um obrigado especial ao Diogo Monteiro por todos estes anos de companheirismo e de entreajuda onde, mesmo quando as nossas visões colidiam, foi sempre bom ver com outra perspectiva.

Ao Vicente Vieira, amigo de longa data, que sempre me apoiou e se mostrou disponível para ajudar. O meu obrigado pelo suporte e lealdade.

À minha mais que tudo, por todo o amor, apoio e paciência demonstrados ao longos destes anos. Sem ti, tudo teria sido muito mais difícil.

A todos os elementos da minha Família, onde cada um me ajudou, à sua maneira, a tornar-me a pessoa que sou agora. Aos que partiram, hoje sei que “se vi mais longe foi por estar de pé sobre ombros de gigantes”.

Por fim, agradeço também a todos, cujo o nome não mencionei, mas que de alguma forma contribuíram, directa ou indirectamente, nesta fase da minha vida.

O meu muito obrigado a todos.

Índice Geral

ÍNDICE DE TABELAS.....	1
ÍNDICE DE FIGURAS	3
1 INTRODUÇÃO	9
1.1 MOTIVAÇÃO.....	9
1.2 OBJECTIVOS.....	10
1.3 ORGANIZAÇÃO DO DOCUMENTO	11
2 ENQUADRAMENTO TEÓRICO.....	13
2.1 INTELIGÊNCIA ARTIFICIAL.....	13
2.2 AGENTES INTELIGENTES.....	13
2.2.1 <i>Arquitecturas Reactivas</i>	14
2.2.2 <i>Arquitecturas Híbridas</i>	16
2.3 APRENDIZAGEM POR REFORÇO	17
2.3.1 <i>Aprender através de recompensas</i>	17
2.3.2 <i>Aprendizagem por diferença temporal</i>	22
2.3.3 <i>Método de aprendizagem SARSA</i>	24
2.3.4 <i>Método de aprendizagem Q-Learning</i>	25
2.3.5 <i>Método de aprendizagem Dyna-Q</i>	26
2.3.6 <i>Método de aprendizagem Q-Learning com Memória Episódica</i>	29
2.4 REDES NEURONAIS ARTIFICIAIS	30
2.4.1 <i>Redes Feed-Forward</i>	32
2.4.2 <i>Aprendizagem em redes neuronais artificiais</i>	32
2.4.3 <i>Técnicas de optimização estocástica</i>	35
3 MÉTODOS ALTERNATIVOS DE APRENDIZAGEM POR REFORÇO	39
3.1 APRENDIZAGEM POR REFORÇO HIERÁRQUICA	39
3.1.1 <i>Feudal Reinforcement Learning</i>	39
3.2 APRENDIZAGEM ACELERADA POR HEURÍSTICA.....	41
3.2.1 <i>Heuristically Accelerated Q-Learning (HAQL)</i>	41
3.3 APRENDIZAGEM BASEADA EM CASOS (CASE BASED R.L.).....	42
3.3.1 <i>Case Based Heuristically Accelerated Q-Learning (CB-HAQL)</i>	42
3.4 APRENDIZAGEM MULTINÍVEL.....	43
3.4.1 <i>Arquitectura Deep-Q</i>	44
3.5 CONCLUSÕES	45

4	O PROBLEMA DA LIMITAÇÃO DE RECURSOS	47
4.1	VISÕES DE RACIONALIDADE	47
4.2	RACIONALIDADE LIMITADA.....	49
4.2.1	<i>Satisfação</i>	50
4.2.2	<i>Heurísticas rápidas e frugais</i>	51
4.3	CONSIDERAÇÕES SOBRE RACIONALIDADE LIMITADA.....	53
4.4	TÉCNICAS A ABORDAR.....	54
4.4.1	<i>Representação de estado</i>	55
4.4.2	<i>Simplificação do mecanismo de aprendizagem</i>	56
4.5	EXEMPLO APLICAÇÃO - TD-GAMMON.....	57
4.6	CONCLUSÕES	58
5	ABORDAGENS PROPOSTAS.....	59
5.1	APRENDIZAGEM COM UTILIZAÇÃO DE HEURÍSTICA.....	59
5.1.1	<i>Integração de reactividade</i>	59
5.1.2	<i>Método Heuristic Sparse Learning - HSL</i>	61
5.1.3	<i>Arquitectura híbrida com camada reactiva com memória</i>	63
5.2	APRENDIZAGEM COM ABSTRACÇÃO DE ESTADO.....	69
5.2.1	<i>Agente adaptativo Deep-Q adaptado</i>	69
5.2.2	<i>Modelo geral de aprendizagem com abstracção de estado</i>	70
5.2.3	<i>Aprendizagem hierárquica quadtree</i>	71
5.2.4	<i>Aprendizagem com memória selectiva de instâncias</i>	73
5.3	APRENDIZAGEM COM RECURSOS MUITO LIMITADOS	77
5.3.1	<i>Representação limitada de estado</i>	77
5.3.2	<i>Vertente robótica</i>	79
5.4	CONCLUSÕES	80
6	CONCRETIZAÇÃO.....	81
6.1	ARQUITECTURA GERAL DE AGENTE	81
6.2	APRENDIZAGEM POR REFORÇO GERAL.....	82
6.2.1	<i>Estrutura base – biblioteca AprendRef</i>	82
6.2.2	<i>Implementação agente Q-Learning</i>	85
6.2.3	<i>Implementação agente Dyna-Q</i>	87
6.2.4	<i>Implementação agente com memória episódica</i>	88
6.3	APRENDIZAGEM COM UTILIZAÇÃO DE HEURÍSTICA.....	90
6.3.1	<i>Integração de reactividade</i>	90
6.3.2	<i>Método Heuristic Sparse Learning - HSL</i>	91
6.3.3	<i>Arquitectura híbrida com camada reactiva com memória</i>	93

6.4	APRENDIZAGEM POR ABSTRACÇÃO DE ESTADO	94
6.4.1	Agente adaptativo Deep-Q adaptado	94
6.4.2	Agente adaptativo hierárquico quadtree	98
6.4.3	Agente adaptativo com memória selectiva de instâncias	100
6.5	APRENDIZAGEM COM RECURSOS MUITO LIMITADOS	101
6.5.1	Agente adaptativo com representação limitada de estado	101
6.5.2	Vertente robótica	102
6.6	CONCLUSÕES	107
7	RESULTADOS EXPERIMENTAIS.....	109
7.1	ROTINAS DE TESTE	109
7.2	CARACTERIZAÇÃO DO AMBIENTE	110
7.3	RESULTADOS DAS ROTINAS	111
7.3.1	Integração de Reactividade – Rotina 1.....	111
7.3.2	Integração de Reactividade – Rotina 2.....	113
7.3.3	Aprendizagem hierárquica Quadtree – Rotina 1.....	114
7.3.4	Aprendizagem Hierárquica Quadtree – Rotina 2.....	116
7.3.5	Aprendizagem com Memória Selectiva de Instâncias (MSI) – Rotina 1.....	117
7.3.6	Aprendizagem MSI & Hierárquica Quadtree – Rotina 1	119
7.3.7	Aprendizagem MSI Integr. react. & Integr. react. simples – Rotina 1.....	121
7.3.8	Aprendizagem MSI Integr. react. & Integr. react. simples – Rotina 2.....	122
7.4	CONSIDERAÇÕES SOBRE OS RESULTADOS EXPERIMENTAIS	124
7.4.1	Diferenças entre os métodos Dyna-Q e Memória Episódica	124
7.4.2	Integração de reactividade.....	125
7.4.3	Agente adaptativo hierárquico quadtree.....	126
7.4.4	Agente adaptativo com memória selectiva de instâncias.....	126
7.4.5	Agente Deep-Q adaptado e treino offline.....	127
7.4.6	Método HSL e Agente com camada reactiva com memória.....	129
7.4.7	Representação limitada de estado	130
8	CONCLUSÃO.....	133
8.1	TRABALHO FUTURO	135
	BIBLIOGRAFIA	139

APÊNDICE A – DETALHE DE ARQUITECTURA	141
AGENTE PROSPECTOR.....	141
ARQUITECTURA REACTIVA	142
ARQUITECTURA ADAPTATIVA BASE.....	143
ARQUITECTURA HÍBRIDA SIMPLES COM INTEGRAÇÃO DE REACTIVIDADE	144
ESTRUTURA COMPOSTA DA ARQUITECTURA COM INTEGRAÇÃO DE REACTIVIDADE.....	145
ARQUITECTURA ADAPTATIVA HIERÁRQUICA QUADTREE	145
ARQUITECTURA ADAPTATIVA COM MEMÓRIA SELECTIVA DE INSTÂNCIAS.....	146
ARQUITECTURA DEEP-Q.....	147
ARQUITECTURA HÍBRIDA DE CAMADA REACTIVA COM MEMÓRIA.....	148
ORGANIZAÇÃO GERAL.....	149
APÊNDICE B – VERTENTE ROBÓTICA.....	151
MONTAGEM TÉCNICA	151
<i>Micro Switch</i>	151
<i>Módulo ultra-sons HC-SR04</i>	152
<i>Módulo luminoso TSL2561</i>	154
<i>Controlador Motores TB6612FNG</i>	155
<i>Impressão 3D de suportes</i>	156
<i>Montagem do agente robótico</i>	158
ARQUITECTURA PRA - PLATAFORMA DE ROBÓTICA DE AGENTE	160
APÊNDICE C – FERRAMENTAS	161
LINGUAGEM PYTHON	161
PYGAME	161
AMBIENTE DE DESENVOLVIMENTO.....	162

Índice de Tabelas

Tabela 1 - Estudo do macro estado em função da monotonia da variação de potencial e da derivada.	67
Tabela 2 - Componentes integrantes do agente robótico.....	103
Tabela 3 - Parametrização dos agentes adaptativos para os métodos base e integração reactiva na rotina 1.	111
Tabela 4 - Parametrização dos agentes adaptativos para os métodos base e integração reactiva na rotina 2.	113
Tabela 5 - Parametrização dos agentes adaptativos para os métodos base e hierárquico <i>quadtree</i> na rotina 1.	114
Tabela 6 - Parametrização dos agentes adaptativos para os métodos base e hierárquico <i>quadtree</i> na rotina 2.	116
Tabela 7 - Parametrização dos agentes adaptativos para os métodos base e com memória selectiva de instâncias da rotina 1.	117
Tabela 8- Parametrização dos agentes adaptativos para os métodos com memória selectiva de instâncias e <i>quadtree</i> da rotina 1.	119
Tabela 9 - Parametrização dos agentes adaptativos para os métodos com integração reactiva simples e integração reactiva com memória selectiva de instâncias da rotina 1.	121
Tabela 10 - Parametrização dos agentes adaptativos para os métodos com integração reactiva simples e integração reactiva com memória selectiva de instâncias da rotina 2.	123
Tabela 11 - Parametrização da rede neuronal para teste <i>offline</i>	128
Tabela 12 - Codificação das acções em sinalização.	156

Índice de Figuras

Figura 1 – Representação de um agente inteligente (adaptado de [1]).	14
Figura 2 – Os veículos de <i>Braitenberg</i> [25] são um dos exemplos de um agente reactivo simples.	15
Figura 3 – Arquitecturas com organização horizontal e vertical respectivamente (adaptado de [4]).	16
Figura 4 – Interação do agente com o ambiente na aprendizagem por reforço (adaptado de [5]).	18
Figura 5 – Regimes de aprendizagem baseado em iteração de política.	22
Figura 6 – Algoritmo do método <i>TD(0)</i>	23
Figura 7 - Sequência de aprendizagem.	24
Figura 8 - Algoritmo do método <i>SARSA</i>	25
Figura 9 - Algoritmo do método <i>Q-Learning</i>	26
Figura 10 – Algoritmo do método <i>Dyna-Q</i>	29
Figura 11 - Algoritmo do método <i>Q-Learning</i> Memória Episódica.	29
Figura 12 - Representação de um neurónio (retirado de [7]).	30
Figura 13 - Modelo base de um perceptrão [7].	31
Figura 14 - Algoritmo do perceptrão.	31
Figura 15 - Esquema abstracto da estrutura de uma rede <i>feed-forward</i> [7].	32
Figura 16 – Representação da descida de gradiente.	33
Figura 17 - Representação do labirinto dividido pela grelha de tarefas [10].	40
Figura 18 – Algoritmo do método <i>HAQL</i>	42
Figura 19 - Algoritmo do método CB-HARL.	43
Figura 20 - Algoritmo do método <i>Deep-Q</i> com repetição de experiências usado para aprendizagem de videojogos [11].	44
Figura 21 - Visões de racionalidade (adaptado de [12]).	49
Figura 22 - Arquitectura horizontal de camadas do agente híbrido simples.	60

Figura 23 - Algoritmo <i>HSL</i>	62
Figura 24 – Arquitectura híbrida com camada reactiva com memória.	64
Figura 25 - Dinâmica da máquina de estados comutadora de acção.....	66
Figura 26 - Algoritmo geral de aprendizagem por reforço recorrendo a abstracção de estado.	71
Figura 27 – Imagem original e a mesma comprimida através da <i>Quadtree</i> [16].	72
Figura 28 - Exemplo de uma representação em árvore [16].....	72
Figura 29 - Algoritmo do mecanismo de memoria selectiva de instâncias apresentado sob a forma de um diagrama de actividade.	75
Figura 30 - Algoritmo de recuperação de memória.	76
Figura 31 - Esquema exemplificativo de sensores contidos num agente; Quadrados representam sensores de distância e círculos os sensores de potencial.	77
Figura 32 – Esquema de vista superior dos componentes do robô. É possível notar os sensores de ultra-sons a azul, os sensores luminosos a laranja, motores a amarelo, unidades de processamento a verde e por fim as rodas a cinzento.....	79
Figura 33 – Ilustração simplista do ambiente em contexto de operação real.	80
Figura 34 - Diagrama de comunicação do agente prospector.....	81
Figura 35 - Diagrama de Actividade Executar do Agente Prospector.	82
Figura 36 -Diagrama de sequência do processar do controlo adaptativo.....	84
Figura 37 - Diagrama de actividade aprender <i>Q-Learning</i>	85
Figura 38 - Execução do agente <i>Q-Learning</i> no ambiente 3 com detalhe 1..	86
Figura 39 - Diagrama de actividade aprender <i>Dyna-Q</i>	87
Figura 40 - Vista da acção/valor/direcção do agente Dyna-Q (n=100) para o ambiente 3 e 6 respectivamente.	88
Figura 41 - Diagrama de actividade aprender Memória Episódica.....	89
Figura 42 - Ambiente nº3 visualizado sob a forma do campo de potencial gerado. O único obstáculo presente caracteriza um óptimo local.....	90

Figura 43 - Diagrama conceptual da tradução do paradigma reactivo para o paradigma orientado a objectos.	91
Figura 44 - Diagrama de actividade processar do controlo <i>HSL</i>	92
Figura 45 - Diagrama de actividade do método processar do controlo híbrido.	93
Figura 46 - Diagrama de actividade do método processar do controlo intermédio, ou reactivo com memória.	94
Figura 47 - Esquema da estrutura da rede neuronal.	95
Figura 48 - Visualizadores da plataforma PSA com a políticas original aprendida (visualizador superior) e a treinada (visualizador inferior).....	96
Figura 49 - Diagrama de actividade aprender do agente <i>Deep-Q</i>	97
Figura 50 - Diagrama de actividade do processo de criação da estrutura em árvore <i>Quadtrees</i>	98
Figura 51 - Diagrama de actividade da recuperação de informação da estrutura em árvore <i>Quadtrees</i>	99
Figura 52 - Visualização da memória de aprendizagem utilizando a estrutura <i>Quadtrees</i> . A política de acção é visível através das setas brancas. Ambiente 3 com grau 2 de detalhe.	99
Figura 53 - Agente adaptativo com memória selectiva de instâncias em simulação no ambiente 3 com detalhe 1.....	100
Figura 54 - Agente adaptativo com memória selectiva de instâncias em simulação no ambiente 3 com detalhe 2.....	100
Figura 55 - Visualizador de estado semelhante. Nos quadrados verdes é possível observar quais os estados posição correspondem ao mesmo estado sensorial.	102
Figura 56 - Diagrama de actividade <i>executar</i> do agente robótico.	104
Figura 57 - Esquema reactivo do módulo comportamental do agente robótico.	105
Figura 58 - Ambiente 3 da PSA.	110
Figura 59 - Ambiente 3b da PSA.	110
Figura 60 - Evolução dos métodos base em relação à integração de reactividade em simulação do ambiente 3 com grau de detalhe 1.	112

Figura 61 – Contabilização do total de passos dos métodos base em relação à integração de reactividade em simulação do ambiente 3 com grau de detalhe 1.....	112
Figura 62 - Evolução dos métodos base em relação à integração de reactividade em simulação do ambiente 3 com grau de detalhe 2.....	113
Figura 63 - Contabilização do total de passos dos métodos base em relação à integração de reactividade em simulação do ambiente 3 com grau de detalhe 2.....	114
Figura 64 - Evolução dos algoritmos base e com memória <i>Quadtree</i> no ambiente 3 com grau de detalhe 1.....	115
Figura 65 - Contabilização do total de passos de cada algoritmo na simulação do ambiente 3 com grau de detalhe 1.	115
Figura 66 - Evolução dos algoritmos base e com memória <i>Quadtree</i> no ambiente 3 com grau de detalhe 2.....	116
Figura 67 - Contabilização do total de passos dos algoritmos base e com memória <i>quadtree</i> no ambiente 3 com grau de detalhe 2.	117
Figura 68 - Evolução dos algoritmos base e com memória selectiva de instâncias no ambiente 3 com grau de detalhe 1.	118
Figura 69 - Contabilização do total de passos dos algoritmos base e com memória selectiva de instâncias no ambiente 3 com grau de detalhe 1.....	118
Figura 70- Evolução dos algoritmos com memória selectiva de instâncias e <i>quadtree</i> no ambiente 3 com grau de detalhe 1.....	119
Figura 71 - Contabilização do total de passos dos algoritmos com memória selectiva de instâncias e <i>quadtree</i> no ambiente 3 com grau de detalhe 1..	120
Figura 72 - Evolução dos algoritmos dos algoritmos com integração reactiva simples e integração reactiva com memória selectiva de instâncias no ambiente 3 com grau de detalhe 2.....	121
Figura 73 - Contabilização do total de passos dos algoritmos dos algoritmos com integração reactiva simples e integração reactiva com memória selectiva de instâncias no ambiente 3 com grau de detalhe 2.....	122
Figura 74 - Evolução dos algoritmos dos algoritmos com integração reactiva simples e integração reactiva com memória selectiva de instâncias no ambiente 3 com grau de detalhe 3.....	123

Figura 75 - Contabilização do total de passos dos algoritmos dos algoritmos com integração reactiva simples e integração reactiva com memória selectiva de instâncias no ambiente 3 com grau de detalhe 3.....	124
Figura 76 – Visualização do primeiro episódio, devido a seguir o potencial da camada reactiva o agente alcançou rapidamente o alvo.	125
Figura 77 - Função acção-valor gerada por uma simulação recorrendo à memória selectiva. A branco destaca-se uma memória de instância.	127
Figura 78 - Agente adaptativo HSL com heurística de potencial utilizando 8 acções. É visível a memória de aprendizagem desconexa.	129
Figura 79 - Gráfico da evolução dos algoritmos base com representação de estado diferente no ambiente 3 com grau de detalhe 1.	130
Figura 80 – Esquema do circuito MicroSwitch.....	151
Figura 81 - Esquema do divisor de tensão.....	153
Figura 82 - Esquema de circuito de um módulo HC-SR04.	153
Figura 83 - Esquema de circuito de sensor TSL2561.....	154
Figura 84 - Esquema do circuito do controlador e respectivos motores.....	156
Figura 85 - Peça 3D finalizada de suporte ao sensor HC-SR04.	157
Figura 86 - Suporte impresso para o sensor TSL6125.	158
Figura 87 - Agente robótico finalizado.	158
Figura 88 - Esquema de ligações completo.....	159
Figura 89 - Interface de visualização da PSA.....	162

1 Introdução

A criação de sistemas autónomos inteligentes tem ganho um maior relevo nos últimos anos, dado o crescimento e acessibilidade de recursos computacionais. Em particular, o ramo da aprendizagem por reforço representa um potencial enorme devido às aplicações em inúmeras áreas, como os videojogos ou a automatização e gestão de processos em tempo real, sendo um exemplo recente o surgimento de veículos autónomos comerciais.

A aprendizagem por reforço, como hoje se conhece, teve origem em duas vertentes distintas e independentes. Uma consistia no estudo da aprendizagem por tentativa e erro, com base na psicologia da aprendizagem animal, surgindo na década de 80 alguns dos primeiros trabalhos no âmbito da inteligência artificial.

A outra corrente focava-se em soluções de optimização através de funções de valor e programação dinâmica, não envolvendo, contudo, aprendizagem. No entanto, o aparecimento de um terceiro ramo envolvendo métodos de diferença temporal contribuiu largamente para a fusão de todos estes domínios e para motivar a génese da aprendizagem por reforço moderna.

Neste documento são abordados modelos e arquitecturas de aprendizagem por reforço, tendo como principal foco o seu estudo relativamente à operação com limitação de recursos. Adicionalmente, são apresentadas propostas de arquitecturas que visam abordar o problema da escassez de recursos na aprendizagem. A criação de uma plataforma física é igualmente realizada no sentido de ser possível testar diferentes configurações e métodos num contexto real.

1.1 Motivação

A existência de tecnologia e materiais que possibilitem soluções de realização de tarefas de forma autónoma não constitui uma novidade. Porém, a divulgação e crescente disponibilidade de componentes electrónicos a custo reduzido e acessível, possibilitou uma revolução na forma como o ser humano interage e cria tecnologia.

A entrada no mercado de diferentes dispositivos e máquinas robóticas permite percepcionar um futuro numa óptica tecnológica, onde os mais diversos tipos de tarefas, e posteriormente serviços, poderão ser executados

sem intervenção humana. Actualmente, um dos exemplos mais destacados, quando se recorre a meios robóticos, são os *drones*.

A aprendizagem por reforço surge assim na linha da frente com vista a fornecer soluções adaptativas para estes contextos. Contudo, numa perspectiva realista, a rapidez de resposta é imperativa, sendo necessário que a aprendizagem seja eficiente e expedita, em contraste com os escassos recursos normalmente existentes nestes módulos robóticos.

1.2 Objectivos

O presente trabalho tem como objectivos:

- O estudo e concretização de mecanismos adaptativos enquadrados em modelos e arquitecturas de agentes inteligentes.
- Exploração de diferentes técnicas com o propósito de avaliar de que forma a limitação de recursos influencia os resultados obtidos dos diferentes métodos de aprendizagem.
- Investigar os resultados obtidos no ponto anterior e encontrar possíveis soluções que permitam mitigar os efeitos adversos dessas limitações.
- Propor soluções para o problema da escassez de recursos sob a forma de arquitecturas de agente.
- Implementar uma Plataforma Robótica de Agente para realização de testes das soluções num contexto de operação real.

1.3 Organização do Documento

O presente documento encontra-se dividido nos seguintes capítulos.

Capítulo 1 – Introdução: Apresenta uma breve introdução ao propósito da realização do trabalho, objectivos propostos, as abordagens definidas e as tecnologias envolvidas.

Capítulo 2 – Enquadramento Teórico: Realiza uma síntese dos conceitos envolvidos com uma breve descrição sobre a inteligência artificial, o ramo da aprendizagem por reforço e o domínio da aprendizagem automática através redes neuronais.

Capítulo 3 – Métodos Alternativos de Aprendizagem por Reforço: Apresenta a descrição de algumas arquitecturas adaptativas propostas para solucionar problemas relacionados com a escassez de recursos.

Capítulo 4 – O Problema da Limitação de Recursos: Apresenta uma análise concisa do problema da limitação de recursos, definindo os seus contornos, explorando as ideias base e técnicas para lidar com o mesmo.

Capítulo 5 – Abordagens Propostas: Expõe os modelos propostos de arquitecturas de agente adaptativo e os aspectos base para o desenvolvimento dos mesmos, de forma a abordar o problema da aprendizagem com recursos limitados.

Capítulo 6 – Concretização: Descreve, de modo específico, todas as fases de implementação consideradas no trabalho até ao objectivo final, fundamentando as opções tomadas, problemas, limitações e resultados intrínsecos a cada arquitectura concretizada, com o respectivo enquadramento teórico.

Capítulo 7 – Resultados Experimentais: São apresentados, detalhados e relatados os testes e validações efectuados às implementações realizadas.

Capítulo 8 – Conclusão: Apresenta as conclusões consequentes do estudo e concretizações realizadas do trabalho, expondo ainda possíveis vertentes ou ideias a seguir em trabalho futuro.

2 Enquadramento Teórico

Neste capítulo é feita uma síntese às temáticas abordadas nesta dissertação, com realce para os conceitos principais do que representa a inteligência artificial e para o enquadramento teórico dos diferentes modelos e arquitecturas nas vertentes concretizadas.

2.1 Inteligência Artificial

A Inteligência Artificial consiste na tentativa de construir e compreender entidades inteligentes [1]. Na sua base, existe a tentativa de procurar estudar e entender o fenómeno da inteligência de uma perspectiva computacional.

Os estudos em Inteligência Artificial actualmente dividem-se em quatro ramos essenciais. Um ramo detém áreas relacionadas com a capacidade de aprendizagem e reconhecimento de padrões. Um outro ramo encontra-se ligado à biologia na tentativa de construir vida artificial. Um terceiro relacionado com a robótica, que interligado com a biologia, espera desenvolver sistemas que alojem vida artificial, mas também, interfaces de interacção com o ambiente. Por último, existe o designado ramo clássico da Inteligência Artificial cuja fundação remonta à Psicologia.

Todos os avanços e métodos alcançados visam a atingir o mesmo propósito: identificar, compreender e modelar as diferentes componentes do comportamento e atribuir capacidades, derivadas da inteligência, como a cognição e a racionalidade, a sistemas artificiais.

2.2 Agentes Inteligentes

No âmbito da inteligência artificial um agente inteligente consiste numa entidade computacional autónoma, com propósitos definidos e capacidades atribuídas no sentido de concretizar objectivos. De acordo com a literatura [1], um agente necessita deter pelo menos três de quatro qualidades: autonomia, reactividade, pró-actividade e eventualmente sociabilidade.

A autonomia num agente é definida como a capacidade de este realizar a sua operação de forma independente num ambiente, ou seja, o agente por si

próprio deve decidir como processar a informação obtida do exterior para que consiga atingir o propósito para o qual foi projectado.

Através da reactividade, pressupõe-se que um agente é capaz de detectar indicadores de mudança no ambiente e dar uma resposta adequada em tempo útil. Pela pró-actividade, um agente deverá não só reagir a estímulos imediatos, mas também actuar de forma orientada aos seus objectivos, tomando a iniciativa sempre que se justifique.

Por último, a capacidade de sociabilidade, quando existente, permite ao agente interagir com outros agentes de forma a concretizar objectivos que estão dependentes de outros ou para concretizar os seus mais rapidamente.

Neste sentido, as arquitecturas de agentes foram desenvolvidas com o intuito de integrar alguma inteligência nos agentes, atribuindo-lhes capacidade de raciocinar, aprender, conhecer, entre outras. Destes atributos emergem características como a cognição e a racionalidade. Quando adequados à finalidade pretendida, todos estes conceitos podem ser modelados num ciclo de percepção-processamento-acção de acordo com o paradigma e modelo considerado.

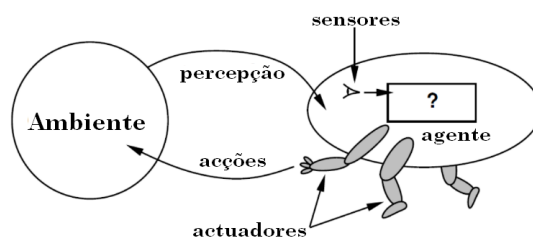


Figura 1 – Representação de um agente inteligente (adaptado de [1]).

2.2.1 Arquitecturas Reactivas

Um denominado agente reactivo interage com o ambiente baseando-se em regras estímulo-resposta. Um sistema autónomo reactivo constitui o mais simples dos agentes inteligentes em termos de processamento, sendo por essa razão dotado de uma maior capacidade de reacção quando comparado a agentes com outro tipo de arquitecturas, sendo por isso adequado para operação em tempo real.

Dado que o modelo reactivo assenta no paradigma comportamental, os objectivos deste tipo de agente encontram-se implícitos nas regras estímulo-resposta definidas. Um modelo interno do mundo é dispensável, uma vez que,

segundo esta abordagem toda a informação necessária para um agente interagir encontra-se no próprio ambiente externo e no acoplamento entre as componentes nucleares da percepção e acção.

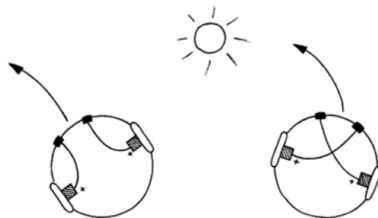


Figura 2 – Os veículos de *Braitenberg* [25] são um dos exemplos de um agente reactivo simples.

Em termos de arquitecturas reactivas existem duas propostas que se destacam: a arquitectura de subsunção [2] e a arquitectura de esquemas comportamentais [3]. Cada uma destas arquitecturas adiciona abordagens muito interessantes na perspectiva de definir comportamentos.

A arquitectura de Subsunção propõe a decomposição de comportamentos mais complexos através da modularização de comportamentos mais simples como uma forma de abstracção. Neste sentido, um comportamento complexo pode ser composto por um conjunto de comportamentos que, na sua forma nuclear poderão ser simples reacções (regras estímulo-resposta). Através do uso de um modelo de supressão, comportamentos considerados mais relevantes podem inibir outros comportamentos.

A arquitectura de Esquemas comportamentais difere da abordagem anterior uma vez que o seu foco se baseia na percepção de campos de potencial. Ao contrário da necessidade de supressão de comportamentos presente na arquitectura de Subsunção, as respostas aos estímulos são retractadas através de vectores e somadas, resultando numa resposta combinada.

O simples uso de campos de potencial para orientar e guiar um agente num ambiente origina um problema denominado de óptimos locais. Dado o tipo de topologia criada por campos de potencial, podem surgir situações onde o agente não consegue atingir a configuração com o maior potencial permitido, uma vez que, não tendo memória, ao avaliar o seu diferencial de potencial o agente não conseguirá capacitar-se que o máximo onde se encontra não corresponde ao máximo global. Este problema pode, porém, ser abordado ou evitado através de alguns métodos baseados em heurísticas ou arquitecturas com memória, respectivamente.

2.2.2 Arquitecturas Híbridas

As limitações intrínsecas a diferentes tipos de arquitectura de agente originaram uma procura por novas soluções. As arquitecturas híbridas surgiram no sentido de colmatar as desvantagens e problemas associados às diversas arquitecturas, tentando combinar as melhores características de cada tipo. No entanto, o factor combinatório da junção das diferentes arquitecturas acarreta um nível de complexidade adicional.

Para lidar com esta complexidade, as arquitecturas são abstraídas e são definidos níveis de competência organizados em camadas funcionalmente especializadas. Numa perspectiva organizacional as camadas intervenientes necessitam de ser integradas e de dar uma resposta coerente em termos comportamentais. Assim, no paradigma híbrido existem dois principais tipos de organização possível: horizontal e vertical.

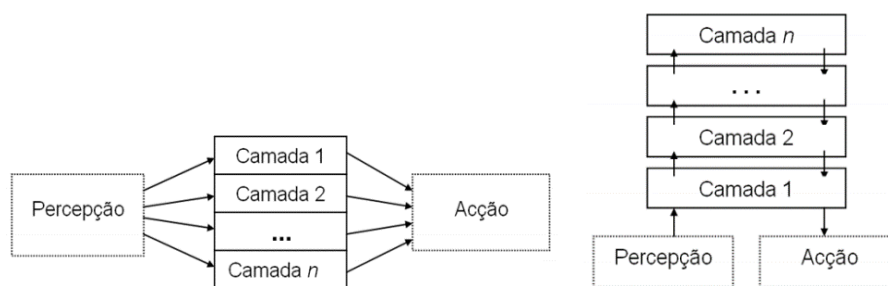


Figura 3 – Arquitecturas com organização horizontal e vertical respectivamente (adaptado de [4]).

A disposição das camadas horizontalmente possibilita a interacção directa destas com o ambiente, enquanto numa organização vertical existe uma propagação da informação percebida de uma camada inferior para uma camada superior. No entanto, ambos os tipos de organização apresentam vantagens e desvantagens. A possibilidade de as camadas interagirem directamente com o ambiente num contexto horizontal provoca uma multiplicidade de interacções, o que por sua vez resulta na geração de uma multiplicidade de respostas. Devido aos mecanismos internos de cada camada, uma resposta potencialmente distinta irá ser produzida em cada nível, existindo a necessidade de analisar e tratar possíveis faltas de coerência.

Para abordar o problema da tomada de decisão é necessário a existência de um coordenador que realize a escolha da resposta a actuar mediante os seus próprios critérios de selecção de acção. De forma paralela, esta solução

origina um ponto de falha localizado no próprio coordenador, visto que em caso de falha poderá comprometer o comportamento de um agente. O coordenador terá de obter conhecimento, relativo a cada camada, em função do seu tipo de critério, contribuindo para um maior número de dependências, sendo imperativo uma nova formulação do coordenador sempre que é adicionada uma nova camada.

Na organização vertical, o número de interações é reduzido pois cada camada tratará da informação de acordo com o seu nível de competência, delegando-a para a camada superior caso não tenha meios de processar tal informação. Nesta óptica, o ponto de falha espalha-se pelas camadas integrantes pois uma falha em qualquer camada poderá comprometer as camadas superiores e inibir uma resposta para as camadas inferiores.

2.3 Aprendizagem por reforço

Nesta secção são apresentados os principais conceitos de como é possível interagir com o ambiente e aprender através da experiência. São também introduzidos e explicados os principais algoritmos propostos, nomeadamente: *SARSA*, *Q-Learning* e *Dyna-Q*. Além da base teórica, são identificadas as limitações inerentes a estes mecanismos e sugeridos métodos para as ultrapassar.

2.3.1 Aprender através de recompensas

A aprendizagem por reforço pode ser definida num contexto simples como aprender o que fazer, a actuar. Em termos formais é possível descrever a aprendizagem por reforço como sendo uma aprendizagem comportamental em função dos efeitos das acções realizadas. A aprendizagem ocorre através da percepção e interacção com um ambiente, sobre o qual não se possui qualquer conhecimento *a priori*, de forma a maximizar um sinal numérico de reforço [5]. Essencialmente, estes métodos lidam com o encontrar de soluções, eventualmente óptimas, para problemas de controlo através de medições *online*. Esta abordagem é uma forma directa de aprender, recorrendo a programação dinâmica baseada numa aprendizagem através de uma função valor.

Estabelecendo um mapeamento entre os estados percebidos e acções é possível descobrir, através de tentativa e erro, que acção fornece a

melhor recompensa numa determinada situação. Em casos mais sofisticados, estas acções podem influenciar não só a recompensa imediata, mas também situações futuras e recompensas subsequentes. Esta característica, conhecida como recompensa diferida, permite a um determinado agente considerar não só retornos imediatos (locais), mas também retornos a longo prazo (globais), permitindo estes últimos orientar um individuo em função de uma motivação global relativa ao estado do ambiente.

A definição destas duas características, tentativa e erro e recompensa diferida, permite identificar com uma melhor resolução o que representa a aprendizagem por reforço.

A aprendizagem por reforço distingue-se da aprendizagem supervisionada principalmente pelo processo autónomo de aprendizagem e pelo factor de continuidade subjacente. Considerando um determinado ambiente, quando um agente permanece num dado estado s , seleccionará e realizará uma acção a , transitará eventualmente para um novo estado s' , obterá o reforço r associado a esse estado e voltará a repetir a sequência com uma nova acção a' até um estado objectivo ($s \rightarrow a \rightarrow r \rightarrow s' \rightarrow a'$). Com a experiencia obtida através da obtenção sequencial de recompensas o agente procederá com a sua aprendizagem incremental.

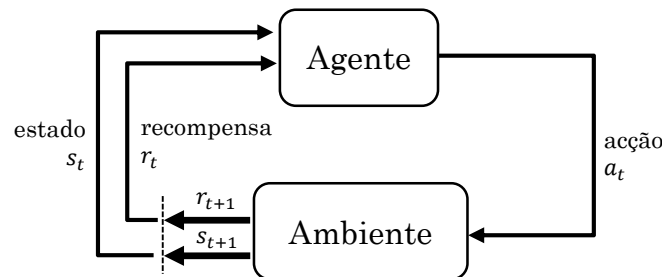


Figura 4 – Interação do agente com o ambiente na aprendizagem por reforço (adaptado de [5]).

Deste processo de aprendizagem surge o conceito de relacionamento entre estados e acções o qual se designa de política comportamental, que pode ser expressa da seguinte forma:

$$\pi: S \rightarrow A(s); s \in S, a \in A$$

Onde S constitui o espaço de estados possíveis e $A(s)$ o conjunto de acções possíveis para um determinado estado s . Em termos concretos, uma política de selecção de acção permite definir para um estado qual a acção a ser tomada.

A existência de uma política comportamental origina um problema substancial neste tipo de aprendizagem. Se, por um lado, o agente pretende obter uma recompensa máxima, deverá seleccionar as acções que permitam chegar a um estado objectivo. Contudo, dado que o agente aprende por si, este necessita, de igual forma, de perceber o ambiente em seu redor e experimentar acções que o levem a estados anteriormente desconhecidos. Estes novos estados representam por si um potencial inexplorado e podem significar até a obtenção de um reforço maior para o agente. Portanto é indispensável considerar o compromisso entre o aproveitamento (*exploitation*) e a exploração (*exploration*). Para garantir o equilíbrio neste compromisso, o uso de diferentes políticas de selecção de acção, permite calibrar o agente para a forma de proceder desejada. Políticas dinâmicas são também usadas, onde em função da quantidade de exploração realizada, o agente pode começar a aproveitar mais em detrimento da exploração (ou reciprocamente). A política de selecção de acção escolhida pode assim influenciar o comportamento de um agente.

O conceito de recompensa diferida e a propagação deste sinal constitui um aspecto chave nesta área. Quando um agente realiza uma acção e um reforço é obtido não existe garantia que o reforço seja significativo. A obtenção de um reforço considerável pode apenas ocorrer aquando do alcance de uma configuração de estado objectivo, pelo que dificuldades são acrescidas ao processo de aprendizagem. Neste contexto, o agente necessita de renunciar a recompensas imediatas e ponderar um retorno futuro através da acumulação de pequenas recompensas que possibilite maximizar uma recompensa significativamente maior a longo prazo.

Para definir o retorno a longo prazo, três modelos são propostos: horizonte finito, horizonte infinito e horizonte infinito com desconto temporal. A noção de política, estabelecida anteriormente, em conjunto com estes modelos de acumulação de reforço, fornece ao agente os mecanismos para regulação do compromisso entre exploração e aproveitamento.

O primeiro modelo, conhecido como horizonte finito, é bastante directo. Considerando um intervalo de domínio temporal finito, espera-se que o agente optimize a sua recompensa dentro desse âmbito.

$$R_t = \sum_{i=0}^n r_{t+i}$$

Este modelo é particularmente útil em instâncias que requeiram limites temporais de actividade, com n correspondendo ao total das interacções temporais definidas.

O modelo de horizonte infinito aposta na maximização da média dos valores de recompensa. Ao contrário das outras abordagens, a convergência para um limite onde as n interacções tendem para o infinito permite obter uma percepção da acumulação máxima que o agente poderá obter.

$$R_t = \lim_{n \rightarrow \infty} \left(\frac{1}{n} \sum_{i=0}^n r_{t+i} \right)$$

A principal fraqueza deste modelo assenta na própria média realizada: não existe diferenciação ou avaliação da evolução da recompensa ao longo do tempo. Da perspectiva de um agente, para um mesmo valor obtido de recompensa máxima, este poderia ter realizado um percurso inconstante com grandes perdas e grandes ganhos ou um percurso onde os grandes ganhos apenas foram obtidos perto do fim. Esta situação não é desejável uma vez que não permite avaliar com precisão o desempenho de políticas distintas (uma vez que os resultados finais obtidos seriam iguais).

Abordando o modelo de horizonte infinito com desconto temporal, este assume uma dedução temporal de forma a permitir definir um limite finito sobre um somatório potencialmente infinito.

$$R_t = \sum_{i=1}^{\infty} \gamma^{i-1} \cdot r_{t+i}$$

Este aspecto garante a não exigência de uma actividade finita por parte do agente. Este denominado “esquecimento”, devido ao parâmetro γ (onde $0 \leq \gamma < 1$), permite assim ajustar o quanto se pretende considerar relativamente a futuras recompensas em contraste com o presente instante.

Para o agente incorporar o conhecimento obtido sob a forma de recompensa, através da percepção de estados e realização de acções, é necessário associar uma valorização aos estados ou aos pares estado-acção. Sendo importante para o processo de aprendizagem classificar as acções tomadas num dado estado, uma função valor é considerada.

Contrariamente à função de recompensa, uma função valor indica a melhor orientação a longo prazo, i.e., o valor associado a um estado representa o montante total que é espectável alcançar futuramente a partir desse estado.

Com esta informação, o agente consegue ter uma predição de quão compensadores (ou não) podem ser os estados seguintes.

Com os conceitos de estado, acção, política comportamental, função de recompensa e função de valor estão definidos os principais subelementos da aprendizagem por reforço. A estes subelementos é possível acrescentar ainda o conceito de uma representação do ambiente exterior ao agente, o qual é designado por modelo interno. Quando um agente contempla no seu processo um modelo interno, preenche ou actualiza a sua representação e usa a informação obtida para obter mais conhecimento através de simulações internas. O acto de recordar é assim expresso metaforicamente neste processo. No entanto, métodos baseados em modelos internos são computacionalmente mais dispendiosos e por consequente a sua utilização deve ser apenas equacionada se o custo computacional for menor que os requisitos reais.

Enunciados os subelementos relativos à aprendizagem por reforço, é possível considerar que qualquer problema possa ser representado de forma atómica num mapeamento entre estados e acções. Este tipo de representação é admissível, no entanto, a representação de estados mais complexos deve ser ponderada visto que, por sua vez resultaria numa explosão combinatória impossível de abordar.

Dentro deste tipo de aprendizagem baseada em função valor, é também possível distinguir duas importantes classes de métodos de aprendizagem atentando à sua própria esquemática no processo de aprender.

Um primeiro esquema é designado por iteração pela política (*Policy-Iteration based – “Actor-critic” learning*), onde a avaliação da política calcula a função valor através da política actual e assumindo a mesma como uma política estacionária. A avaliação de política expressa a forma como calcular a função valor para uma política arbitrária, expondo o *problema da predição*.

A estrutura da política é designada de actor, sendo este usado para seleccionar as acções, e a função valor estimada é conhecida como a crítica, pois critica as acções feitas pelo actor. Neste esquema, a aprendizagem é sempre *on-policy*: a crítica conhece e critica qualquer política seguida pelo actor.

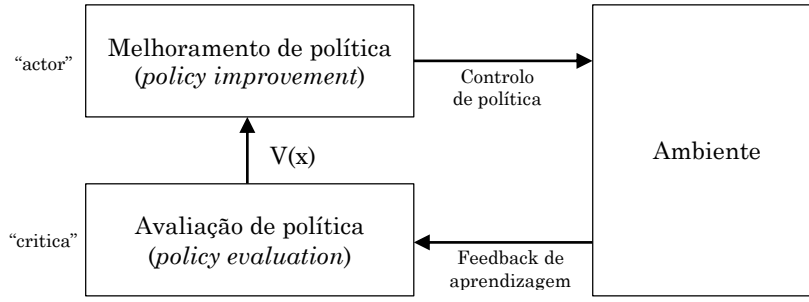


Figura 5 – Regimes de aprendizagem baseado em iteração de política.

Nos métodos que comportam esta avaliação de política encontram-se: *Monte Carlo*, *diferença temporal* – $TD(\lambda)$, etc. O aperfeiçoamento da política é baseado na iteração da política em adição com algum eventual processo de exploração. O segundo tipo de regime é baseado na iteração por valor (*value-based iteration*), que corresponde a uma versão *online* das recursões. O algoritmo *Q-learning* apresenta-se como a base deste tipo de classe.

2.3.2 Aprendizagem por diferença temporal

A aprendizagem por diferença temporal (*TD*) aprende a prever o valor de uma dada variável ao longo de vários intervalos de tempo. Esta tarefa é designada por problema da predição. Esta forma de aprendizagem é uma abordagem que permite assim prever uma quantidade que depende dos valores futuros de um determinado sinal.

Na sua forma original, a diferença temporal é um processo de aprendizagem, onde o sinal de treino para uma predição será uma predição futura. A aprendizagem por diferença temporal produz em cada passo temporal discreto t , uma estimativa ou previsão, p_t , do valor seguinte:

$$Y_t = y_{t+1} + \gamma y_{t+2} + \gamma^2 y_{t+3} + \dots = \sum_{i=1}^{\infty} \gamma^{i-1} y_{t+i}$$

Onde γ é um factor de desconto, com $0 \leq \gamma < 1$. Cada estimativa é uma previsão pois envolve valores futuros de y . O factor de desconto restringe o grau de influência dos valores futuros de y em relação às previsões actuais.

A diferença temporal é frequentemente aplicada na aprendizagem por reforço com o intuito de prever uma medida da quantidade total de recompensa esperada no futuro. Neste sentido, a aprendizagem por diferença

temporal é uma técnica não supervisionada, onde o agente aprende a prever o valor esperado a ocorrer no final de uma sequência de estados.

O algoritmo mais simples – $TD(0)$

Para entender melhor este algoritmo é possível atentar ao caso mais simples da previsão por diferença temporal: a predição de apenas uma iteração futura ($\gamma = 0$). Por sua vez, esta forma no contexto de aprendizagem é conhecida por $TD(0)$.

Executar apenas uma previsão de um passo define que $p_t = y_{t+1}$ para cada iteração t . A aprendizagem pode ser utilizada para corrigir o erro iterativamente e actualizar a função de predição à medida que novos valores são obtidos. Esta operação é realizada através do cálculo do erro entre a previsão actual, p_t , e a previsão alvo y_{t+1} . Num contexto onde a predição é completa, ou seja, onde $\gamma \neq 0$, seria necessário calcular toda a predição, i.e., $Y_t = y_{t+1} + \gamma y_{t+2} + \gamma^2 y_{t+3} + \dots$

Quando aplicada esta ideia ao cálculo da função valor num contexto de aprendizagem:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

Onde $r_{t+1} + \gamma V(s_{t+1})$ corresponde à estimativa de retorno e $V(s_{t+1}) - V(s_t)$ à diferença temporal. O algoritmo para calcular a política de um agente que recorra a esta técnica é apresentada na Figura 6.

Iniciar $V(s)$ arbitrariamente.
Repetir (para cada episódio):
 Iniciar s .
 Repetir (para cada passo):
 $a \leftarrow$ acção dada por $\pi(s)$
 Executar a , observar r, s' .
 $V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$
 $s \leftarrow s'$
 Até s ser terminal.

Figura 6 – Algoritmo do método $TD(0)$.

$TD(\lambda)$

O algoritmo $TD(\lambda)$ ou TD-Lambda foi proposto por *Richard Sutton* com base na investigação sobre a aprendizagem por diferença temporal. Esta abordagem abrange o problema da predição considerando todas as futuras diferenças temporais, todos os passos T , com uma ponderação baseada no decaimento de memória, λ .

$$\hat{V}(s_n) := \hat{V}(s_n) + \alpha \left(\sum_{m=0}^{\infty} \lambda^m d_{n+m} \right), \quad \text{com } 0 \leq \lambda \leq 1 \text{ e } d_n \equiv 0 \text{ para } n \geq T$$

O parâmetro λ consiste num configurador do decaimento, onde um valor elevado permite a manutenção de um rastro maior e uma maior influência de uma recompensa ao longo de mais estados. Se $\lambda=1$, é obtido a configuração Monte Carlo. As propriedades de convergência desta solução são semelhantes ao $TD(0)$, contudo, através da escolha apropriada de λ , é possível obter resultados mais rápidos em determinadas situações. Assim, esta prática considera resultados de acções mais afastadas no tempo, permitindo acelerar a aprendizagem.

2.3.3 Método de aprendizagem SARSA

O método de aprendizagem por reforço *SARSA* consiste num método cuja origem do nome deriva do seu processamento interno, i.e., advém das sequências episódicas de um agente. Estando num estado s , realiza a acção a , obtém um reforço r , progride para um novo estado s' e volta a realizar uma acção a' , durante o processo de aprendizagem. O algoritmo *SARSA* compreende uma forma de predição assente na noção de aprendizagem por diferença temporal (*TD - Temporal Difference Learning*).

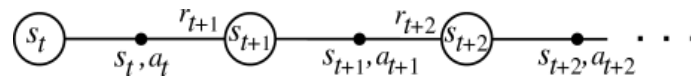


Figura 7 - Sequência de aprendizagem.

A utilização deste método comporta duas características importantes: um controlo de aprendizagem do tipo *on-policy* e uma função valor de estado-acção designada de $Q(s,a)$. O controlo de aprendizagem *on-policy* expressa que, a política de um agente para o processo de aprendizagem é a mesma que

a usada para agir. Nesse sentido para explorar todas as acções, políticas como ϵ -greedy ou ϵ -soft são sugeridas.

No contexto da função de valor, o agente aprende através da actualização de uma estimativa do valor de estado-acção com base na mudança (diferença temporal) desse valor entre instantes sucessivos. A formulação da função $Q(s,a)$ é semelhante à função valor de estado ($V(s)$) do algoritmo $TD(0)$, uma vez que apenas a função de valor é alterada:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$$

De notar ainda que esta abordagem consiste numa aprendizagem sem o agente deter uma representação interna do ambiente. Esta solução resulta na necessidade de armazenar apenas os valores da função $Q(s,a)$ e por consequência menos dispendiosa em termos de recursos em relação a outras soluções com modelos internos mais complexos.

```

Iniciar  $Q(s,a)$  arbitrariamente.
Repetir (para cada episódio):
  Iniciar  $s$ .
  Escolher  $a$  para  $s$  utilizando uma política derivada de  $Q$  (ex:  $\epsilon$ -greedy)
  Repetir (para cada passo):
    Executar  $a$ , observar  $r, s'$ 
    Escolher  $a'$  para  $s'$  utilizando uma política derivada de  $Q$  (ex:  $\epsilon$ -greedy)
     $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$ 
     $s \leftarrow s'; a \leftarrow a'$ 
  Até  $s$  ser terminal.

```

Figura 8 - Algoritmo do método SARSA.

2.3.4 Método de aprendizagem Q-Learning

A solução *Q-Learning* proposta por *Watkins* [6] resulta do conhecimento prévio de outras soluções e sugere um método sem recurso a modelo interno para alcançar uma política óptima. De forma análoga ao SARSA, este agente utiliza uma mesma função de valor $Q(s,a)$ e expressão de actualização, existindo apenas diferenciação no controlo de aprendizagem.

Um controlo *off-policy* é definido para o agente para que existam duas políticas de selecção de acção: uma para o agente agir e outra usada para a aprendizagem. No contexto da aprendizagem, através do controlo *off-policy*, o

agente actualiza sempre a sua função $Q(s,a)$ com base numa política comportamental óptima:

$$\Pi^*(s) = \operatorname{argmax}_a Q^*(s,a)$$

O uso da política óptima permite que um comportamento óptimo seja sempre escolhido no processo de actualização da função $Q(s,a)$. Tal significa que para um determinado estado s , a acção a^* será designada por acção óptima se for a acção que maximiza a função $Q(s,a)$ para o respectivo estado. Com esta alteração a expressão da actualização da função $Q(s,a)$ é a seguinte:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Independentemente da acção realizada pelo agente no ambiente, no processo de actualização é sempre escolhida a acção óptima que corresponda à obtenção da função $Q(s,a)$ óptima, levando a que um comportamento óptimo seja obtido a longo prazo.

```

Iniciar  $Q(s,a)$  arbitrariamente.
Repetir (para cada episódio):
  Iniciar  $s$ .
  Repetir (para cada passo):
    Escolher  $a$  para  $s$  utilizando uma política derivada de  $Q$  (ex:  $\epsilon$ -greedy)
    Executar  $a$ , observar  $r, s'$ 
     $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$ 
     $s \leftarrow s'$ 
  Até  $s$  ser terminal.
  
```

Figura 9 - Algoritmo do método Q -Learning.

2.3.5 Método de aprendizagem Dyna-Q

Nos métodos de aprendizagem por reforço sem modelo interno, o sinal de reforço é propagado pela actualização da função de valor quando o agente transita entre estados. Quando a propagação alcançada é suficiente para um agente gerar uma política óptima ou relativamente satisfatória, o tempo de convergência é geralmente muito elevado comparativamente ao desejável, no sentido do agente ser eficiente.

No Q -Learning, uma solução óptima pode ser obtida ao convergir iterativamente. Devido à função de valor apenas propagar a informação entre

o estado actual e o próximo estado para onde se transita, é necessário, para um agente aprender, realizar um número considerável de episódios (e de iterações de actualização da função de valor) para um valor de reforço se propagar extensivamente. Esta situação advém do reforço obtido ser significativo apenas em iterações prévias ao alcance do objectivo.

A arquitectura *Dyna-Q*, proposta por *Sutton*, introduz um modelo interno de forma a contornar o problema da propagação de valor e tempo de convergência. O modelo interno é criado e actualizado através das percepções do ambiente em cada iteração. Aproveitando a informação contida no modelo, o agente produz experiências adicionais, com base em simulações, possibilitando uma actualização da função de valor e por consequência uma convergência mais rápida dos valores. Este aproveitamento da experiência real obtida, resulta numa aprendizagem bastante mais veloz comparativamente ao método *Q-Learning*.

Alterações no mundo, resultantes do ambiente externo ser dinâmico, podem, no entanto, invalidar a rápida aprendizagem efectuada. Num contexto de ambiente estocástico ao invés de determinístico não é possível assumir que o agente, dado um estado e uma acção, transitará sempre para o mesmo estado. Numa transição de estado, dada uma acção, o agente transita para um estado com uma determinada probabilidade. Esta questão pertinente levanta dificuldades visto que, numa simulação interna o agente não possui certeza sobre para qual estado irá transitar. As iterações internas poderão resultar numa aprendizagem errada não representativa do ambiente uma vez que, o modelo interno que o agente possui encontra-se desactualizado ou é actualizado a um ritmo muito inferior em comparação com o dinamismo exterior.

A representação interna do mundo construída pelo algoritmo *Dyna-Q* abrange os elementos essenciais para modelar e reproduzir as ocorrências reais. Para possibilitar a operação real do agente assim como as simulações internas são definidos três conceitos: uma função de transição, uma função de recompensa e um conjunto de pares estado-acção.

$$Modelo(s, a) = \begin{cases} S \\ A \\ \hat{T}(s, a, s') \\ \hat{R}(s, a) \end{cases}$$

Para qualquer modelo, determinista ou não, o modelo interno define a tradução dos eventos que acontecem no ambiente na forma da função de transição de estados T . Para um estado s e uma acção a é associado um novo estado s' (se determinista) ou uma probabilidade p (se não determinista).

Paralelamente, associado à função de transição, a função de recompensa R representa a informação do reforço associado a um estado s e acção a .

Estas funções tornam assim o conhecimento do agente mais robusto permitindo executar as simulações internas. Além do modelo, um parâmetro n irá especificar o número de simulações a realizar entre passos reais. Estas duas características diferenciam assim o método *Dyna-Q* do método *Q-Learning* devido à capacidade de realizar uma propagação dos sinais de reforço da função $Q(s,a)$ de forma mais rápida em função das n iterações definidas. No caso de n ser nulo não existem simulações internas a serem executadas pelo que a aprendizagem do agente assume a forma do método *Q-learning*.

Em situação determinística, a função de transição e recompensa mapeiam para um par estado-acção um novo estado s' e um reforço r respectivamente. Em termos não determinísticos é necessário estimar o modelo interno:

$$\hat{T}(s, a, s') = \frac{\text{Número de experiências } (s, a, s')}{\text{Número de experiências } (s, a)}$$

$$\hat{R}(s, a) = \frac{\text{Soma recompensas obtidas por executar acção } a \text{ no estado } s}{\text{Número de experiências } (s, a)}$$

Não existindo uma certeza associada ao reforço, é necessário estimar o valor com base na média dos valores de reforço obtidos sempre que um estado s realiza a mesma acção a . Na transição é necessário calcular, dado um par estado-acção, a probabilidade p de alcançar um novo estado s' . O cálculo dessa probabilidade é realizado através divisão do *número de passagens pelo estado (s,a) com transição para o estado s'* pelo *número total de passagens pelo par (s,a)* (independente do novo estado).

Relativamente à diferenciação do algoritmo, o método *Dyna-Q* resulta numa extensão do algoritmo *Q-Learning* adicionando duas componentes: actualização do modelo interno e simulações.

```

Iniciar  $Q(s,a)$  e  $\text{Modelo}(s,a)$  para  $s \in S, a \in A(s)$ .
Repetir para sempre:
     $s \leftarrow$  estado corrente (não terminal).
     $a \leftarrow \varepsilon\text{-greedy}(s,Q)$ 
    Executar  $a$ , observar  $r, s'$ 
     $Q(s,a) \leftarrow Q(s,a) + a[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$ 
     $\text{Modelo}(s,a) \leftarrow s', r$  (assumindo ambiente determinista)
    Repetir N passos:
         $s \leftarrow$  estado aleatório observado previamente.
         $a \leftarrow$  acção aleatória executada previamente em  $s$ .
         $s', r \leftarrow \text{Modelo}(s,a)$ 
         $Q(s,a) \leftarrow Q(s,a) + a[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$ 

```

Figura 10 – Algoritmo do método *Dyna-Q*.

2.3.6 Método de aprendizagem Q-Learning com Memória Episódica

Incorporando as características da aprendizagem referida anteriormente, a memória episódica é utilizada no sentido de conseguir executar passos de simulação sem a necessidade de integrar um modelo interno. Desta forma as experiências ocorridas são registadas numa memória e posteriormente seleccionadas de forma aleatória em passos de aprendizagem simulados.

```

Iniciar  $Q(s,a)$  arbitrariamente e memória episódica ME.
Repetir para sempre:
     $s \leftarrow$  estado corrente (não terminal).
     $a \leftarrow \varepsilon\text{-greedy}(s,Q)$ 
    Executar  $a$ , observar  $r, s'$ 
     $Q(s,a) \leftarrow Q(s,a) + a[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$ 
     $ME \leftarrow (s, a, s', r)$ 
    Repetir N passos:
         $(s, a, s', r) \leftarrow ME$  (episódio aleatório experienciado previamente.)
         $Q(s,a) \leftarrow Q(s,a) + a[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$ 

```

Figura 11 - Algoritmo do método *Q-Learning* Memória Episódica.

Contudo, este método contém um problema ao nível da memória episódica. Caso não haja limite, os registos em memória permanecem em crescimento, podendo exigir mais recursos relativamente a memória necessária.

2.4 *Redes Neurais Artificiais*

O funcionamento e incrível capacidade de processamento do cérebro humano constitui um dos maiores mistérios para a humanidade. Com o decorrer dos avanços científicos e dos processos de investigação, é hoje possível descrever um cérebro como uma complexa rede biológica de ligações através de células designadas de neurónios. O processo, descrito simplifadamente, consiste em elaboradas transmissões de sinais eléctricos entre neurónios que resultam em inúmeros diferentes padrões, traduzindo-se posteriormente em imensos tipos de resposta neurológica.

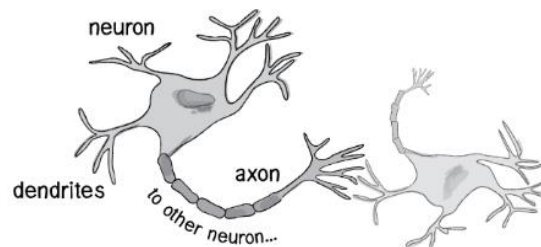


Figura 12 - Representação de um neurónio (retirado de [7]).

A tentativa de reproduzir a actividade neuronal, resultante do comportamento deste tipo de células, não tardou a surgir na área da computação [8]. O conceito de neurónio é então descrito como sendo uma célula que pode receber entradas, realizar o seu processamento e gerar uma saída. As várias propostas e aplicações evoluíram mais tarde para o modelo computacional, designado como rede neuronal artificial.

Uma rede neuronal enquadra o conceito dos neurónios, num sistema computacional de paradigma conexionista. Neste tipo de paradigma a informação é processada e propagada de forma paralela através dos nós (neurónios) integrantes de uma rede. Ao contrário da abordagem procedimental, tipicamente presente nos sistemas codificados através de uma linguagem de programação, o fluxo de informação não segue de forma linear.

Contrariamente ao que se pensava previamente, uma rede neuronal encontra-se bastante longe de simular o funcionamento real de um cérebro. Contudo, este modelo é extremamente útil, por exemplo, no reconhecimento de padrões, um tipo de problema relativamente fácil para o cérebro humano, mas muito árduo num contexto computacional.

A capacidade de uma rede neuronal aprender e realizar ajustes na sua estrutura interna permite que sejam amplamente utilizadas actualmente no ramo da inteligência artificial. As suas características adaptativas são

empregues em: reconhecimento de padrões, processamento de sinal, detecções de anomalias, entre outras aplicações.

Com o desenvolvimento dos conceitos subjacentes a este modelo computacional, surgiu o primeiro modelo de uma rede neuronal nomeado Perceptrão [9]. Neste modelo simples de rede neuronal são admitidas uma ou mais entradas, determinados pesos associados às ligações, o seu processamento e por fim uma saída, tal como o modelo neuronal descrito anteriormente. O perceptrão é também o tipo mais básico de uma rede do tipo “*feed-forward*”, onde a informação a ser processada, flui sempre no sentido da entrada para a saída sem existir realimentação.

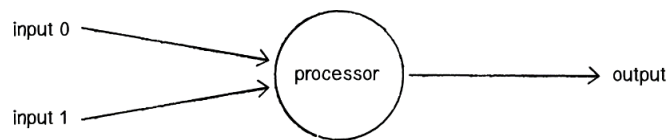


Figura 13 - Modelo base de um perceptrão [7].

- | |
|--|
| <ol style="list-style-type: none">1. Para cada entrada, multiplicar essa entrada pelos pesos associados.2. Realizar o somatório de todas as entradas pesadas.3. Calcular a saída através da passagem do resultado do somatório pela função de activação. |
|--|

Figura 14 - Algoritmo do perceptrão.

Este tipo de rede simples tornou-se a base embrionária para os diferentes tipos de redes neurais que têm sido propostas e existem presentemente.

Todavia, alterações nos pesos e pendoros (*biases*) de perceptrões de uma rede produzem alterações bruscas na saída, provocando eventualmente resultados incorrectos. Este problema é ultrapassável introduzindo um novo tipo de neurónio, com uma função de activação diferente: a Sigmóide. A alteração desta característica permite um comportamento semelhante aos perceptrões, mas reflectindo uma saída alterada e atenuada aquando da modificação dos pesos e pendoros dos nós.

2.4.1 Redes *Feed-Forward*

Com a investigação do modelo do perceptrão, este tipo de redes evoluiu para as redes neuronais multicamada. Estas redes podem conter mais do que um nó e mais do que uma camada, tendo o modelo mais simples normalmente três camadas: a de entrada, uma escondida e a de saída. Sendo que cada uma destas pode conter o número desejável de nós.

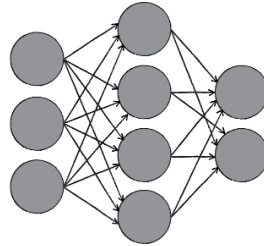


Figura 15 - Esquema abstracto da estrutura de uma rede *feed-forward* [7].

A adição de um número superior de camadas escondidas varia de acordo com os requisitos e propósitos de utilização da rede, sendo que por norma, mais camadas escondidas possibilitam operações de discriminação adicionais no processamento do espaço de características.

2.4.2 Aprendizagem em redes neuronais artificiais

No início de uma rede, os seus pesos são por norma definidos de forma aleatória pois não são conhecidos os valores correctos para o propósito da rede. Nesse sentido, surge o conceito de treino da rede onde é necessário embutir conhecimento específico da aprendizagem pretendida.

Colocar uma rede a aprender, consiste em fornecer exemplos com as características de entrada para posteriormente comparar a sua saída com uma referência fornecida, sendo a avaliação realizada através do cálculo do erro. Não sendo praticável a exploração manual de todas as configurações, o conceito base compreende a minimização de uma função de custo (ou erro) através da propagação contrária do sinal de erro.

A designação desta função prende-se com o objectivo de medir o custo de prever a saída y' quando a resposta real é y . A sua origem prende-se com o facto de as funções de activação serem diferenciáveis, permitindo a minimização do erro através da actualização dos pesos. Sendo difícil obter

uma solução óptima, este processo tenta descobrir uma solução satisfatória, que minimize o erro da resposta da rede.

$$J(w) = \frac{1}{2} \sum_i (y_i - y'_i)^2$$

$y'_i \in \mathbb{R}$

A equação apresentada define a função de custo $J(w)$ que expressa a soma dos erros quadráticos que pode ser usada como uma função de custo. Os termos y e y' correspondem à referência de treino e à predição, respectivamente. No entanto, diversas funções de custo podem ser consideradas.

A conjugação do espaço de características em conjunto com a função de custo pode também influenciar directamente o desempenho. A superfície do erro obtida no hiperespaço altera-se conforme a função considerada, com diferentes funções a corresponderem a hiper-planos diferentes. Esta ocorrência pode facilitar, ou não, o encontrar de um erro mínimo satisfatório.

A própria procura por mínimos constitui um desafio devido à exploração do espaço definido pela função. Consoante o problema, esta exploração pode revelar-se dispendiosa, quando em contraste com o tempo necessário até encontrar uma solução. Adicionalmente, uma técnica não guiada de aprendizagem pode não encontrar sequer solução.

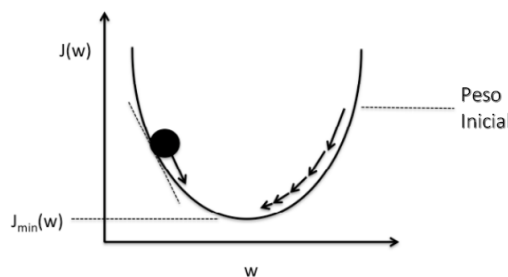


Figura 16 – Representação da descida de gradiente.

Atendendo a esta problemática, a técnica da descida de gradiente permite obter informação do gradiente e avançar no sentido da minimização do valor da função. Assim, é possível seguir na direcção de mínimos e evitar explorar todo o espaço da função, evoluindo o erro iterativamente a um passo pré-definido.

$$\Delta \mathbf{w} = -\eta \nabla J(\mathbf{w})$$

Assim, temos que calcular a derivada parcial da função de custo para cada peso no vector de peso.

$$\Delta w_j = -\eta \partial J / \partial w_j$$

Para o exemplo da função de custo considerada, a derivada parcial para um peso poderá ser calculada da seguinte forma:

$$\begin{aligned} \frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i (t^{(i)} - o^{(i)})^2 \\ &= \frac{1}{2} \sum_i \frac{\partial}{\partial w_j} (t^{(i)} - o^{(i)})^2 \\ &= \frac{1}{2} \sum_i 2(t^{(i)} - o^{(i)}) \frac{\partial}{\partial w_j} (t^{(i)} - o^{(i)}) \\ &= \sum_i (t^{(i)} - o^{(i)}) \frac{\partial}{\partial w_j} \left(t^{(i)} - \sum_j w_j x_j^{(i)} \right) \\ &= \sum_i (t^{(i)} - o^{(i)}) (-x_j^{(i)}) \end{aligned}$$

(t = target, o = output)

Assim, temos que:

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = -\eta \sum_i (t^{(i)} - o^{(i)}) (-x_j^{(i)}) = \eta \sum_i (t^{(i)} - o^{(i)}) x_j^{(i)},$$

Podendo de seguida realizar a actualização do peso:

$$\mathbf{w} := \mathbf{w} + \Delta \mathbf{w}$$

Contudo, esta técnica não está isenta de problemas, nomeadamente quando se atenta ao passo de aprendizagem. Se esta taxa de aprendizagem for demasiado grande, a descida de gradiente irá ultrapassar o mínimo e divergir. Se a taxa de aprendizagem for muito pequena, o algoritmo vai exigir muitas iterações para convergir e poderá ficar facilmente preso em mínimos locais.

Realizar o escalamento das características consideradas contribui também para melhores resultados. Quando os valores se encontram na mesma escala, é mais fácil encontrar a taxa de aprendizagem adequada, o que contribui para atingir a convergência de forma mais rápida.

Sendo uma forma de aprendizagem supervisionada, conjuntos de treino e de teste são considerados quando da realização do treino. Este factor implica, por sua vez, que um problema de sobre-adaptação (*overfitting*) também poderá ocorrer. Esta situação acontece quando uma rede treinada se adapta em demasia ao conjunto de treino, não conseguindo produzir bons resultados no de teste.

2.4.3 Técnicas de optimização estocástica

Tentar o treino de uma rede, num contexto onde o conjunto de treino tem uma grande dimensionalidade, implica um cálculo pesado em termos computacionais. Esta limitação implica directamente um esforço maior e mais tempo despendido na aprendizagem por ser necessário determinar o gradiente com todos os pesos para o conjunto de treino.

2.4.3.1 Descida estocástica de gradiente

Recorrendo a uma perspectiva de simplificação drástica, é possível em cada iteração estimar o gradiente através da escolha aleatória de um exemplo. Esta técnica, designada de descida estocástica do gradiente, pode ser utilizada de forma expedita em tempo real pois não necessita de recordar que exemplos foram percorridos nas iterações prévias.

Mesmo sendo uma aproximação da direcção (gradiente) esta abordagem tem certas vantagens. Sendo uma estimação, permite a convergência do algoritmo mais rapidamente devido à actualização dos pesos ser realizada após cada exemplo. Este factor é também vantajoso para grandes conjuntos de dados (*datasets*) ou em contexto de operação real onde novos dados, recentemente adquiridos, podem ser imediatamente usados no treino.

Outra motivação prede-se com a função de custo ser tipicamente não convexa. Esta característica, usando dados diferentes em cada iteração, pode ajudar a ultrapassar mínimos locais. A actualização dos pesos [10] é realizada de acordo com a regra seguinte:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

Onde θ corresponde aos pesos, η o parâmetro de passo e $\nabla L(\theta_t)$ a função que calcula o gradiente. Para um contexto de larga escala, é também comum utilizar o conceito de separar o conjunto de treino em pequenos conjuntos (*mini-batches*). Esta solução possibilita obter um compromisso entre a tradicional descida e as vantagens estocásticas e suavizar a convergência. No processo de descida, é frequente o gradiente mudar rapidamente em cada iteração, pois o custo encontra-se a ser calculado através de dados diferentes. Esta situação é muitas vezes parcialmente mitigada pela reutilização do valor do gradiente da iteração anterior, escalado por um hiper parâmetro de momento [10]:

$$\begin{aligned}v_{t+1} &= \mu v_t - \eta \nabla L(\theta_t) \\ \theta_{t+1} &= \theta_t + v_{t+1}\end{aligned}$$

Onde v corresponde ao hiper parâmetro de momento, sendo usado na restante actualização dos pesos. Este suplemento permite eventualmente melhorar o processo de minimização do erro, fornecendo alguma informação de segunda ordem sobre o gradiente.

2.4.3.2 Método RMSProp

Sendo a optimização o foco destas técnicas, diversos métodos baseados na descida estocástica são propostos com o objectivo de melhorar e acelerar o processo de minimização. O *RMSProp* faz uso da magnitude dos gradientes recentes para normalizar o gradiente actual. Uma média actualizada do valor eficaz (*root mean squared*) é mantida para normalizar o gradiente corrente. A média do valor eficaz é calculada segundo a forma [10]:

$$g_{t+1} = \gamma g_t + (1 - \gamma) \nabla L(\theta_t)^2$$

A actualização dos pesos é realizada subtraindo o gradiente corrente normalizado:

$$\theta_{t+1} = \theta_t - \frac{\eta \nabla L(\theta_t)}{\sqrt{g_{t+1}} + \epsilon \sqrt{g_{t+1}}}$$

O *RMSProp* tem várias vantagens de utilização. Não só sendo igualmente compatível com pequenos lotes de aprendizagem (*mini-batches*), como também se revela um optimizador robusto devido às informações de pseudo curvatura (uma estimativa de um segundo momento).

Conjuntamente, uma versão com adição de um momento de primeira ordem também pode ser considerada, tal como de seguida apresentado.

$$\mathbf{m}_{t+1} = \gamma \mathbf{m}_t + (1 - \gamma) \nabla L(\theta_t)$$

$$\mathbf{g}_{t+1} = \gamma \mathbf{g}_t + (1 - \gamma) \nabla L(\theta_t)^2$$

$$\mathbf{v}_{t+1} = \mu \mathbf{v}_t - \frac{\eta \nabla L(\theta_t)}{\sqrt{\mathbf{g}_{t+1} - \mathbf{m}_{t+1}^2 + \epsilon}}$$

$$\theta_{t+1} = \theta_t + \mathbf{v}_{t+1}$$

3 Métodos alternativos de Aprendizagem por Reforço

Neste capítulo são apresentadas algumas soluções a problemas reais que se enquadram no âmbito da aprendizagem por reforço. São analisadas as características e vantagens destes tipos de aprendizagem e como foram aplicados nos diversos casos.

3.1 Aprendizagem por Reforço Hierárquica

A essência da exploração consiste na tentativa de atenuar a incerteza. Neste sentido, a exploração, na aprendizagem por reforço, refere-se à estratégia usada por um agente para adquirir novas informações sobre o ambiente, sendo que esta capacidade pode influenciar directamente a aprendizagem.

Uma forma de afectar positivamente o processo de aprendizagem por reforço, reside em permitir a aprendizagem simultânea em várias resoluções no espaço e tempo. O paradigma abstracto pode ser assim evocado através do conceito de delegação de tarefas a diferentes níveis de actuação e aquisição de conhecimento.

3.1.1 *Feudal Reinforcement Learning*

Seguindo a linha de pensamento da aprendizagem hierárquica, a aprendizagem por reforço feudal, ou *Feudal Reinforcement Learning*, propõe uma solução para o problema de escalabilidade onde, o tempo de aprendizagem não escala bem com o número de parâmetros. Para lidar com esta problemática é definida a noção de gestor, existindo uma hierarquia de domínio onde gestores detêm poder absoluto sobre os seus subgestores. Neste seguimento, um gestor pode definir tarefas, recompensar ou punir os seus subgestores adequadamente.

Esta aprendizagem a diferentes níveis de abstracção assenta assim sobre dois aspectos chave. Primeiramente, os gestores de nível superior devem recompensar os subgestores pelas suas acções, quer tenham satisfeito

ou não os comandos. Secundariamente, os gestores apenas necessitam de deter conhecimento do estado do sistema na granularidade das próprias escolhas de tarefas.

Para ilustrar esta proposta, um desafio padrão sob a forma de labirinto foi proposto. Uma grelha é colocada sobre o ambiente do labirinto com o objectivo de o repartir sucessivamente em áreas com menor grão. Os gestores são atribuídos de seguida a cada parte e em diferentes níveis. O estabelecimento desta grelha define assim uma tarefa conjunta e os diferentes níveis da hierarquia.

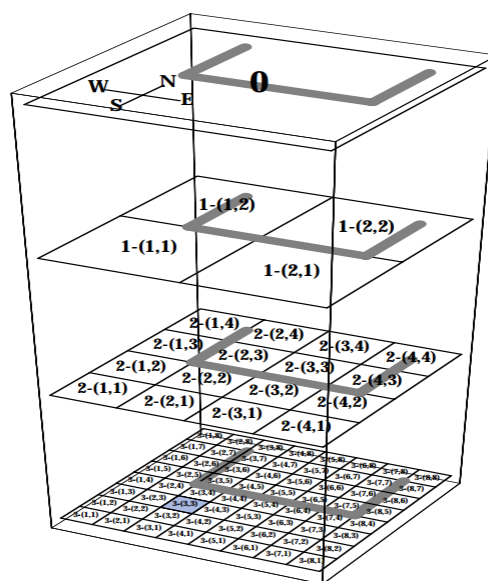


Figura 17 - Representação do labirinto dividido pela grelha de tarefas [10].

Esta proposta depende, no entanto, da existência de um sistema plausível de gerir, e de preferência, baseado numa divisão hierárquica natural do espaço de estados disponível. Para algumas tarefas, pode ser muito ineficiente, pois força cada subgestor a aprender a satisfazer as subtarefas definidas pelo gestor, independentemente das mesmas serem ou não adequadas. Assim, este método tem maior utilidade em ambientes nos quais as tarefas definidas podem ser alteradas.

3.2 Aprendizagem acelerada por heurística

No contexto deste método de aprendizagem, uma heurística consiste numa estimação adequada do custo ou distância desde um estado até um objectivo. Nesse sentido, uma função heurística faz uma subestimação da distância, com a heurística até ao objectivo a ser menor ou igual à distância real.

Fazendo uso do conceito de heurística, é possível aplicar uma função deste tipo a um problema de aprendizagem por reforço, para acelerar a convergência dos algoritmos. Este uso é vantajoso, pois pode ser utilizado para escolher convenientemente as acções e guiar a exploração no processo de aprendizagem.

3.2.1 Heuristically Accelerated Q-Learning (HAQL)

No seguimento das ideias referidas anteriormente, diversas técnicas têm sido propostas. Uma destas propostas adiciona capacidades heurísticas ao algoritmo base *Q-Learning*, onde uma função heurística é definida e actualizada iterativamente. Em cada iteração a função heurística é útil na geração da política comportamental dada pela equação [11]:

$$\pi(s) = \begin{cases} \arg \max_a [F(s, a) \bowtie \xi H(s, a)^\beta] & \text{se } q \leq p, \\ a_{\text{random}} & \text{caso contrário,} \end{cases}$$

Onde \bowtie define o operador a usar. Esta política quando aplicada ao caso mais simples, o HAQL, assume a forma:

$$\pi(s) = \begin{cases} \arg \max_a [\hat{Q}(s, a) + \xi H(s, a)] & \text{se } q \leq p, \\ a_{\text{random}} & \text{caso contrário,} \end{cases}$$

Por sua vez, a heurística calculada baseia-se na equação seguinte:

$$H(s, a) = \begin{cases} \max_i \hat{Q}(s, i) - \hat{Q}(s, a) + \eta & \text{se } a = \pi^H(s), \\ 0 & \text{caso contrário,} \end{cases},$$

A política formulada é a única diferença, fazendo uso de uma ponderação através da função heurística, onde ξ é um parâmetro que visa controlar a influência heurística.

```

Iniciar  $\hat{Q}_t(s, a)$  e  $H_t(s, a)$  arbitrariamente.
Repetir (para cada episódio):
  Iniciar  $s$ .
  Repetir (para cada passo):
    Actualizar os valores de  $H_t(s, a)$  como desejado.
    Escolher acção  $a$  através de  $\pi(s)$ .
    Executar  $a$ , observar  $r, s'$ 
     $Q(s, a) \leftarrow Q(s, a) + a[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'$ ;
  Até  $s$  ser terminal.

```

Figura 18 – Algoritmo do método *HAQL*.

3.3 Aprendizagem baseada em casos (*Case Based R.L.*)

O raciocínio baseado em casos (*Case Based Reasoning*) faz uso do conhecimento adquirido em situações experienciadas anteriormente (casos) de forma a extrapolar a aprendizagem para novos problemas emergentes.

Com base nesta vertente, é possível considerar uma nova abordagem com o objectivo de utilizar o conhecimento baseado em casos como uma heurística. Esta forma permite acelerar algoritmos de aprendizagem por reforço e combinar assim estas duas áreas. Esta solução poderá revelar-se bastante vantajosa pois por norma a convergência de um algoritmo tende apenas a ser alcançada aquando de um reconhecimento intensivo do espaço de pares estado-acção, cuja exploração consome tempo excessivo.

3.3.1 Case Based Heuristically Accelerated Q-Learning (CB-HAQL)

Combinando as áreas de aprendizagem por reforço e raciocínio baseado em casos, foi proposto o algoritmo *CB-HARL*. Esta extensão ao algoritmo *HAQL* sugere a recuperação de casos, compatíveis com a situação de operação corrente, de forma a construir uma função heurística correspondente ao caso.

Nesta proposta, a cada iteração é calculada a medida de semelhança de cada caso com a situação actual e determinado um custo de adaptação. Na eventualidade de um caso ser recuperado com sucesso, onde a semelhança se encontra acima de um limiar e o custo de adaptação é baixo, uma heurística é determinada seguindo a equação [11]:

$$H(s, a) = \begin{cases} \max_i \hat{Q}(s, i) - \hat{Q}(s, a) + \eta & \text{se } a = \pi^H(s), \\ 0 & \text{Caso Contrário,} \end{cases}$$

Toda a função heurística é calculada e usada durante um período de tempo, fazendo uso do conjunto de acções recuperadas do caso escolhido. Posteriormente a selecção de acção é realizada de forma análoga à solução HAQL.

```

Iniciar  $\hat{Q}_t(s, a)$  e  $H_t(s, a)$  arbitrariamente.
Repetir (para cada episódio):
  Iniciar  $s$ .
  Repetir (para cada passo):
    Calcular Similaridade e Custo.
    Se existir um caso que possa ser reutilizado:
      Recuperar caso e adaptar se necessário.
      Calcular  $H_t(s, a)$  com as acções sugeridas pelo caso seleccionado.
    Escolher acção  $a$  através de  $\pi(s)$ .
    Executar  $a$ , observar  $r, s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'$ 

```

Figura 19 - Algoritmo do método CB-HARL.

3.4 Aprendizagem Multinível

As redes neuronais apresentam um elevado potencial no sentido de lidar com a multiplicidade de estado e processamento presente na componente de aprendizagem por reforço. Uma vez que estes modelos computacionais são extremamente eficientes na discriminação e reconhecimento de padrões é plausível a sua utilização como uma aproximação da função acção-valor que seria eventualmente gerada.

3.4.1 Arquitectura *Deep-Q*

A concepção de uma arquitectura de agente que no seu núcleo substitua a função acção-valor por uma função de aproximação na forma de uma rede neuronal permite um contexto de operação sob recursos limitados.

A este nível a arquitectura *Deep-Q* [11] conseguiu alcançar resultados favoráveis ao aprender com sucesso um número considerável de jogos da consola *Atari 2600*. Neste ambiente de teste, uma câmara para captar as imagens do jogo serve de entrada no sistema e todos os comandos, presentes no controlo da consola, actuam como a saída, definindo as acções possíveis.

Recorrendo a uma rede neuronal convolucional para processamento visual, um estado é gerado pelas últimas quatro *frames* obtidas, que ao serem processadas pela rede geram um conjunto de blocos. Estes blocos são por sua vez utilizados como entrada numa rede neuronal do tipo *feed-forward* multicamada. Com a estrutura da rede definida o algoritmo de aprendizagem proposto é apresentado na Figura 20.

Iniciar a memória de repetição D com capacidade N .
Iniciar a função acção-valor Q com pesos aleatórios θ .
Iniciar a função acção-valor \hat{Q} com pesos $\theta^- = \theta$.
Repetir (para cada episódio):
 Iniciar sequência $s_1 = \{x_1\}$ e sequência pré processada $\varphi_1 = \varphi(s_1)$
 Repetir (para cada passo):
 Com probabilidade ε seleccionar uma acção aleatória a_t ,
 Caso contrário seleccionar $a_t = \operatorname{argmax}_a Q(\varphi(s_t), a; \theta)$.
 Executar a_t no emulador e obter r e imagem x_{t+1} .
 Definir $s_{t+1} = s_t, a_t, x_{t+1}$ e pré processar $\varphi_{t+1} = \varphi(s_{t+1})$
 Armazenar transição $(\varphi_t, a_t, r_t, \varphi_{t+1})$ em D .
 Amostrar um *minibatch* aleatório de transições de D .
 Definir $y_i = \begin{cases} r_j & \text{se episódio termina no passo } j + 1, \\ r_j + \gamma \max_{a'} \hat{Q}(\varphi_{j+1}, a'; \theta^-) & \text{caso contrário,} \end{cases}$
 Executar um passo de descida de gradiente em $(y_i - Q(\varphi_{j+1}, a'; \theta^-))^2$ referente aos parâmetros θ da rede.
 A cada C passos, reiniciar: $\hat{Q} = Q$
 Até ao passo terminal.
Até ao fim.

Figura 20 - Algoritmo do método *Deep-Q* com repetição de experiências usado para aprendizagem de videojogos [11].

O funcionamento deste algoritmo assume algumas semelhanças com os métodos de aprendizagem que recorrem a simulações internas. A memória de aprendizagem é ocupada por uma arquitectura de rede neuronal cujo treino é

iterativamente realizado recorrendo a técnicas estocásticas durante cada passo do agente. Importante também referir a existência de duas instâncias da mesma rede, onde uma instância da rede é treinada e substituída a cada número de passos uma segunda instância da rede utilizada na selecção de acção.

3.5 Conclusões

Neste capítulo apresentaram-se algumas das abordagens e propostas, no domínio da aprendizagem por reforço, que sugerem soluções na óptica de acelerar a convergência dos algoritmos base.

Foram introduzidos alguns conceitos e soluções que recorrem a métodos heurísticos ou aprendizagem baseada em casos, que ajudam a direccionar a exploração, e outros, como o *Deep-Q*, cuja proposta faz uso da aprendizagem multicamada e suas estruturas.

No capítulo subsequente, será abordado em detalhe o problema da escassez de recursos, justificando a motivação inerente para as propostas referidas anteriormente.

4 O Problema da Limitação de Recursos

Qualquer solução pode, a dado momento, debater-se com a carência de recursos necessários à resolução do problema. O desenvolvimento da área de inteligência artificial não difere dessa restrição, existindo sempre compromissos nos tipos de recursos essenciais: a memória, capacidade de processamento e tempo. Este problema pode ser abordado de duas perspectivas diferentes, primeiramente um foco interno às capacidades do agente e, alternativamente, estudando a relação e correspondência do ambiente com o agente inteligente.

A aprendizagem por reforço, sendo um grande ramo da área, sofre igualmente das mesmas limitações. O esforço necessário na aprendizagem encontra-se directamente relacionado com a complexidade intrínseca ao problema cuja solução se pretende. Por consequência, a eficiência e tempo despendido representam um factor qualitativo e uma medida de desempenho face a diferentes métodos de resolução.

Este capítulo pretende fornecer uma perspectiva global do problema da escassez de recursos e contextualizar as diferentes visões de racionalidade com o âmbito da dissertação.

4.1 Visões de Racionalidade

A faculdade de raciocinar implica fundamentar o processo de tomada de decisão com base no conhecimento detido. O conceito de racionalidade consiste assim, na capacidade de tomar a melhor decisão em função do conhecimento actual. A capacidade de inferir sobre o mundo está limitada pelo tempo, conhecimento e poder computacional. Em função destes factores, e do seu grau de limitação, é possível identificar diversas visões de racionalidade.

Sem patamar limitativo, uma visão supra-computacional emerge, onde um sistema capaz de tomar sempre a melhor decisão possível é conceptualizado, uma vez que este possui todo o tempo necessário, conhecimento infinito e poder computacional absoluto. Embora este domínio não seja viável em termos concretos, permite estabelecer um ponto de

comparação com a realidade. Esta ideia clarifica uma visão assente na verdade e no possível, uma racionalidade ecológica.

No entanto, a própria visão e significado de racionalidade sofreu evoluções profundas. *Gingerenzer* [12] expressa estas alterações destacando duas revoluções principais. Durante os dois milénios desde Aristóteles, o pensamento intelectual ocidental consistia em duas formas de conhecimento: prova demonstrativa e raciocínio provável.

Com a invenção do cálculo probabilístico, uma nova visão de raciocínio pragmático floresceu, desenvolvendo a ideia de uma perspectiva humilde onde existe um grau de incerteza sobre o conhecimento humano e do mundo. Esta revolução probabilística influenciou todas as vertentes científicas e a forma como olhamos para o funcionamento da mente. Segundo *Gingerenzer* [12], nesta perspectiva, estima-se que as funções mentais recorrem a computações ao nível da probabilidade e utilidade. Esta perspectiva designa o raciocínio, julgamento e tomada de decisão sobre as leis da probabilidade, imperando modelos mais complexos e cada vez mais realistas. Contudo, o problema reside precisamente na dimensão e rigor destes modelos, levando ao abandono de configurações simples e limitadas, estas últimas mais adaptadas à psicologia e pensamento humano.

A segunda e nova revolução, proposta por *Gingerenzer*, manifesta o desejo ambicioso de modelar o raciocínio e tomada de decisão humana através do uso de heurísticas expeditas e frugais. A motivação para tal, surge pela escassa necessidade de computação e ausência de cálculos probabilísticos, ou de utilidade, para realizar inferências com tempo e conhecimento limitado. Estas heurísticas são modelos de racionalidade limitada, focando-se nas normas de comportamento humano e atenuando uma abordagem demasiado probabilística.

A racionalidade assume diversas formas, existindo uma bifurcação entre o domínio realista e o supra-computacional. A Figura 21 apresenta as várias visões de racionalidade.

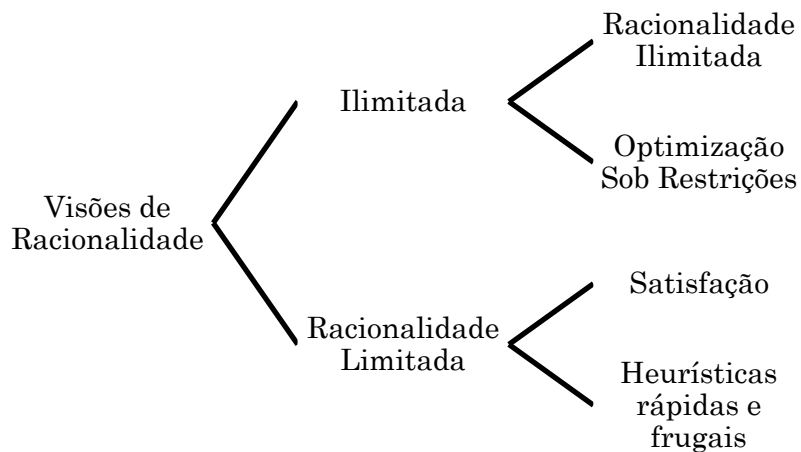


Figura 21 - Visões de racionalidade (adaptado de [12]).

Sobre a perspectiva ilimitada, encontra-se uma visão de racionalidade ilimitada, com realizações de maximização de utilidade e redes *bayesianas* a serem o destaque nesta vertente baseada na teoria das probabilidades. A otimização sob restrições também constitui um desafio pois, para alcançar uma otimização, é necessário procurar e obter toda a informação necessária para tal. A própria procura de informação compreende a principal diferença da racionalidade ilimitada em relação às outras visões. Não existe previsão de quando uma procura se revela suficiente ou completa, podendo nem ser possível quando realizada com restrições. Neste sentido, a otimização sob restrições assume que, uma condição de paragem otimiza a procura de informação relativamente ao tempo. Este modelo especifica que devem ser calculados os custos e benefícios, com a procura a ser feita até a condição de paragem ser atingida, ou seja, até os custos ultrapassarem os benefícios. Estes cálculos e processamento adicional podem exigir ainda mais conhecimento e poder que o teoricamente requerido na racionalidade ilimitada.

4.2 Racionalidade Limitada

A vertente oposta da racionalidade ilimitada apresenta uma visão onde a racionalidade é limitada. Esta visão expressa que um agente deve inferir sobre o mundo real num período de temporal realista, operando com a informação obtida e recursos computacionais que dispõe. Esta visão de racionalidade não pretende estudar ou alcançar a otimização de tarefas em ambientes.

Hebert Simon, considerado o precursor deste ramo, estabelece as limitações da mente e a estrutura do ambiente como sendo duas componentes essenciais e interligadas [13]. As limitações mentais implicam que o julgamento e tomada de decisão devem ser elaborados com base nas capacidades já constatadas da mente. Por sua vez, o ambiente dispõe a informação de acordo com a sua estrutura, sendo necessário o estudo do mesmo de forma a concluir a viabilidade e compatibilidade das componentes do agente inteligente.

Nesta perspectiva, a estrutura do ambiente, e a informação presente, são fundamentais para compreender a cognição e comportamento relativamente ao processo adaptativo, uma vez que o mesmo é referente ao próprio ambiente.

Racionalidade Ecológica

Paralelamente, é importante constatar uma outra perspectiva, não estando esta preocupada com a satisfação de critérios internos. A racionalidade ecológica é uma noção que destaca a importância da correspondência entre a estratégia e o ambiente, sendo um conceito importante para entender o *porquê* e *quando* a racionalidade limitada funciona [13].

A razoabilidade dos modelos de racionalidade limitada deriva da sua racionalidade ecológica, não da coerência ou consistência interna das escolhas. Uma estratégia é ecologicamente racional em função do seu grau de adaptação em relação à informação contida no ambiente, seja o mesmo físico ou social [13].

No sentido dos objectivos e noções de uma racionalidade limitada, duas propostas são evidenciadas: satisfação e heurísticas rápidas e frugais. As mesmas provando por si que não é necessário deter poder e recursos supra-computacionais para raciocinar.

4.2.1 Satisfação

Na maioria das situações no mundo real, estratégias óptimas são desconhecidas ou mesmo impossíveis de descobrir. Mesmo considerando um caso controlado, como um jogo de xadrez, damas ou outro jogo de tabuleiro, é difícil obter uma jogada óptima devido ao tempo limitado para calcular tal jogada, mesmo quando a melhor jogada possível existe efectivamente. Estes

cenários remetem para a clara dificuldade de obtenção de políticas de acção óptimas em situações menos bem definidas como acontecem num ambiente natural.

Sendo limitada, depressa a mente humana encontrou solução através de métodos que realizam uma aproximação para lidar com os desafios diários. Esta ideia de aproximação traduz-se directamente no conceito de satisfação. A satisfação consiste na metodologia para tomada de decisão sobre um conjunto de opções descobertas consecutivamente quando não existe conhecimento adicional de possibilidades futuras [12]. Estabelecendo um patamar de aspiração ajustável [14], a satisfação funciona concebendo um procedimento, onde a procura por uma opção termina quando o mesmo limiar é atingido.

4.2.2 Heurísticas rápidas e frugais

Uma heurística consiste em qualquer abordagem para resolução de um problema, aprendizagem ou descoberta pelo uso de um método prático, aproximado, não garantido solução óptima, mas suficiente para auxiliar no alcance de objectivos. As heurísticas abordam um problema, incompleto dado o conhecimento disponível, recorrendo a métodos que não são necessariamente precisos, ou não correspondem à realidade, mas que permanecem úteis no sentido de fornecer orientação nas direcções apropriadas.

Na biologia humana, os métodos heurísticos são capacidades da mente que permitem tirar vantagem da estrutura de informação presente no ambiente e alcançar decisões. A mente humana foca-se assim em formas e configurações onde, heurísticas simplistas podem conduzir a inferências precisas e úteis, com um mínimo de esforço de processamento. Neste sentido, estes métodos não são o resultado de um sistema mental incompleto ou com falhas. Para obter boas decisões num ambiente rodeado de incerteza, as heurísticas ganham uma condição de indispensabilidade e importância significativa, pois a sua capacidade para explorar e aproveitar as estruturas de informação no ambiente determinam o sucesso das mesmas num contexto limitado a nível racional.

O nosso desenvolvimento permitiu ao ser humano equipar-se com a capacidade de obter as características relevantes ao meio e ajustar as ferramentas de acordo com este. Este facto implica que as estratégias são seleccionadas de acordo com as condições do ambiente que nos são impostas.

Payne, Bettman e Johnson [15] propõem que os algoritmos escolhidos dependem da quantidade de tempo existente para fazer uma escolha, ou uma inferência, e o quanto tal decisão pode custar.

No domínio da inteligência artificial, as abordagens heurísticas têm sido restringidas no contexto de certos métodos poderem ser aplicados de modo geral a qualquer problema, ao invés de uma avaliação estudada e cuidada de considerar as heurísticas ajustadas a um conjunto e impróprias a outro.

Uma questão relevante emerge: como são seleccionadas as heurísticas? A razão mais importante é que, tarefas específicas exigem ferramentas específicas, ou seja, cada heurística é especializada em determinada classe de problemas, o que significa que a outra maioria não seja aplicável a uma dada situação. De acordo com *Gerd Gigerenzer* e o grupo de pesquisa *ABC* [12], é possível considerar duas vertentes sobrepostas a fim de determinar a escolha heurística: tarefas adaptativas específicas, por exemplo, escolha de um companheiro, e tarefas de inferência específicas, como categorização ou estimativas.

As heurísticas rápidas e frugais têm em conta que a informação pode ser difícil de encontrar em primeiro lugar. Além disso, estas heurísticas são muito mais fáceis de seguir em contexto de operação real e baseiam-se em três princípios simples:

- Regras simples de busca;
- Regras simples de paragem;
- Regras simples de decisão.

Desta forma as razões primárias e vantajosas em relação a estas heurísticas prendem-se com estas serem:

- Ecologicamente racionais, podendo ser implementadas com tempo, esforço mental, ou com outros recursos limitados;
- Em muitos casos adquirem um desempenho tão bom quanto os algoritmos mais sofisticados (regressões lineares e multi-lineares [12]);
- Fornecer informações sobre a inteligência e processos mentais aplicados na vida real.

Dos métodos heurísticos existentes, alguns demonstram-se muito eficazes dado a sua simplicidade. Um exemplo concreto reside na heurística de reconhecimento [12], esta heurística faz uso da capacidade de reconhecimento obtida por via da evolução da mente humana e permite a um determinado ser tomar decisões através do aproveitamento da sua própria

ignorância. Reconhecendo apenas uma parte das alternativas, existe uma hipótese de a decisão mais correcta ser efectivamente a opção reconhecida em detrimento das restantes. Este método além de rápido, actua com conhecimento limitado e possibilita uma precisão satisfatória, em certas problemáticas, quando comparado a outros métodos [12].

Outra heurística é a designada por "*tirar o melhor*" ("*Take the best*"), cuja tomada de decisão se baseia em considerar apenas uma boa razão e ignorar o resto. O critério para a escolha da razão útil consiste nas próprias capacidades discriminativas, conseguindo obter uma boa precisão, comparativamente a outros métodos não heurísticos, e com apenas uma parte do poder de processamento necessário. Traduzindo-se na sua alta frugalidade, i.e., quantas pistas, ou razões, são necessárias processar para chegar a uma decisão.

Em geral, heurísticas são fáceis de usar, exigem pouca memória e capacidade computacional, e por consequência parecem ser modelos muito plausíveis para explicar o comportamento humano (e animal), podendo ser da mesma forma uma grande vantagem para sistemas de inteligência artificial.

4.3 Considerações sobre Racionalidade Limitada

O conceito de racionalidade máxima pode ser definido como um processo de raciocínio onde, a melhor decisão pode ser tomada se existir total conhecimento sobre o domínio do problema. Na realidade, processar e adquirir toda a informação, referente a um determinado problema, consiste numa tarefa árdua ou mesmo impossível. Esta limitação de conhecimento obriga a encontrar soluções com os recursos disponíveis e adquiridos, não existindo sempre uma garantia na obtenção de uma solução óptima.

No domínio da inteligência artificial, um agente racional é aquele que maximiza a sua utilidade esperada, dado o seu conhecimento presente. Porém, os recursos do agente podem encontrar-se limitados pela sua própria qualidade. Esta incapacidade pode advir de dois factores integrantes do agente: recolha de informação incompleta do ambiente em seu redor e robustez ao realizar o processamento da mesma.

A recolha de informação torna-se incompleta no sentido em que a sua captação é realizada por sensores. Estes, por sua vez, não possuem uma gama de alcance infinito para o seu domínio e, também, podem perceber com erro ou ruído o mundo exterior. Existindo ainda um factor de discretização sobre o ambiente observado.

O processamento da informação percebida é também um desafio devido ao factor de escala intrínseco: maior quantidade de informação constitui uma maior carga ao nível do processamento e memória. Esta robustez incide directamente sobre o agente e a sua capacidade de resposta, onde este tentará fornecer uma solução mais racional possível para os seus recursos.

Nesta perspectiva, um sistema incapaz de alcançar uma solução óptima, com os recursos que detém, visa fornecer uma solução orientada num contexto de satisfação. Todavia, uma solução sub-óptima não constitui necessariamente uma má solução em termos qualitativos. Em termos de qualidade, uma solução sub-óptima pode ser avaliada em função da complexidade associada ao problema e da ponderação entre satisfação e optimização relativas ao(s) objectivo(s) do agente.

4.4 Técnicas a abordar

Encontrar métodos de resolução que se provem eficazes e satisfatórios exige, primeiramente, compreender a extensão e o detalhe das limitações inerentes. No domínio da aprendizagem através de recompensas, é possível identificar partes que poderão influenciar directamente o tempo de convergência e desempenho.

Para melhorar o tempo de resposta, e suavizar o impacto no processo de aprendizagem dos diversos componentes, é possível considerar um conjunto de conceitos, com aplicação em diferentes frentes, nomeadamente: simplificação, abstracção, focagem, aproximação e ainda modularização.

O contexto de simplificação representa a perspectiva mais elaborada, uma vez que, é possível tentar simplificar a forma de estado ou do próprio mecanismo de aprendizagem. A simplificação pode estar presente em diversos aspectos dos conceitos referidos.

O conceito de abstracção enuncia uma forma de lidar com a complexidade através da redução do detalhe. Esta redução implica obrigatoriamente algum tipo de perda de informação, sendo necessário atentar a problemas como o fenómeno de *aliasing*.

Técnicas de abstracção podem ser aplicadas no processo de aprendizagem pela forma como os estados são armazenados em memória. Contrariamente ao que acontece na simplificação de um estado, o estado é observado, mantendo o mesmo formato, mas com informação relevante a ser agregada e detalhe descartado.

Por sua vez, através da focagem, é realizada uma selecção da informação a processar de acordo com condições e critérios relevantes a um problema. Uma abordagem focada, pode ser aplicada em diferentes vertentes como, mecanismos limitativos de memória ou tempo de processamento. Um exemplo concreto poderia ser a escolha selectiva de experiências para simulação interna no mecanismo de aprendizagem.

A motivação do uso de técnicas de aproximação pode ser explicada com base no princípio de *Pareto* que, quando aplicado ao contexto de aprendizagem, expressa que, com 20% do esforço ou custo, seja viável alcançar 80% dos resultados. Na prática, métodos, como os heurísticos, permitem efectivamente realizar uma aproximação à resolução óptima de problema. Assim, é possível sacrificar a qualidade da solução em prole de uma menor complexidade computacional.

Por fim, a modularização é um conceito que visa a reduzir a complexidade através da identificação e divisão em partes distintas. Com esta forma de estruturação, é possível decompor sucessivamente uma problemática em módulos de menor sofisticação. Concretamente, é possível observar o seu uso e vantagens na própria elaboração da aprendizagem por reforço, onde os seus componentes e mecanismo são identificados e separados.

Adicionalmente a utilização de uma abordagem híbrida de forma a discriminar as acções de um agente em termos comportamentais e de acordo com a racionalidade existente no momento pode ser vantajoso. Por sua vez, os próprios conceitos referidos anteriormente podem ser adicionados na vertente adaptativa em conjunto com o sistema híbrido.

4.4.1 Representação de estado

Em termos concretos, alguns componentes, como a representação de estado considerada ou a multiplicidade dos pares estado-acção armazenados, podem prejudicar o tempo de processamento e conduzir a uma situação não escalável.

Situações deste tipo surgem quando a dimensão do espaço de estados evolui em conjunto com a exploração do ambiente, não existindo uma dimensionalidade fixa. Esta característica não é de todo desejada, devido à explosão combinatória e aumento de complexidade no processamento dos pares estado acção.

Uma representação de estado simplificado permite tornar a dimensão do estado finita ou pelo menos consideravelmente menor e, por consequente, tornar o processamento possível em tempo útil.

Uma representação posicional, ou seja, através do uso de coordenadas para referir de forma absoluta um local no espaço, apresenta problemas de escalabilidade relevantes. Evoluindo para uma representação de estado relativa, por exemplo recorrendo a valores de distância retornados pelos sensores, o espaço de estados resultante consistirá num espaço de estados cuja dimensão não é influenciada pela escala do ambiente.

4.4.2 Simplificação do mecanismo de aprendizagem

A necessidade de recursos é observável no algoritmo *Dyna-Q*. Quando o número de pares estado-acção aumenta, maior é a memória necessária. Adicionalmente, à medida que o número de simulações internas aumenta, maior é o processamento requerido. Porém, estes recursos justificam-se com o intuito de combater o tempo de convergência elevado inerente ao método *Q-Learning*.

Deste modo, uma abordagem de certa forma mais leve, é proposta. Ao invés de ser considerado um modelo interno ao agente, a sua memória é apenas constituída pela função valor $Q(s,a)$ e uma sequência episódica de estados s , acções a , reforços r e novos estados s' . Pelo que em vez de um modelo interno ter-se-á:

$$\text{Memória Episódica} \leftarrow (s,a,r,s')$$

Comparativamente ao *Dyna-Q*, o agente não irá escolher pares estado-acção aleatórios de um modelo interno. Durante n iterações simuladas o agente irá proceder à escolha aleatória de episódios passados, contidos na sua memória e irá realizar, para cada um, a actualização da função $Q(s,a)$ correspondente.

A sugestão deste método adiciona de facto agilidade à propagação dos sinais de reforço, mas, embora mais simples, este método não se encontra isento de desvantagens. Nem todos os episódios possuem a mesma utilidade, uma vez que os mais relevantes estão associados a reforços significativos (positivos ou negativos). Uma filtragem de sequências episódicas poderia levar a um melhor desempenho. Configurações prévias a episódios muito

significativos poderiam ser prioritárias de forma a propagar os valores mais eficazmente.

Este conceito de eliminação selectiva de memória pode traduzir-se num mecanismo de gestão de episódios que, operando conjuntamente com um limite de memória, poderão ser imprescindíveis quando o tempo operacional de um agente tende para infinito. Dado que a cada iteração um novo episódio (nova entrada) é criado na memória, numa perspectiva sem limite temporal os recursos acabarão consumidos.

4.5 Exemplo aplicação - *TD-Gammon*

Em 1992, *Gerry Tesauro* criou o programa *TD-gammon* com o objectivo de ser possível aprender e jogar o famoso jogo *Backgammon*. Não sendo um exemplo recente, a verdade é que esta aplicação do algoritmo *TD(λ)* com uma função linear de aproximação baseada numa rede neuronal multicamada mantém-se actual e constitui um importante caso de estudo.

Não entrando em detalhes sobre regras, num jogo *Backgammon* existem 24 posições (26 com a barra de jogo e fora do tabuleiro), 15 peças brancas, 15 peças pretas e um número considerável de possíveis movimentos (devido ao dado). Ao considerar todos estes detalhes, é possível constatar que o número de possíveis posições no jogo é muito superior quando comparado a um exemplo já complexo como o xadrez. A representação computacional destas possíveis posições, refira-se estados, torna-se impossível dado o factor de ramificação estimado de 400 para este jogo.

A rede *TD-gammon* criada por *Tesauro* tem uma camada de entrada com a representação do tabuleiro e uma camada escondida. A entrada de representação da primeira versão contém 198 entradas. A camada escondida recebe a projecção da entrada para 40 nós escondidos. A camada de saída da rede consiste em quatro unidades logísticas com o objectivo de estimar a probabilidade das peças brancas, ou pretas, alcançar uma vitória regular ou “fazer gamão”, estimando os quatro valores de resultado ao mesmo tempo. A abordagem combinada utilizada por *Tesauro* permitiu contornar a limitação dos recursos.

4.6 Conclusões

Numa perspectiva de escassez de recursos, o sacrifício de uma solução óptima, em detrimento de uma outra com menor qualidade, poderá ser considerado caso a nova solução permaneça útil e os seus ganhos se reflectam numa melhoria relativamente à convergência, emergindo nesse sentido, o conceito de satisfação.

Os seres humanos vivenciam um mundo real de escassez e limites. Mesmo o tempo e a informação parecendo escassos, é necessário tomar decisões de alto risco. Numa tomada de decisão nem sempre existe quantidade suficiente de tempo, dados ou capacidades para utilizar abordagens sofisticadas. Felizmente, as heurísticas, fornecem alguns atalhos de menor complexidade, mas prontamente activos, que actuam com um desempenho tão bom ou melhor em relação aos modelos sofisticados e mais elaborados, pelo menos no contexto de mundo real, com informação e tempo limitado.

Os conceitos e as heurísticas aqui apresentados não constituem soluções complexas ou na vanguarda da inteligência artificial, mas ainda assim, são surpreendentemente úteis. Estas heurísticas aproximam-se de uma abordagem mais biológica, uma inteligência artificial baseada na biologia.

5 Abordagens Propostas

Neste capítulo são apresentadas abordagens aos problemas referentes à limitação de recursos apresentados no capítulo 4. As abordagens encontram-se dispostas em função do seu foco, sendo introduzidas primeiramente métodos com recurso a heurísticas, técnicas de abstracção e por último, seguindo uma perspectiva muito limitada de recursos.

5.1 Aprendizagem com Utilização de Heurística

A heurística possui um papel fundamental na tomada de decisão, considerando alguns autores que é utilizada constantemente no comportamento humano e animal [13]. As abordagens propostas nesta secção inspiram-se na visão de racionalidade limitada e nos conceitos de heurísticas rápidas e frugais introduzidas na secção 4.2.2.

5.1.1 Integração de reactividade

Na perspectiva de um agente reactivo, os óptimos locais representam uma séria limitação neste tipo de arquitecturas. Para abordar esta problemática, a introdução de memória é fundamental para que o agente evite este tipo de situações. A aprendizagem por reforço apresenta-se como uma solução neste campo, no entanto, o seu tempo de convergência para uma solução satisfatória ou óptima está dependente quer dos recursos, quer da dimensão do ambiente e quantidade de informação a processar.

Uma arquitectura híbrida apresenta um potencial de resolução uma vez que tenta combinar as vantagens de ambos os métodos, a fim de obter melhores resultados em termos de resposta e desempenho.

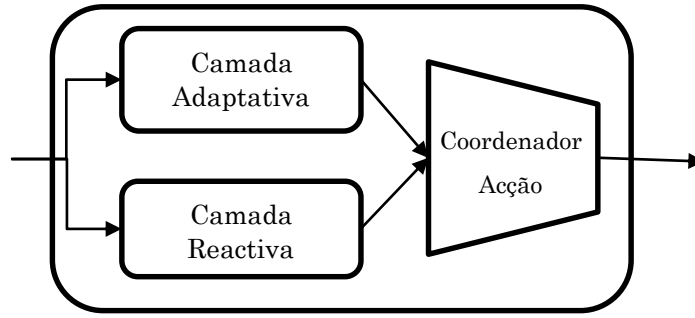


Figura 22 - Arquitectura horizontal de camadas do agente híbrido simples.

Para a questão abordada foi considerada uma organização horizontal com uma camada reactiva e outra adaptativa. Para resolver o dilema de mediar a escolha de acção presente neste tipo de organização, duas soluções diferentes para a coordenação são propostas.

Uma função coordenadora de acção consiste na verificação da existência da função valor Q para o par estado-acção onde o agente se encontra. A função para a coordenação C é apresentada de seguida.

$$C(s, a_r) = \begin{cases} \text{resposta adaptativa,} & (s, a_r) \in Q \\ \text{resposta reactiva,} & (s, a_r) \notin Q \end{cases}$$

Sendo s o estado actual do agente e a_r a acção que corresponde à acção gerada pela resposta reactiva. Esta solução revela-se proveitosa uma vez que a camada reactiva actua, neste contexto, como uma forma de exploração mais orientada. Esta orientação em relação ao objectivo advém do módulo comportamental “Seguir Potencial” integrado, que proporciona uma heurística fornecida através do campo de potencial. Mesmo entrando em zonas de óptimos locais, com a existência de comportamentos cíclicos, a função valor $Q(s, a)$ irá ser formada na camada adaptativa e, após algumas iterações, uma política comportamental, que orientará o agente a sair do máximo local, será produzida.

Uma outra política de selecção de acção para a camada híbrida consiste em apenas responder de modo adaptativo caso a função valor para o par estado-acção actual tiver valor negativo.

$$C(s, a) = \begin{cases} \text{resposta adaptativa,} & Q(s, a) < 0 \\ \text{resposta reactiva,} & Q(s, a) \geq 0 \end{cases}$$

Sendo s o estado actual do agente e a a acção para o momento actual do agente. A motivação desta abordagem deve-se ao facto de em ambientes sem existência de máximos locais, a arquitectura reactiva produzir um comportamento óptimo devido ao módulo comportamental de seguir potencial. Assim sendo, um comportamento adaptativo é apenas necessário aquando existem situações que o agente não consegue ultrapassar. Em situações de óptimos locais, a função Q encontra-se tipicamente negativa ao convergir ao fim de algumas iterações, levando à produção de uma política comportamental óptima no sentido de afastar o agente do local.

No entanto é necessário a definição de uma função de reforço que considere esta utilização particular da função $Q(s,a)$. Não poderão existir penalizações pela obtenção de objectivos devido à posterior propagação do valor positivo por os outros estados. Esta propagação poderá tornar valores negativos úteis de pares da função $Q(s,a)$ em valores neutros ou novamente positivos. Adicionalmente, o reforço negativo dado por efectuar um movimento normal no ambiente também não pode ser considerado uma vez que a função valor ficará negativa em regiões sem óptimos locais e actuará como falso positivo para o agente.

5.1.2 Método *Heuristic Sparse Learning* - *HSL*

O mecanismo de aprendizagem *HSL* combina uma introdução de heurística com um armazenamento esparsa, relativamente à propagação do sinal de reforço reflectido na função $Q(s,a)$. Nesse sentido, este algoritmo diferencia-se de uma abordagem híbrida pois, a camada de abstracção criada pela camada reactiva e adaptativa é inexistente. A heurística influencia directamente a aprendizagem e especifica que estado é, ou não, relevante.

A solução proposta atenua ambas as vertentes computacionais de recursos limitados. A capacidade de seguir um potencial heurístico, devido à política de selecção de acção específica, permite economizar recursos ao nível do processamento. Por outro lado, a definição de estados relevantes para aprendizagem equaciona a poupança de memória.

Este método sofre, no entanto, do mesmo tipo de problemas inerentes às propostas que recorrem a métodos heurísticos. Tal significa que, para cada problema, poderá ser difícil formular uma heurística, mesmo se possível. Sendo a aprendizagem por reforço aplicável de forma geral ao propósito de um problema, a representação de estado e a definição de uma heurística admissível são necessários em cada caso.

-
- 1) Iniciar $Q(s,a)$, s^* , S_q e M .
 - 2) Repetir (para cada episódio):
 - a) Iniciar s .
 - b) Repetir (para cada passo):
 - i. $a \leftarrow hsl \pi'(s)$
 - ii. Executar a e observar r e s'
 - iii. $\Delta H = H(s') - H(s)$
 - iv. Se $\Delta H \leq 0 \cup s \in S_q \cup s' \in S_q$:
 - a. $S_q \leftarrow s$ (Memorizar estado relevante)
 - b. $Q(s,a) \leftarrow Q(s,a) + a[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
 - c. $M \leftarrow (s,a,r,s')$ (Memorizar episódio)
 - d. Repetir N vezes:
 - i. $(s_i, a_i, r_i, s'_i) \leftarrow M_{random}$
 - ii. $Q(s_i, a_i) \leftarrow Q(s_i, a_i) + a[r + \gamma \max_{a'} Q(s'_i, a'_i) - Q(s_i, a_i)]$
 - v. $s = s'$
 - vi. $a = a'$
 - c) Até $s = s^*$
Até um parâmetro final ser atingido.

Figura 23 - Algoritmo *HSL*.

Política de selecção de accção $\pi(s)$

$$hsl \pi'(s) = \begin{cases} \arg \max_a H(s,a), & \text{caso } Q(s,a) = \emptyset \\ \varepsilon\text{-greedy } \pi(s), & \text{caso contrário.} \end{cases}$$

$$\varepsilon\text{-greedy } \pi(s) = \begin{cases} \arg \max_a Q(s,a) & \text{caso } q \leq p, \\ a_{random} & \text{caso contrário} \end{cases}$$

$$a_h = \arg \max_a H(s,a)$$

Função Heurística $H(s,a)$

$$H(s,a) = H(s'), \quad \text{para } s' = T(s,a)$$

$$H(s) = \begin{cases} \frac{1}{1 + D(s,s^*)}, & \text{caso } s^* \neq \emptyset \\ 0, & \text{caso contrário} \end{cases}$$

Onde s é o estado corrente, s' o próximo estado e s^* o estado final.
 D corresponde a uma função de distância euclidiana: $D = \|s - s^*\|$.

O método HSL faz uso da heurística de duas formas. Primeiramente a política de selecção de acção faz uso da heurística no caso de inexistência de informação presente na função Q para o par estado-acção corrente, onde a acção seleccionada resulta inteiramente da acção que obtém o maior valor da função heurística para o estado actual.

Secundariamente, a heurística é utilizada para avaliar a relevância do estado corrente com base na variação heurística calculada. Esta é considerada relevante caso seja negativa ou nula. Com base nesta avaliação, os estados que são considerados relevantes são adicionados a uma memória e incluídos no processo esparso de aprendizagem.

5.1.3 Arquitectura híbrida com camada reactiva com memória

No âmbito desta dissertação, a procura por uma solução que possibilite uma aprendizagem eficiente, e um comportamento expedito, apresenta um contraste com as limitações ocorrentes nesta área. Considerando um problema de configuração geral, a solução passa por conceber um método de aprendizagem por reforço que consiga prevalecer a situações com óptimos locais da forma mais rápida possível, quer em termos de aprendizagem, quer em resposta comportamental.

Num ambiente, onde óptimos locais sejam inexistentes, um agente reactivo pode cumprir o seu propósito e obter um comportamento óptimo. A solução, para que óptimos locais sejam ultrapassados ou evitados, consiste na necessidade de adicionar memória para fins de aprendizagem.

Com base nestas ideias, é possível definir a criação de um agente que detenha apenas dois propósitos genéricos: aproximar de uma solução e aprender a evitar óptimos locais. Este pensamento assenta convenientemente numa arquitectura híbrida, com a reactividade a possibilitar um aproximar da solução eficiente.

Assente na ideia da heurística do reconhecimento enunciada na secção 4.2.2, uma camada reactiva com memória tentará replicar a mesma num contexto de reconhecer situações anteriormente vividas e aplicar o seu reconhecimento para auxiliar à tomada de decisão.

Relativamente à componente de aprendizagem, uma mudança na linha de pensamento permite definir o objectivo de aprendizagem como simplesmente abandonar ou precaver zonas sub-óptimas no espaço de estados, ao invés de, atingir um determinado estado óptimo.

5.1.3.1 Organização geral da arquitectura

As arquitecturas híbridas são uma proposta para soluções onde o tempo de resposta é um factor indispensável. O critério de selecção de resposta entre as camadas consiste num problema de coordenação num contexto de coerência e boa tomada de decisão. Na perspectiva de um coordenador, a escolha de uma acção depende não só das respostas, mas principalmente de informação recolhida que permita obter o conhecimento suficiente de forma a tomar preferência na coordenação.

Uma camada intermédia alivia, neste contexto, o coordenador de processar toda a informação referente às restantes camadas e basear assim a tomada de decisão com base na heurística de reconhecimento embutida nesta camada.

Se a característica reactiva, com base na informação heurística, permite orientar o comportamento do agente, no contexto da camada adaptativa, um mecanismo de aprendizagem poderá também possibilitar e orientar a aprendizagem no sentido correcto da política de acção após transpor obstáculos sob óptimos.

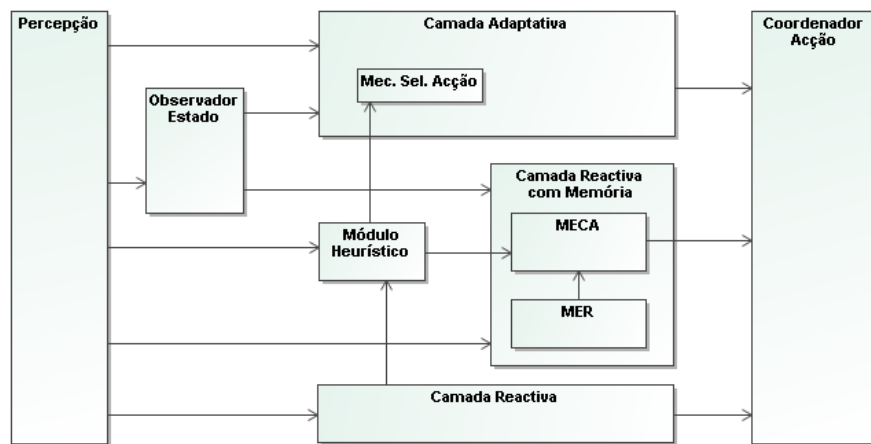


Figura 24 – Arquitectura híbrida com camada reactiva com memória.

Na Figura 24 – Arquitectura híbrida com camada reactiva com memória, são identificados os componentes da arquitectura, através de blocos, e os fluxos de informação, por setas. Os termos MECA e MER definem respectivamente a Máquina de estados comutadora de acção e a Memória de estados relevantes. A arquitectura é semelhante à utilizada na integração de reactividade, mantendo-se a organização horizontal, existindo agora uma camada adicional intermédia. Cada componente é descrito nas secções que se seguem.

5.1.3.2 Observador de Estado

A arquitectura proposta integra diferentes componentes com propósitos bem definidos. Um dos critérios importantes relativos à robustez passa por permitir a definição universal de um problema de aprendizagem. A concepção de um módulo que possa ser substituído em função do estado desejado fornece as características desejadas nessa linha de pensamento.

O conceito de observador de estado integra esta ideia no sentido em que permite a formalização de um estado consoante as necessidades impostas pelo problema de aprendizagem. Esta imposição implica, assim, a criação de observadores de estado específicos, mas não únicos, e de acordo com as características do ambiente. Os métodos de aprendizagem abstraem-se assim de qualquer possível estado, sendo os seus mecanismos usados de forma genérica.

5.1.3.3 Módulo Heurístico

Seguindo a vertente modular do observador de estado, o módulo heurístico contém a especificação de diferentes funções ou modelos heurísticos conforme a abordagem. Por sua vez, a informação heurística pode ser alimentada em função da percepção, consoante o critério heurístico pretendido.

A função heurística é posteriormente utilizada de forma a fornecer informação de variação de potencial à camada intermédia e à camada adaptativa, esta última se necessário.

5.1.3.4 Subsistema Reactivo

A componente reactiva pretende dotar o agente com uma melhor capacidade de resposta na navegação do ambiente quando não existe obstáculo. Este facto advém da solução reactiva, com seguimento de potencial, obter um caminho óptimo em campo de exploração livre. Neste sentido, este subsistema garante racionalidade máxima ao alcançar uma política comportamental óptima em áreas ausentes de óptimos locais.

Em função dos estímulos esta camada dará uma resposta reactiva de acordo com os esquemas comportamentais definidos. A mesma resposta será depois alvo de critério de selecção pelo coordenador de acção.

5.1.3.5 Subsistema Intermédio – Reactivo com Memória

O subsistema intermédio é responsável por agregar informação pertinente e suficiente de forma a determinar o macro estado do agente. Esse mesmo macro estado define uma política para escolha da resposta entre camadas com base no reconhecimento de situações. Esse estado macro é futuramente usado como referência na coordenação de acção do agente aquando da selecção da política comportamental.

O processamento da camada intermédia reúne dois componentes com esse objectivo em linha: uma máquina de estados comutadora de acção (MECA) e uma memória de estados relevantes (MER). Estes intervenientes auxiliam a camada na detecção de macro estado e eventual comutação.

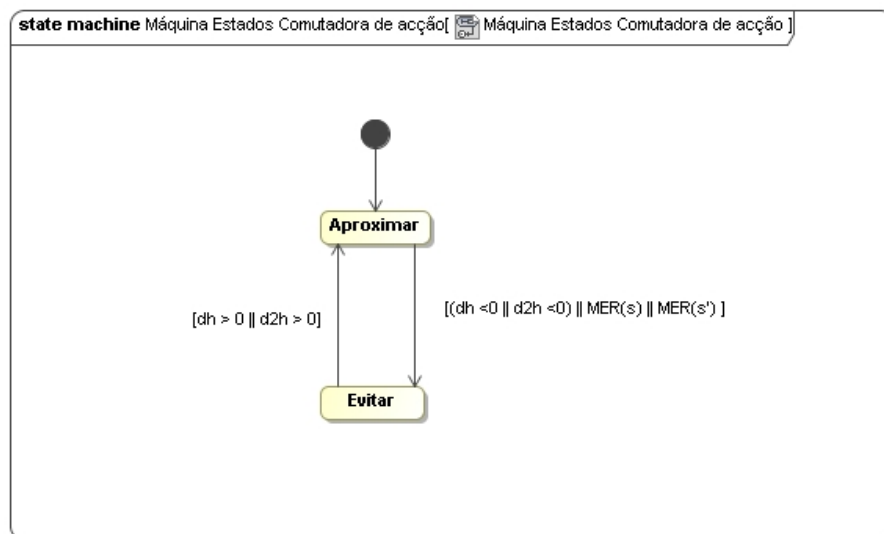


Figura 25 - Dinâmica da máquina de estados comutadora de acção.

Os estados considerados desta camada são dois: aproximar e evitar. De acordo com os objectivos da arquitectura do agente, a sua vocação terá sempre dois formatos, correspondente a aproximar do estado objectivo ou evitar um obstáculo / óptimo local.

A comutação entre estes estados representa um sério desafio na definição das próprias condições de transição. Estas necessitam de ser

consideradas num contexto genérico e independente do ambiente. Deste modo, a solução encontrada passou por estudar o comportamento da variação de potencial.

Sinal		Macro estado
Variação Potencial (dH)	Derivada (d2H)	
+	+	Aproximar
-	+	Evitar
-	-	Evitar
-	-	Evitar

Tabela 1 - Estudo do macro estado em função da monotonia da variação de potencial e da derivada.

O estudo da variação de potencial e a avaliação de sinais permite obter uma heurística de reconhecimento simples e binária relativamente à identificação de uma situação favorável ou não ao agente. Estas, associadas à memória de estados relevantes (MER) possibilita a identificação de situações onde o agente pode aproximar alvo, escolhendo a resposta reactiva, ou evitar obstáculo, seleccionando a resposta adaptativa.

5.1.3.6 Subsistema Adaptativo

O subsistema adaptativo processa paralelamente a informação percebida em conjunto com as restantes camadas. Nesse sentido, uma das características remete para a sua versatilidade pois é possível utilizar qualquer mecanismo de aprendizagem por reforço.

No entanto, a maior particularidade existente na camada adaptativa prende-se com a utilização de um mecanismo de aprendizagem que recorra a informação heurística, como alguns já abordados neste trabalho. Em caso de necessidade, o módulo heurístico propaga a devida informação para a camada adaptativa e por consequente para o método requerente.

A configuração do mecanismo de aprendizagem a utilizar é realizada sem pormenores e à semelhança dos agentes adaptativos não híbridos.

Apenas existe a necessidade de prevenir problemas na coerência das acções entre as camadas.

5.1.3.7 Coordenador de Acção

O coordenador de acção dispõe a prioridade das acções a serem realizadas com base na informação transmitida dos subsistemas. O macro estado do agente, fornecido pela camada intermédia, é ponderado e o tipo de resposta (reactiva ou adaptativa) é escolhida, resultando numa política comportamental distinta consoante o objectivo actual do agente.

Esta arquitectura estabelece dois fundamentos gerais bem explícitos: a camada reactiva obtém comportamento óptimo para alcance do objectivo e, a camada adaptativa é responsável por aprender a evitar quaisquer estados sub-óptimos que constituam obstáculo à componente reactiva.

Política de selecção de acção $\pi(s)$

$$\pi(s) = \begin{cases} \textit{resposta adaptativa}, & \textit{caso evento relevante} \\ \textit{resposta reactiva} , & \textit{caso contrário.} \end{cases}$$

Onde o evento relevante corresponde ao critério de zona de aprendizagem ou não. Com base na decisão relevante de aproximar ou evitar, o coordenador processa a acção seleccionada e retorna-a no formato para actuação.

5.2 Aprendizagem com Abstracção de Estado

A redução de detalhe fornece uma ferramenta poderosa para a técnicas de abstracção no sentido de diminuir a complexidade. Nesta secção são abordadas diferentes perspectivas de abstracção e ainda proposto um modelo geral de aprendizagem com abstracção sobre o qual duas abordagens assentam.

5.2.1 Agente adaptativo *Deep-Q* adaptado

Adaptar a arquitectura *Deep-Q*, sobre um contexto diferente de jogos de consola, requer uma ponderação relativamente aos parâmetros que influenciam a aprendizagem. Em primeiro lugar, a estrutura interna da rede neuronal difere em função do problema de aprendizagem. No caso original, uma rede convolucional é necessária para obter o estado; no entanto, num outro contexto, tal pode não ser útil. Deste modo, é assumida apenas a manutenção da estrutura discriminadora, cuja rede neuronal *feed-forward* multicamada define.

Adicionalmente, o próprio conceito de episódio necessita de ser aplicado ao problema em questão. Um episódio original corresponderá a uma sessão de jogo até o sistema perder. Numa visão generalista, poderão existir duas vertentes de definição de um episódio. Numa consideração, um episódio pode corresponder ao acontecimento de uma dada transição de estado, ou seja, na óptica da aprendizagem por reforço pode corresponder à informação observada aquando da transição de estado (s,a,r,sn) .

Noutro contexto, um episódio poderá corresponder ao conjunto de transições e informação correspondente por onde um agente atravessou até atingir um determinado estado objectivo. De forma complementar, também a dimensão do número de episódios escolhidos para treino (a cada iteração do algoritmo), pode diferir em função do problema e mesmo do conceito de episódio escolhido.

O parâmetro Q pode também influenciar bastante o desempenho do agente uma vez que, corresponde à frequência de actualização da rede neuronal, está correntemente em utilização para tomada de decisão das acções. Um valor de Q elevado poderá traduzir-se numa resposta lenta em termos adaptativos, mas mais estável em termos comportamentais. Inversamente, um Q baixo poderá reproduzir numa adaptação mais rápida. No entanto este parâmetro faz um maior sentido numa perspectiva de treino paralelo da rede ao invés de *online*. A justificação para esse facto prende-se

com a necessidade de não desperdiçar o conhecimento aprendido quando não existe enquadramento distribuído.

Definir a estrutura da rede neuronal exige modificações relativamente à rede original do algoritmo proposto. Tal sucede-se, pois, o ambiente de operação é diferente. O estado do agente permanece a entrada da rede, embora num novo formato de duas entradas: uma para cada coordenada da posição do agente. Porém, a saída de rede é modificada para corresponder ao valor da função $Q(s,a)$ para uma dada acção a ; existindo um número de saídas na rede equivalente ao número de acções praticadas pelo agente.

Para investigar a possibilidade de se aplicar uma arquitectura neuronal num agente adaptativo é necessário realizar algumas verificações prudentes. Um estado, equivalente à posição do agente, e um ambiente simples, não exprimem facilidade num contexto de aprendizagem de uma rede neuronal. Por esta razão, a realização de testes à credibilidade da rede neuronal e sua capacidade de aplicação ao problema são importantes. Assim, é possível atentar uma abordagem inicial de realizar uma tentativa de aprendizagem de uma política, previamente aprendida, expressa num função $Q(s,a)$.

Seja no contexto de operação de um agente *Deep-Q* ou na aprendizagem de uma função $Q(s,a)$ já preenchida, o método de treino da rede neuronal a ser escolhido influencia o desempenho e o tempo de convergência para a aprendizagem. Nesse sentido, uma abordagem estocástica faz sentido tendo em conta as vantagens e poupança temporal característica destes métodos.

5.2.2 Modelo geral de aprendizagem com abstracção de estado

As técnicas de abstracção necessitam de responder a duas questões fundamentais. A primeira parte remete para o critério de quando abstrair ou manter detalhe. O segundo pormenor, envolve a identificação de situações onde deve ser realizada aprendizagem, ou seja, verificar se o estado a um nível superior de abstracção se mantém para a percepção seguinte. Se dois estados foram percebidos e resultarem no mesmo estado abstracto, não existe sentido em aprender. Desta forma apenas as transições ao nível da abstracção devem ser passíveis de aprendizagem.

Para se compreender melhor os contornos dos pontos referidos anteriormente, um modelo geral para aprendizagem recorrendo a abstracção de estado é proposto, no sentido de apresentar e modelar a lógica de aprendizagem.

```

Iniciar  $Q(s,a)$  arbitrariamente e Memória de Aprendizagem de Abstracção  $M_A$ .
Definir limiar de relevância para o reforço  $\lambda_r$ .
Repetir (para cada episódio):
  Iniciar  $s$ .
  Repetir (para cada passo):
    Escolher  $a$  de  $s$  utilizando a politica derivada de  $Q$  (ex:  $\epsilon$ -greedy)
    Executar  $a$ 
    Observar  $r, s'$ 
    Obter estados abstractos  $S_A \leftarrow M_A(s)$  e  $S'_A \leftarrow M_A(s')$ 
    Se  $|r| > \lambda_r$  (reforço relevante) ||  $S_A \neq S'_A$  (existe transição):
       $M_A \leftarrow (s,a,r,s')$  (registar experiência)
       $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$ 
       $a \leftarrow a'$ ;
       $s \leftarrow s'$ ;
    Até  $s$  ser terminal.

```

Figura 26 - Algoritmo geral de aprendizagem por reforço recorrendo a abstracção de estado.

A memória de abstracção representa a estrutura de memória de aprendizagem de uma forma genérica, sendo que a mesma deverá garantir os mecanismos necessário que permita realizar a abstracção de estado, através do registo de experiências, e retornar um estado abstracto quando requerido. O limiar de relevância para o reforço λ_r é um parâmetro cujo propósito prende-se com definir um patamar de saturação, ao qual um reforço associado numa transição de estado é relevante ou não.

5.2.3 Aprendizagem hierárquica *quadtree*

A discriminação de estados constitui um desafio no contexto da sua representação, sendo necessário lidar com a multiplicidade de estados possíveis. As *Quadtree* são estruturas de dados bidimensionais úteis na representação ou recuperação eficiente da informação presente. Este tipo de estrutura armazena a informação da superfície através da subdivisão recursiva em rectângulos, que por sua vez permitem obter um melhor detalhe.



Figura 27 – Imagem original e a mesma comprimida através da *Quadtree* [16].

Este comportamento resulta numa estrutura em árvore constituída por nós com diferentes níveis de resolução. A sua criação contempla três componentes principais: o algoritmo de construção, de acesso e de medição de detalhe. Este último define o critério sobre o qual a construção da representação se processa, ou seja, avalia se o detalhe actual é suficiente ou se é necessário subdividir.

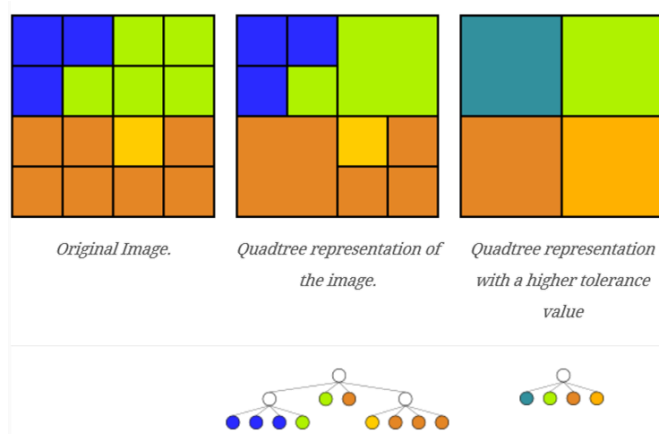


Figura 28 - Exemplo de uma representação em árvore [16].

Embora este tipo de estrutura seja maioritariamente aplicada no domínio da compressão de dados, nomeadamente na compressão de imagens, existe potencial para aplicar esta estrutura na aprendizagem por reforço sob a forma de uma memória de aprendizagem. Esta concepção permite executar e aprender num nível superior de abstracção e generalizar a política comportamental para estados agregados e sem informação relevante, reflectindo-se numa melhor eficiência e desempenho do agente em atingir o alvo.

Contudo, é necessário adaptar a criação desta estrutura para compatibilizar com o mecanismo e processo de aprendizagem. O algoritmo de construção é decomposto num processo de inserção, onde o aumento de detalhe está dependente do reforço obtido ser ou não relevante para um dado critério, e num algoritmo de actualização, onde cada nó representativo de um macro estado irá actualizar a sua função valor.

5.2.4 Aprendizagem com memória selectiva de instâncias

O processo de concepção que envolve este tipo de memória de abstracção, resulta do culminar da investigação e aquisição de conhecimento das diversas abordagens exploradas. Com base no comportamento e nos bons indicadores obtidos utilizando a estrutura *Quadtree*, esta remete para o potencial das técnicas baseadas em abstracção. No entanto, a *Quadtree* é específica da representação de estado bidimensional e, adicionalmente, tem problemas na transição entre estados abstractos, por exemplo, onde surgem transições em diagonal. A definição de um método de abstracção compatível e mais robusto à dimensionalidade de diferentes espaços de estados é assim pretendido.

O conceito de instância define uma memória de estado específica. Essa memória pode, no entanto, ser generalizada se a região onde se encontra não estiver associada a experiências relevantes, nomeadamente no que se refere aos reforços obtidos. A influência destas instâncias pode assim ocorrer a diferentes níveis de detalhe, nomeadamente a nível concreto e a nível abstracto. O controlo deste detalhe e da sua influência pode ser concretizado sob a forma de um raio de abstracção.

Uma motivação adicional para a concepção deste método prende-se com a criação iterativa de instâncias, não existindo necessidade de pré-definir um número fixo, como acontece em processos de classificação com centróides como é o caso do algoritmo K-Médias.

A função de acção-valor e de política serão geradas com base nestas instâncias. A função acção-valor $Q(s,a)$ ficará associada a cada estado presente na memória de instâncias. Para compreender este novo tipo de abstracção é importante identificar os seus componentes:

- Memória de Instâncias – Define o conjunto de instâncias que foram iterativamente criadas. Representam um estado que é utilizado na aprendizagem.
- Memória Selectiva – Define o conjunto de estados no qual ocorreram eventos relevantes. A estas memórias está associado um raio selectivo.

Para definir o raio de influência e critério de criação de memória selectiva são especificados alguns raios com propósitos explícitos:

- Raio de detalhe – define um raio para recuperação e teste da existência de instância para uma região de detalhe mínimo. Por consequente especifica a granularidade desejada na criação de instancias para um ambiente.
- Raio selectivo – define a dimensão da região de influência para uma memória selectiva.
- Raio de abstracção – define a dimensão da região abstracta para uma memória de instância.

O mecanismo de memória selectiva de instancias é apresentado de seguida, enquadrando-se o mesmo no algoritmo geral aprendizagem por abstracção descrito anteriormente.

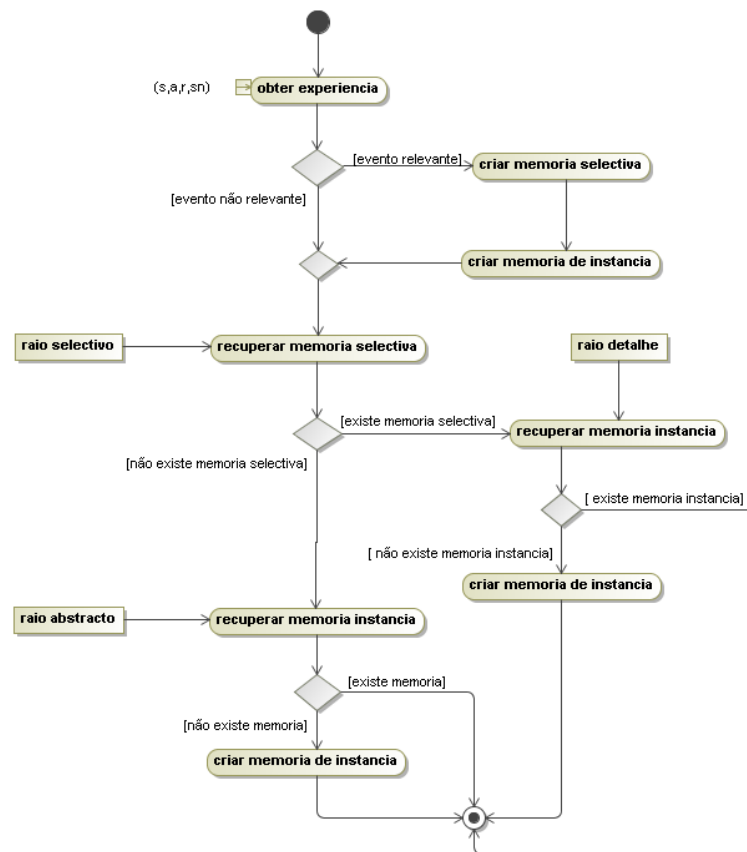


Figura 29 - Algoritmo do mecanismo de memória selectiva de instâncias apresentado sob a forma de um diagrama de actividade.

Com a criação de memórias a ser assegurada pelo mecanismo e algoritmo de criação, prende-se a questão da recuperação de memórias. Para um estado recebido, este será comparado com as memórias de instância existentes de acordo com uma medida de semelhança. Esta medida é concretizada sob a forma de uma métrica de distância, que é posteriormente usada para calcular qual a memória de instância mais próxima, se existir. Se a memória mais próxima existir, o raio de influência da memória é comparado à distância obtida e, caso a mesma esteja dentro da região definida, a memória é recuperada. O algoritmo de recuperação de memória é apresentado de seguida.

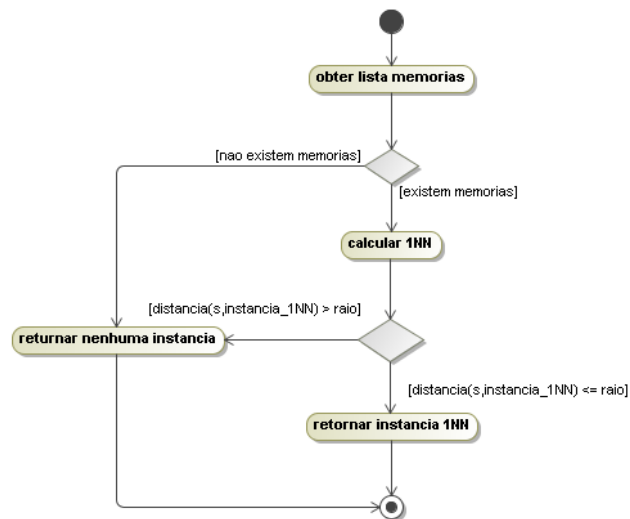


Figura 30 - Algoritmo de recuperação de memória.

Esta memória de aprendizagem baseada em abstracção não implica qualquer alteração aos mecanismos de aprendizagem ou à própria estrutura do agente adaptativo. Sendo completamente modular, esta técnica é compatível com a integração de reactividade.

5.3 Aprendizagem com Recursos Muito Limitados

Em cenários de operação real, as representações de estado estão não só condicionadas pelas características do ambiente, mas também, devido a restrições intrínsecas ao sistema. Nesta secção são abordadas propostas no sentido de estudar tais implicações, começando por compreender e explorar a representação limitada de estado no domínio da simulação e aplicar posteriormente numa vertente robótica.

5.3.1 Representação limitada de estado

A simplificação do estado representa um potencial de investigação para a redução de complexidade. O mesmo problema pode permitir alcançar uma solução, de objectivo semelhante, considerando representações de estado distintas.

Como representação de estado do agente, o seu posicionamento relativamente a um referencial cartesiano permite obter um estado composto por coordenadas absolutas. Esta representação é útil pois contempla que, cada estado seja singular, não sendo possível a ocorrência de ambiguidades no espaço de estados. Embora a singularidade seja proveitosa, manter uma representação deste tipo revela-se difícil de manter num contexto de escalabilidade ou de operação em cenário real. Neste contexto, uma representação que se abstenha destes factores limitativos é pertinente.

Constatando os processos biológicos dos seres vivos, a obtenção da informação exterior é realizada através dos diferentes mecanismos sensoriais. Esta abordagem, motiva a elaboração de um estado que poderá ser robusto quanto à escalabilidade do ambiente. Neste sentido, ao invés de uma posição observada, o estado do agente consistirá na informação sensorial captada. Contudo, é imprescindível atentar que, a qualidade e proveito do estado varia em função do tipo de sensores contidos no agente e na forma como essa informação é processada.

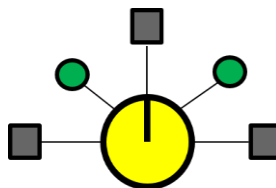


Figura 31 - Esquema exemplificativo de sensores contidos num agente; Quadrados representam sensores de distância e círculos os sensores de potencial.

Considerando diferentes tipos de sensores é possível formular diferentes tipos de estados sensoriais. Em termos concretos e exemplificativos, um estado, designado sensorial, pode conter informação das distâncias captadas em cada direcção onde existem sensores ou até, conter informação de variação de potencial.

- $S = (Distância_{frente}, Distância_{esquerda}, Distância_{direita})$
- $S = [(Distância_{frente}, Distância_{esquerda}, Distância_{direita}) , (dP_{esquerda}, dP_{direita})]$

Conjuntamente, se o agente for dotado de melhores sensores e capacidade de processamento, o estado sensorial pode evoluir no sentido de acrescentar informação de novos sensores ou, fornecer mais detalhe nos utilizados. Por exemplo, a informação de distância captada pode dar lugar a um sonar com identificação do obstáculo detectado. Em contexto de simulação estas diferentes formulações são alcançáveis devido à versatilidade da plataforma de simulação de agentes.

A redução de complexidade, que esta simplificação acarreta, conduz consequentemente a uma menor exigência em termos de recursos ao nível do processamento. O tempo de convergência pode assim diminuir, sem modificar os mecanismos adaptativos.

Com estas diferentes definições, o espaço de estados tomará uma forma díspar, deixando de ser sequencial na transição, como acontece com um estado posição. Este facto, aliado à posição relativa em termos direccionais, pode levar o agente a percorrer o ambiente de uma forma não linear, mas em concordância e fazendo sentido no percurso do espaço de estados.

Um agente pode percepcionar estados distintos como sendo o mesmo, em função da qualidade discriminativa da informação. Esta ocorrência, nomeada de *aliasing* de estado (*state aliasing*), provoca uma aprendizagem que pode ser extrapolada para situações semelhantes, mas que, o agente identifica como sendo a mesma. Não obstante aos problemas, é necessário encontrar um equilíbrio na elaboração do estado, para que não existam demasiados estados iguais. Neste aspecto, a escolha de uma representação de estado necessita de ser estudada com o intuito de validar a compatibilidade com o problema.

5.3.2 Vertente robótica

A vertente robótica tem a motivação de aplicar em ambiente real alguns agentes implementados. Para que a utilização dos agentes seja realizada com o mínimo de alterações necessárias, a plataforma robótica de agente (PRA) compatibiliza e incorpora toda uma interface semelhante à plataforma de simulação de agentes (PSA). Esta abordagem permite manter a coerência em termos funcionais, pressupondo o uso das interfaces estabelecidas para o agente.

O agente robótico mantém o mesmo problema do agente simulado em cenário real: o agente necessita de atingir um estado objectivo, ultrapassando um ambiente com determinada configuração de obstáculo. Em condições reais, contudo, o objectivo definido não pode ser levemente considerado como um alvo abstracto, como acontece na simulação. Nessa linha de pensamento é importante transpor o desafio simulado para uma circunstância real mantendo toda a ideia e conceito base da problemática.

A abordagem mais semelhante encontrada, em relação ao contexto da simulação, consiste num agente robótico seguidor de luz. O agente robótico terá analogamente sensores e actuadores, traduzindo-se estes por sua vez em sensores paralelos à simulação.

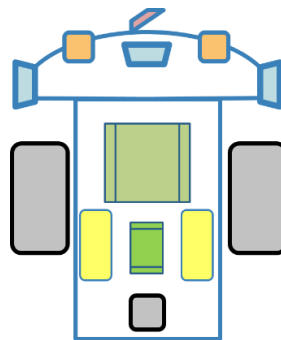


Figura 32 – Esquema de vista superior dos componentes do robô. É possível notar os sensores de ultra-sons a azul, os sensores luminosos a laranja, motores a amarelo, unidades de processamento a verde e por fim as rodas a cinzento.

Sensores luminosos correspondem a sensores de potencial e sensores de ultra-sons correspondem a uma versão simplificada do sonar existente na PSA, que apenas mede distâncias. Por sua vez, o actuador de movimento dará lugar a um par de rodas motorizadas, cuja intensidade e direcção de rotação permitirá a realização flexível de movimento em diversos sentidos.



Figura 33 – Ilustração simplista do ambiente em contexto de operação real.

O alvo a atingir é a origem de potencial, concretizada sob a forma de uma fonte emissora de luz. Esta consistirá em qualquer instrumento que cumpra tal propósito, desde que a intensidade seja suficiente detectável. Na medida dos obstáculos, estes consistem em qualquer objecto que possibilite criar o desafio pretendido ao agente, com destaque a deterem superfícies lisas uniformes para melhor leitura dos sensores de distância.

5.4 Conclusões

As vertentes a explorar inserem-se em três visões com diferentes focos sobre os recursos limitados. Os métodos que fazem uso de heurística inspiram-se nos conceitos das heurísticas rápidas e frugais, integrando essa informação através de uma camada reactiva ou realizando uma análise da variação de potencial.

A abstracção de estado assenta no conceito de satisfação do domínio da racionalidade limitada, tendo sido proposto um método geral de aprendizagem por abstracção e duas técnicas que têm em conta os conceitos de abstracção identificados.

6 Concretização

Neste capítulo são descritas as várias fases da componente prática realizada, relacionando a sua concretização com o respectivo enquadramento teórico. Primeiramente é introduzida e explicada a caracterização da arquitectura geral do agente inteligente. De seguida são abordadas as diferentes arquitecturas e soluções concebidas representativos das várias técnicas investigadas. Por fim é descrita a implementação da arquitectura de recursos limitados concebida e da vertente robótica.

6.1 Arquitectura Geral de Agente

O agente, um sistema autónomo inteligente, possui a mesma arquitectura a um nível abstracto e independentemente da abordagem. O agente percepção o ambiente em seu redor, realiza o seu processamento interno e actua através de acções geradas desse processamento. Este nível de abstracção permite modularizar o processamento interno do agente quando da sua concretização.

Para o trabalho proposto, um agente prospector será implementado, estendendo o conceito de agente definido na plataforma PSA e contendo um sensor múltiplo e um actuador. De forma a definir o processamento interno, uma interface controlo é especificada com o objectivo de modularizar os diferentes controlos a serem implementados.

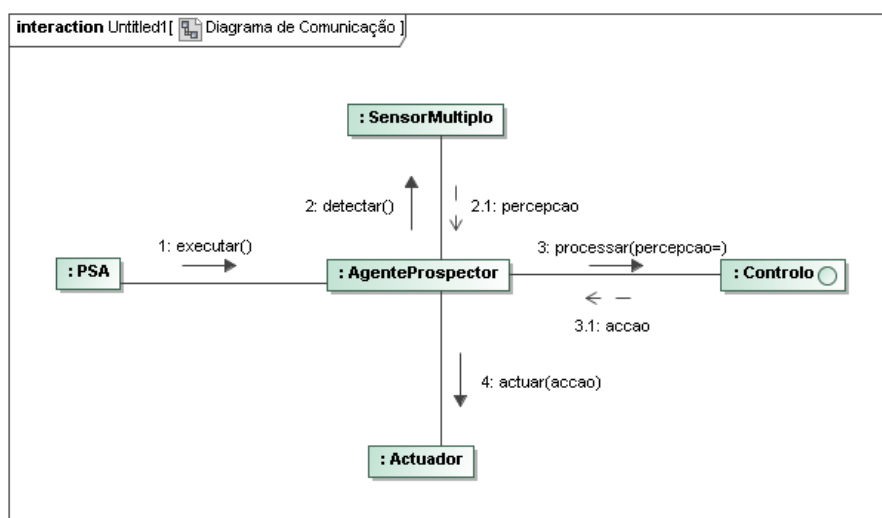


Figura 34 - Diagrama de comunicação do agente prospector.

Na vertente da engenharia de *software*, esta conduta é vantajosa pois, permite o total foco no estudo e implementação dos diferentes paradigmas de controlo e arquitecturas inerentes ao processamento interno do agente.

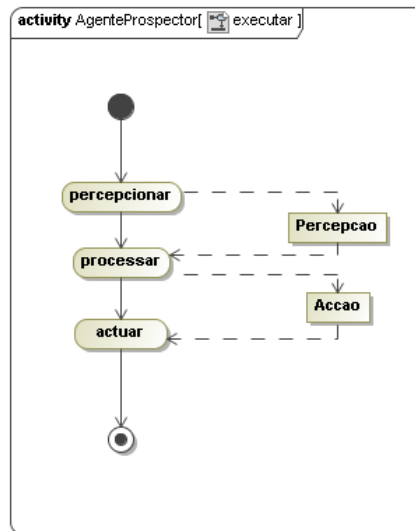


Figura 35 - Diagrama de Actividade Executar do Agente Prospector.

6.2 Aprendizagem por Reforço Geral

Nesta secção são descritas as concretizações gerais em relação aos mecanismos base da aprendizagem por reforço. A estrutura base destes mecanismos é implementada através da biblioteca criada para o efeito.

6.2.1 Estrutura base – biblioteca AprendRef

A modularidade da arquitectura do agente permite a criação de um controlo específico para a aprendizagem por reforço. Sendo um objectivo a implementação de técnicas adaptativas, nomeadamente dos algoritmos *Q-Learning* e *Dyna-Q* (extensões ao algoritmo base *SARSA*), foi definida a biblioteca “*AprendRef*” com a estrutura de classes representativa destes algoritmos.

Precisamente por estes algoritmos serem extensões da ideia base do algoritmo *SARSA* é possível identificar semelhanças funcionais e colocá-las em evidência através do uso da herança. Assim, à classe base de

aprendizagem por reforço foi possível adicionar as restantes características particulares de cada algoritmo.

Independente do algoritmo a usar, a aprendizagem por reforço necessita de duas componentes adicionais que garantem o seu funcionamento. O mecanismo de selecção de acção que, sendo definida como uma interface, permite a criação de diversos mecanismos de selecção de acção baseados em políticas distintas. Em termos concretos, para possibilitar explorar o suficiente sem um aproveitamento constante foi concretizado o mecanismo de selecção de acção ε -greedy. Este tipo de selecção de acção considera uma probabilidade ε de o agente decidir explorar ao invés de seleccionar a acção que maximiza o valor da função $Q(s,a)$ para o seu estado corrente.

No contexto da aprendizagem é necessária memória de forma a armazenar a função valor para os pares. No entanto, proceder à pré-alocação de espaço para os valores possíveis pode relevar-se dispendioso em termos de recursos. Portanto, para evitar o alocamento de toda uma função $Q(s,a)$, recorreu-se ao conceito de memória esparsa. Esta permite de uma perspectiva exterior assumir que, a memória contém toda a informação necessária, mas interiormente apenas existem os valores previamente inseridos. Quando é realizado um acesso sobre um valor inexistente, é devolvido um valor padrão inicialmente definido. A memória esparsa foi implementada sob a forma de uma classe com recurso a um dicionário, um tipo particular de estruturas de dados em *python*.

A dinâmica interna, associada ao controlo do agente na aprendizagem por reforço, passa primeiramente por observar o estado para o qual o agente transitou. Com essa informação, fornecida pela percepção, o agente utiliza o seu mecanismo de selecção de acção no sentido de obter a próxima acção a realizar, acção essa, escolhida de acordo com a política comportamental definida. Caso uma acção seja seleccionada, um valor de reforço é calculado com base na função de reforço e percepção existente. Obtido o reforço, estão reunidos todos os parâmetros para que o mecanismo de aprendizagem execute um passo aprendizagem e actualize a função valor. Por fim, a informação do estado percebido e da acção seleccionada são guardados para a próxima iteração temporal e a acção seleccionada é executada.

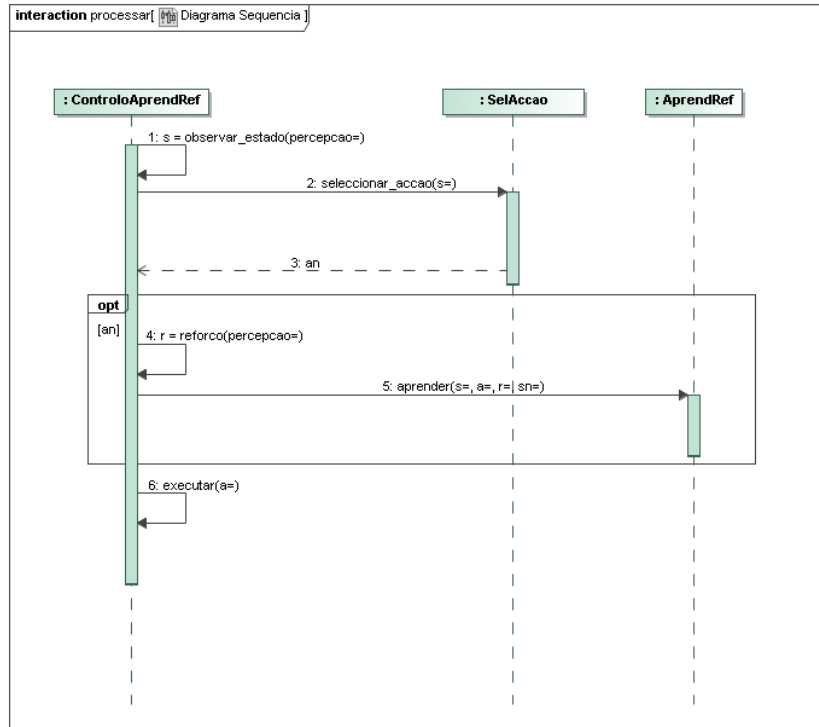


Figura 36 -Diagrama de sequência do processar do controlo adaptativo.

A função reforço, embutida neste controlo, tem em vista o uso exclusivo de informação sensorial básica por parte do agente. Esta condição implica um agente cego, ou seja, não existe visibilidade do ambiente e apenas é detectada a existência de colisão ou situação de alvo recolhido. Assim, a função de reforço *R* retorna um reforço *r* mínimo negativo equivalente ao custo de um movimento ou, pode somar um reforço máximo positivo, em caso de recolher um alvo, ou somar um reforço máximo negativo, em caso de colisão.

A incorporação da função de reforço faz sentido na medida em que a função existente é adequada ao problema simulado. No entanto, de forma a generalizar e modularizar esta componente de aprendizagem, é possível fazer uma refactorização à forma como o reforço é integrado na arquitectura. De forma semelhante ao mecanismo de selecção de acção, é possível criar uma interface *Reforco* que obrigue à implementação do respectivo método. Desta forma, o controlo fica totalmente modular em relação aos outros componentes, com a delegação da geração do sinal de reforço para classes próprias com implementação própria de suas caracterizações de reforço.

6.2.2 Implementação agente *Q-Learning*

A implementação do mecanismo *Q-Learning* utiliza a base estabelecida e implementa o seu método de aprendizagem. Utilizando o mecanismo de selecção de acção ϵ -greedy e a componente de memória de aprendizagem (sob a forma de memória esparsa) é possível concretizar o processo de aprender.

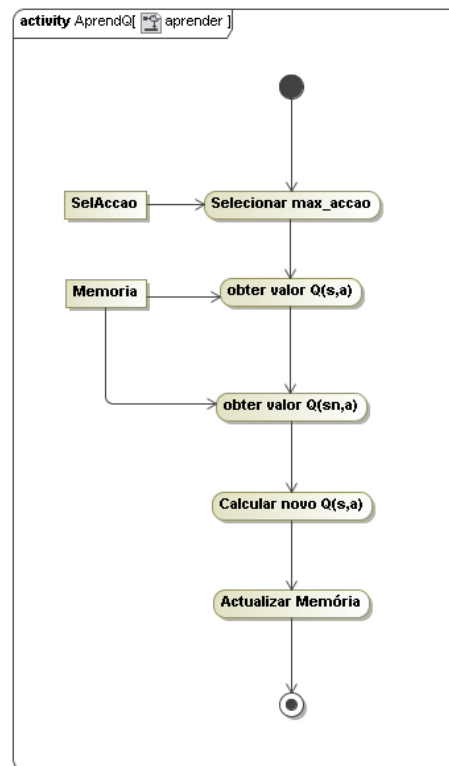


Figura 37 - Diagrama de actividade aprender *Q-Learning*.

Primeiramente, o valor para a função valor Q é obtido com base do par estado-acção (s,a) . De seguida, a acção que maximiza a função valor para o estado s' é determinada. Esta acção a' é depois utilizada em conjunto com o estado s' de forma a obter o valor da função Q correspondente. Posteriormente, este valor é usado no cálculo e actualização do novo valor Q para o par (s,a) , com base na expressão de propagação de valor.

No entanto, a propagação da função $Q(s,a)$ é um dos principais problemas do *Q-Learning*. O valor é apenas propagado para pares estado-acção adjacentes, sendo demoroso e consumidor de tempo para convergir numa solução. Este problema é acentuado em função da complexidade do ambiente, onde a multiplicidade de estados existentes pode resultar num tempo elevado, quer de exploração, quer de convergência.



Figura 38 - Execução do agente *Q-Learning* no ambiente 3 com detalhe 1.

A figura anterior mostra o agente adaptativo com mecanismo *Q-Learning*, sendo visível no visualizador a função $Q(s,a)$ construída. Mesmo considerando ambientes complexos este método garante a obtenção, em algum momento, de uma solução óptima. Porém, a política óptima, ou até mesmo uma já satisfatória, pode não ser obtida num tempo útil.

Outro aspecto desta implementação é o mecanismo de selecção de acção ϵ -greedy, que permite determinar se o agente explora em detrimento de aproveitar. O ajuste deste parâmetro ϵ pode auxiliar o agente no sentido de determinar outros caminhos possíveis para chegar ao(s) estado(s) objectivo(s). Contudo, o método ϵ -greedy representa também um problema: mesmo com uma política óptima encontrada, não existe garantia que o agente a irá executar continuamente pois, o mecanismo de selecção de acção pode gerar, em determinado momento, uma escolha de acção exploratória. Neste seguimento, o uso de um mecanismo de selecção de acção ϵ -greedy mais sofisticado, que varie o valor da probabilidade de exploração ϵ em função de uma heurística que traduza o quanto o ambiente já foi explorado, pode revelar-se vantajoso.

Avaliando o desempenho do agente *Q-Learning* em termos práticos, verifica-se que, mesmo num ambiente simples como o ambiente 3 com detalhe 1, a obtenção de um resultado satisfatório é um sucesso relativo. Este agente efectivamente ultrapassa a limitação do óptimo local do agente reactivo, no entanto, o tempo de convergência para uma solução satisfatória ou mesmo

óptima, é consideravelmente elevando. Devido a essa situação, uma consideração de ambientes com multiplicidade de estados mais elevada é desencorajada.

Adicionalmente, é importante denotar a orientação singular em relação a um estado objectivo, ou seja, num ambiente com vários alvos, exemplificando o ambiente 5, estes algoritmos permanecem fixados à política comportamental aprendida para atingir sempre o mesmo alvo.

6.2.3 Implementação agente *Dyna-Q*

Na arquitectura *Dyna-Q*, o núcleo de aprendizagem permanece o mesmo em relação ao algoritmo *Q-Learning*. Contudo, um modelo interno é introduzido e modelado sob a forma de dois dicionários representativos das funções de transição T e reforço R . Nesse sentido, o passo de aprendizagem típico do algoritmo *Q-Learning*, é complementado com a actualização do modelo interno e posteriormente pela simulação interna de n passos de aprendizagem.

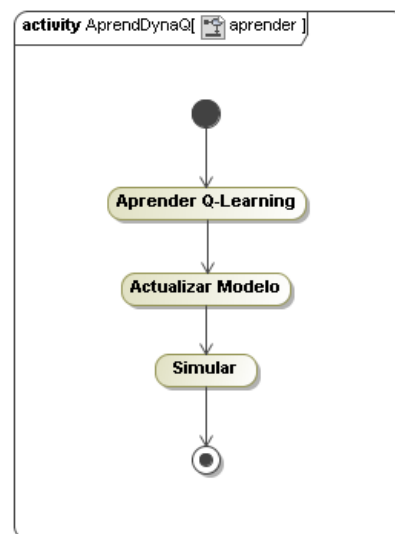


Figura 39 - Diagrama de actividade aprender *Dyna-Q*.

Na perspectiva de engenharia de *software*, o uso da herança permite reutilizar o mecanismo de aprendizagem do algoritmo *Q-Learning*, sendo apenas importante actualizar o mecanismo para contemplar os processos adicionais. Para representar estes processos foram criados respectivamente os métodos para actualizar o modelo interno e realizar a simulação.

No contexto da simulação interna, o módulo *random* da linguagem *python* é usado para escolher, de forma aleatória a cada passo n , uma chave da função de transição T representativa de um par estado-acção. Esse par é posteriormente utilizado para obter o próximo estado, em conjunto com o reforço associado, e realizar uma nova iteração de aprendizagem.

Relativamente ao desempenho do agente *Dyna-Q*, verifica-se que a sua aprendizagem é mais rápida devido às simulações. A propagação do valor, ocorre de forma consideravelmente mais rápida e em função do número de passos de simulação escolhido. A rapidez na convergência da função valor, sendo bastante rápida, também permite que este agente seja mais robusto a dinamismos no ambiente. A figura seguinte mostra exemplos de execução da função $Q(s,a)$ para os ambientes.

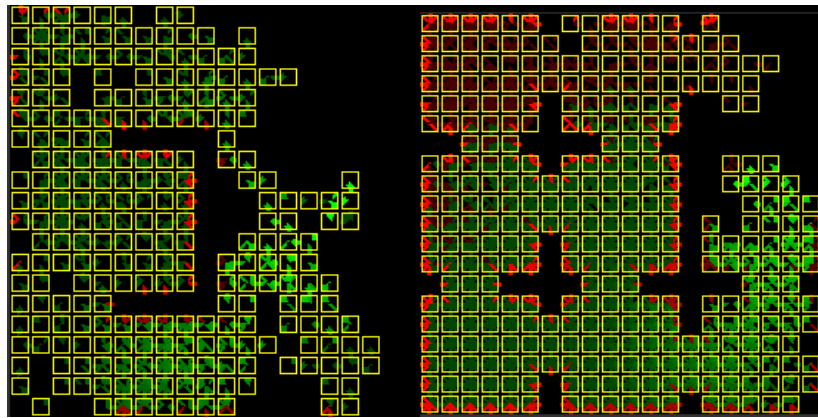


Figura 40 - Vista da acção/valor/direcção do agente Dyna-Q ($n=100$) para o ambiente 3 e 6 respectivamente.

Em relação a resultados operacionais nos diferentes tipos de ambiente, verifica-se que o sacrifício de recursos é compensado. Em ambientes mais complexos, como o ambiente 6 comparativamente ao 3, os reforços negativos são propagados mais rapidamente, fruto das simulações.

6.2.4 Implementação agente com memória episódica

A aprendizagem através de memória episódica consiste numa nova extensão da aprendizagem *Q-Learning*. Ao invés de ser considerado um modelo interno ao agente, a sua memória é apenas constituída pela função valor $Q(s,a)$ e uma sequência episódica de estados s , acções a , reforços r e novos estados s' .

Comparativamente ao algoritmo introduzido pela arquitectura *Dyna-Q*, a arquitectura é idêntica. A excepção consiste na forma como as experiências são armazenadas em memória, sendo assimiladas sequências completas (s,a,r,sn) equivalentes a um episódio experienciado pelo agente. Esta forma de assimilação resulta num método de simulação de experiências mais simples, onde a cada n passo simulado, um episódio é seleccionado para aprendizagem.

Em termos computacionais, esta solução consome mais recursos de memória. Porém, o facto de serem armazenados episódios permite um melhor controlo da aprendizagem, uma vez que, é possível criar posteriormente mecanismos para selecção de episódios mais relevantes para aprendizagem. Adicionalmente, este método pode ser aplicado num contexto mais geral devido a não ser necessário definir o modelo interno para cada problemática a solucionar.

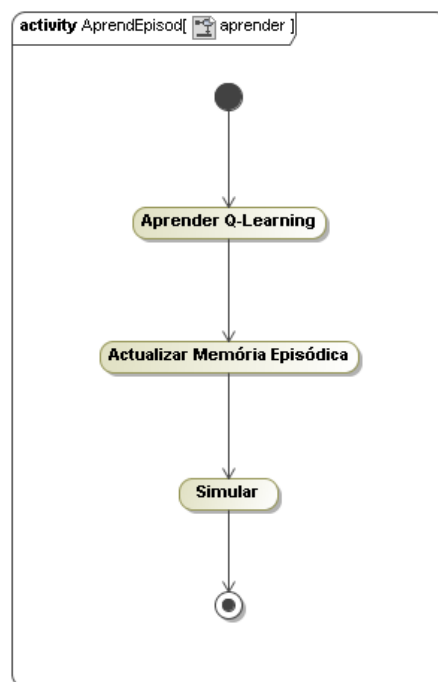


Figura 41 - Diagrama de actividade aprender Memória Episódica.

6.3 Aprendizagem com Utilização de Heurística

6.3.1 Integração de reactividade

A validação de resultados e consequente avaliação do desempenho dos algoritmos assenta na conformidade de utilização do mesmo ambiente em cada caso. Em termos de concretização, para validação do funcionamento dos diferentes algoritmos, não é necessário um ambiente demasiado complexo. Tal traduz-se na utilização de um ambiente de teste simples tipicamente representativo do problema dos óptimos locais.

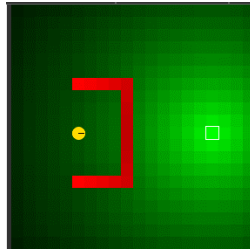


Figura 42 - Ambiente nº3 visualizado sob a forma do campo de potencial gerado. O único obstáculo presente caracteriza um óptimo local.

Uma arquitectura híbrida foi implementada por via da criação de um Controlo Híbrido, correspondente a um controlo do agente. No interior deste, os respectivos controlos das camadas reactivas e adaptativas são criados. Também, para modularizar as soluções de coordenadores propostos, uma interface, designada CoordAccao, foi criada para definir o método “seleccionar_acciao”, obrigatório para todos os futuros coordenadores.

No contexto da coordenação de acção que verifica existência de valor na função Q , foi necessário adicionar um método “existe_valor” às memórias de aprendizagem a intervirem. Este método, embora semelhante ao “obter”, foi necessário de forma a evitar o valor de omissão e retornar um booleano.

6.3.1.1 Controlo Reactivo - Biblioteca ECR

Para formalizar a concretização de uma camada reactiva, foi definida uma biblioteca de esquemas comportamentais reactivos (ECR). O objectivo desta estruturação consiste em fornecer uma abordagem que permita realizar a codificação de regras reactivas.

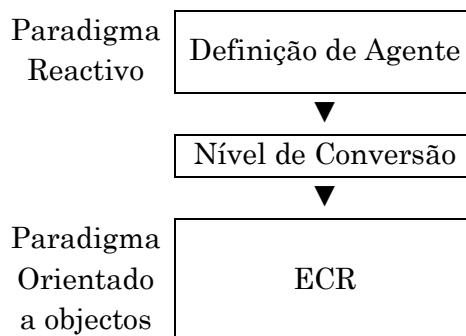


Figura 43 - Diagrama conceptual da tradução do paradigma reactivo para o paradigma orientado a objectos.

Um esquema comportamental consiste no processamento da percepção e na geração de uma resposta. Podendo a formalização de um esquema comportamental ser complexa, procedeu-se à sua modularização interna através da definição de uma *Reacção* e *Módulos Comportamentais*. Desta maneira, um controlo reactivo pode ser acoplado a um agente, processar a percepção recebida e, obter uma reacção em função do esquema comportamental atribuído.

No paradigma reactivo, uma reacção consiste na representação de uma regra estímulo-resposta. No entanto, é por vezes necessário seleccionar a resposta adequada consoante os diferentes estímulos activos. Uma reacção composta (módulo comportamental), permite modelar um comportamento nesse sentido e definir a forma como as respostas são seleccionadas em função do estímulo. Este tipo de mecanismo de reacção é proveitoso uma vez que, possibilita a execução paralela, com precedência ou com combinação de acções, em caso de activação simultânea de vários estímulos.

Através da implementação destes componentes e igualmente pela definição dos módulos comportamentais hierárquico e por prioridade, é possível estruturar internamente os comportamentos complexos pretendidos.

6.3.2 Método *Heuristic Sparse Learning* - HSL

A implementação deste novo método de aprendizagem proposto requer a definição de um novo controlo de aprendizagem por reforço, uma vez que, a diferença, em relação aos outros métodos, incide no controlo ao invés do mecanismo. Este algoritmo contempla assim diversos mecanismos de aprendizagem, fazendo a diferença através do controlo do passo de aprendizagem e do mecanismo de selecção de acção. O processamento

realizado por o controlo *HSL* é apresentado na Figura 44 - Diagrama de actividade processar do controlo *HSL*..

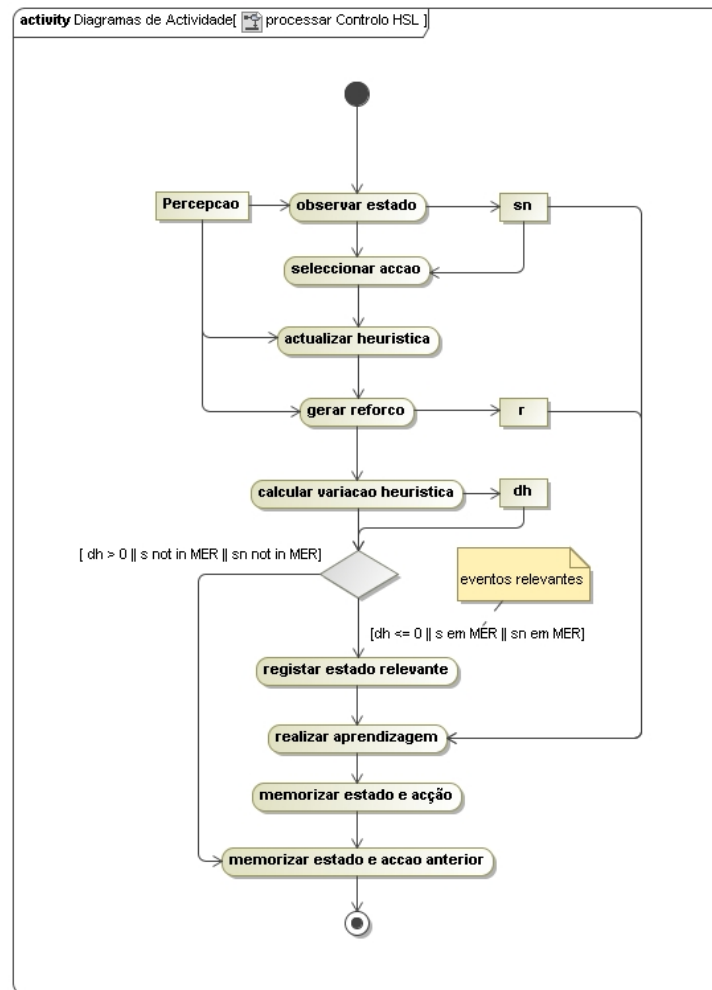


Figura 44 - Diagrama de actividade processar do controlo *HSL*.

O mecanismo de selecção de acção traduz a política comportamental do *HSL*. Se um par estado-acção já estiver na memória de aprendizagem, a política ϵ -greedy é seguida, caso contrário a acção que maximiza a heurística considerada para o estado actual é escolhida. Dada a necessidade funcional da política ϵ -greedy, o mecanismo de selecção de acção *HSL* deriva do próprio mecanismo ϵ -greedy.

6.3.3 Arquitectura híbrida com camada reactiva com memória

Para a implementação desta arquitectura foi necessário atentar à concretização do controlo híbrido, que agrega as camadas intervenientes, e a camada reactiva com memória, que conterá a memória de estados relevantes e a máquina de estados comutadora de acção. O diagrama de actividade em seguida descreve o processamento do controlo híbrido de acordo com a organização horizontal.

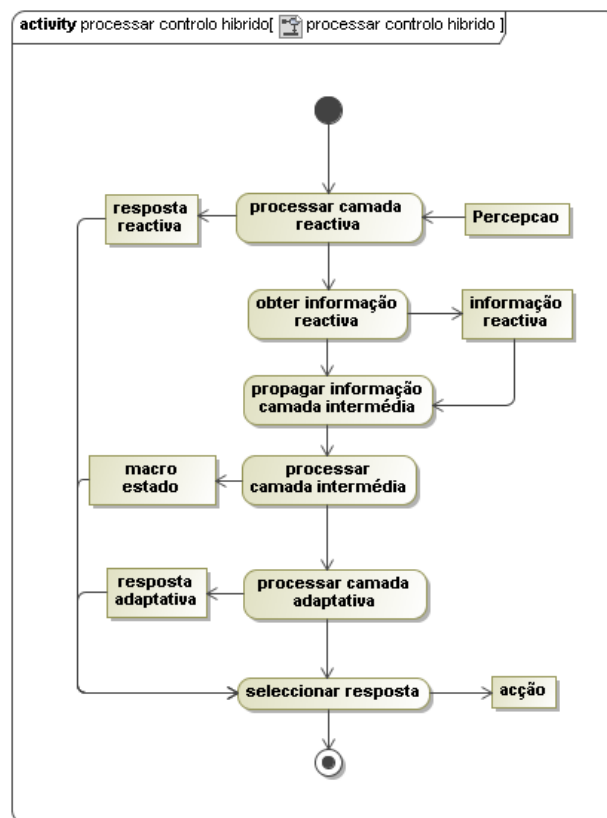


Figura 45 - Diagrama de actividade do método processar do controlo híbrido.

De maneira semelhante, a implementação do controlo reactivo com memória, ou intermédio, terá um processamento associado que permita aplicar a heurística de reconhecimento, concretizada pela máquina de estados comutadora de acção.

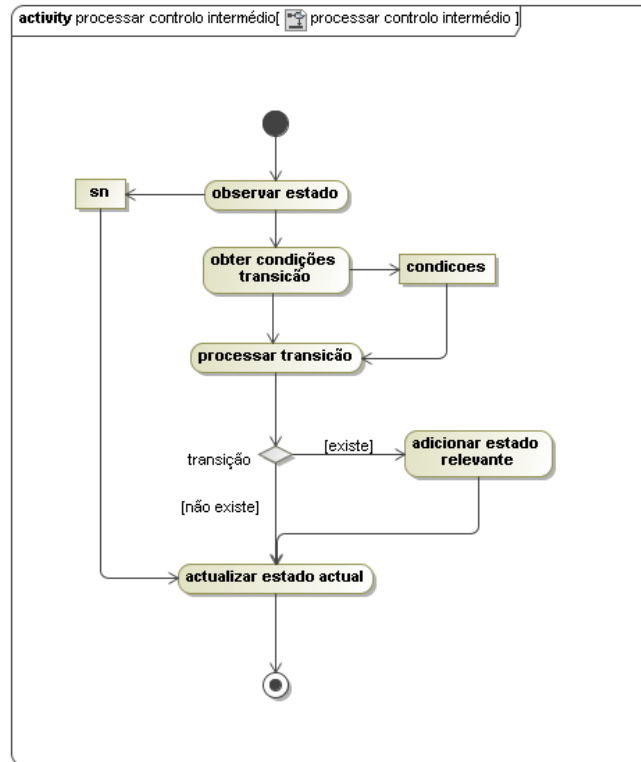


Figura 46 - Diagrama de actividade do método processar do controlo intermédio, ou reactivo com memória.

6.4 Aprendizagem por Abstracção de Estado

Nesta secção são concretizados os algoritmos de aprendizagem que recorrem a diferentes técnicas para obter uma abstracção de estado. A abordagem recorrendo a redes neuronais é realizada através do agente *Deep-Q* adaptado ao contexto dos ambientes da PSA. Posteriormente são descritos os procedimentos das abordagens de memória hierárquica com estrutura *Quadtree* e da memória selectiva de instâncias.

6.4.1 Agente adaptativo *Deep-Q* adaptado

Informação que permita saber qual a estrutura correcta para a rede neuronal do agente nesta problemática é inexistente dado o contexto de utilização original da arquitectura *Deep-Q*. Por esse motivo, é importante investigar primeiramente a viabilidade de uma rede neuronal cujas entradas

consistem na posição do agente e as saídas equivalentes a cada valor para cada acção da função $Q(s,a)$.

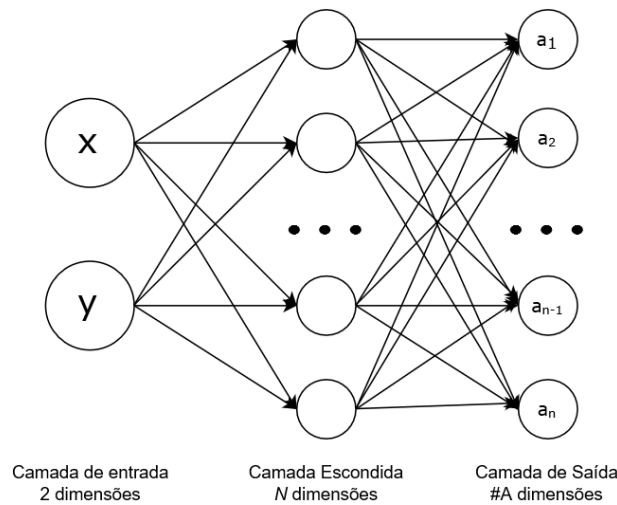


Figura 47 - Esquema da estrutura da rede neuronal.

O sucesso do processo de treino por parte da rede consiste igualmente pela forma como a sua estrutura é definida. Para testar a viabilidade e treinar uma política de acção é importante considerar os diferentes parâmetros da rede e o próprio processo de treino.

Para uma representação de estado posicional, baseado em coordenadas, a camada de entrada da rede detém duas dimensões. A camada escondida possui um número variável de neurónios de que podem alternar consoante o ambiente e de forma a estudar o desempenho da própria rede. Por fim, a camada de saída terá o número de neurónios de saída equivalente à cardinalidade de acções existentes ou consideradas. Com esta estrutura, a rede neuronal actua como a componente de memória de aprendizagem, onde, para um estado (x,y) serão obtidos valores da função valor $Q(s,a)$ em cada nó correspondente a determinada acção.

Assim, a implementação de um teste *offline* consiste na obtenção de uma política comportamental já aprendida, sob a forma de uma função valor, e utilizar a mesma para treinar uma rede neuronal.

Esta abordagem cuidada possibilita uma avaliação da viabilidade de uma rede neuronal aprender em situações deste tipo, e também, afinar os parâmetros envolvidos antes da rede ser integrada no agente. Se todos os estados gravados obtiverem uma saída da rede com política semelhante à previamente aprendida, então fica provado a capacidade de aprendizagem da rede.

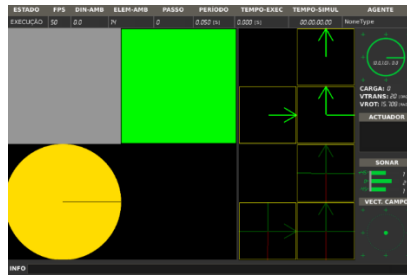


Figura 48 - Visualizadores da plataforma PSA com a políticas original aprendida (visualizador superior) e a treinada (visualizador inferior).

O sucesso do treino *offline* da rede fornece bons indicadores quanto à capacidade de aprendizagem. Contudo, com a integração da rede na arquitectura do agente a aprendizagem será feita num contexto *online* de operação, onde é importante garantir que o processo de treino da rede seja realizado dentro do tempo útil da iteração de aprendizagem.

O objectivo comum à investigação de métodos em escassez de recursos prende-se com a obtenção de um agente expedito na aprendizagem e convergência. Considerando a designação original de um episódio, que consiste em todos os passos experienciados, o agente necessita de alcançar o estado objectivo para completar e adquirir um episódio de aprendizagem. Esta abordagem, utilizada originalmente [11], desvia-se da visão pretendida de um agente que aprenda a cada passo de execução e não a cada episódio.

Assim, a tentativa de aprendizagem designa um episódio como sendo a experiência adquirida a cada passo. Nesta vertente, o agente tentará desde o primeiro instante aprender seleccionando um conjunto de experiências para treinar a cada passo.

Esta solução adquire alguns riscos na medida em que existe incerteza quanto à existência de informação presente nas experiências ser suficiente para aprender ao invés de diversos episódios compostos por sequências de experiências.

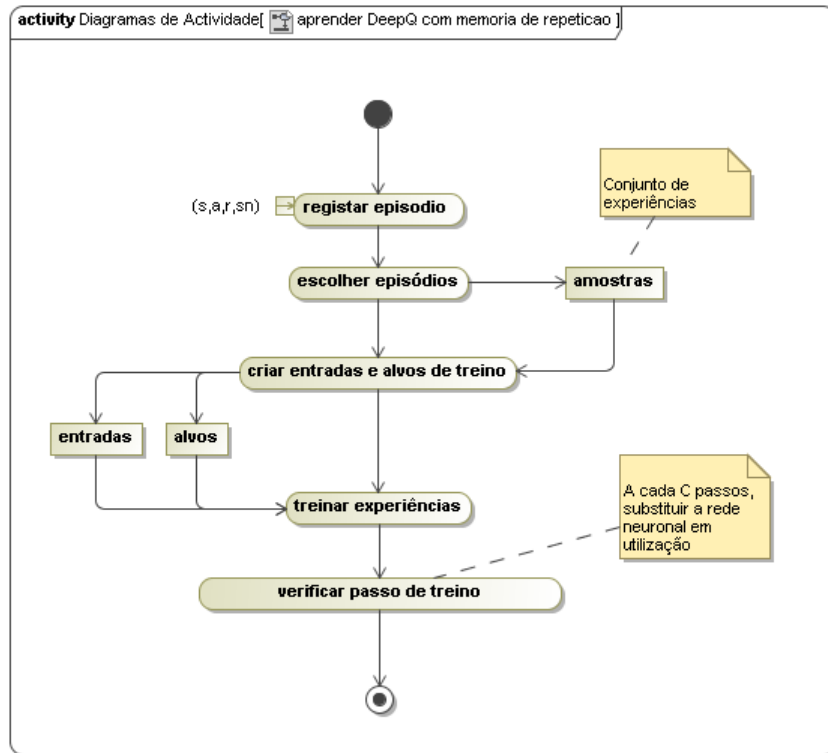


Figura 49 - Diagrama de actividade aprender do agente *Deep-Q*.

Através do diagrama de actividade da Figura 49 - Diagrama de actividade aprender do agente *Deep-Q*, verifica-se que não existe muitas diferenças relativamente ao algoritmo *Deep-Q* presente na Figura 20. Um episódio, ou também designada experiência, é registado na memória de repetição. Um conjunto de experiências são escolhidas aleatoriamente para criar o conjunto de treino para treinar no presente passo de execução. As entradas (os estados), e alvos são criados, estes últimos através da já conhecida equação:

$$y_i = \begin{cases} r_j & , \quad \text{se episódio termina no passo } j + 1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-), & \text{caso contrário} \end{cases}$$

Posteriormente, todas as amostras utilizadas num número pré-definido de iteração de treino. As etapas presentes no processo de aprendizagem são bastante semelhantes ao método *Dyna-Q*. Esta etapa é semelhante devido às iterações de treino na rede, que corresponderia às simulações internas realizadas pelo algoritmo *Dyna-Q*.

6.4.2 Agente adaptativo hierárquico *quadtree*

Ao realizar uma representação de estado mais eficiente em termos da redução espacial, são criadas camadas de abstracção adicionais. Os macros estados presentes nestas camadas são úteis pois agrupam informação de vários estados. Quando existe um estado mais detalhado, ou seja, de menor nível abstracto, a informação presente nesse(s) estado(s) é mais relevante e deve ser considerada em detrimento de informação mais abstracta. Devido à gestão da informação presente nos diferentes níveis é importante distinguir quando existiu condições de verdadeira aprendizagem.

O controlo de aprendizagem, sendo responsável por efectuar o passo de propagação do valor, necessita de explicitar quando deve efectivamente aplicar o processo de aprendizagem. Um agente que transite de um estado para outro, mas que se mantenha no mesmo macro estado sem acontecimento relevante, não acrescenta nenhuma informação útil para aprendizagem. Desta forma, o processo para aprender prende-se com duas condições que necessitam de ser verificadas anteriormente.

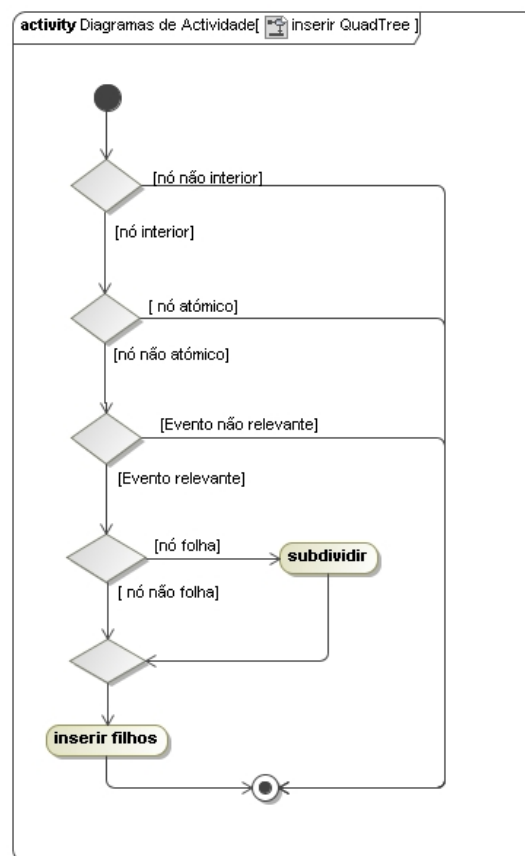


Figura 50 - Diagrama de actividade do processo de criação da estrutura em árvore *Quadtree*.

A primeira condição prende-se com o reforço necessitar de ser relevante, ou seja, de existir uma acção que retornou uma recompensa pertinente, o que no caso da estrutura *Quadtree*, poderá ter levado à subdivisão de um macro estado. A segunda condição advém da verificação de transição entre estados (abstractos ou não), i.e., se o novo estado para o qual o agente transitou é diferente do anterior.

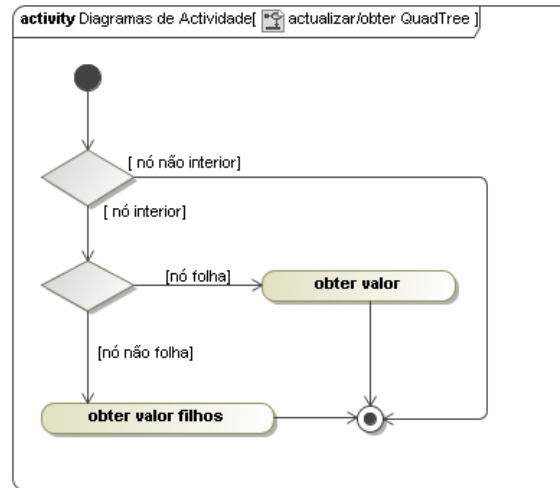


Figura 51 - Diagrama de actividade da recuperação de informação da estrutura em árvore *Quadtree*.

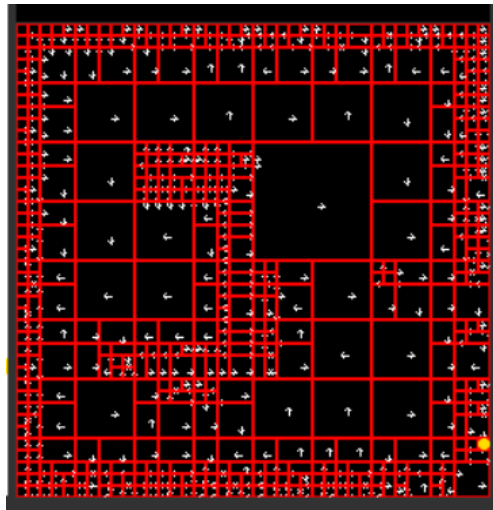


Figura 52 - Visualização da memória de aprendizagem utilizando a estrutura *Quadtree*. A política de acção é visível através das setas brancas. Ambiente 3 com grau 2 de detalhe.

6.4.3 Agente adaptativo com memória selectiva de instâncias

A implementação de um agente adaptativo com memória selectiva de instâncias é bastante transparente, uma vez que apenas é necessário substituir a memória de aprendizagem, mantendo-se a restante estrutura.

Após definir os mecanismos e parâmetros de aprendizagem, verifica-se que esta solução obtém uma estrutura interessante relativamente à memória formada e que pode ser visualizada nas figuras seguintes.



Figura 53 - Agente adaptativo com memória selectiva de instâncias em simulação no ambiente 3 com detalhe 1.



Figura 54 - Agente adaptativo com memória selectiva de instâncias em simulação no ambiente 3 com detalhe 2.

6.5 Aprendizagem com Recursos Muito Limitados

6.5.1 Agente adaptativo com representação limitada de estado

Adaptar o agente para aferir uma representação de estado diferente de uma posição, consta em alterações triviais no controlo de aprendizagem. Encontrando-se todo o processo modularizado, a operação de observar o estado do agente apresenta-se concretizada sobre um método próprio. Desta forma e numa primeira iteração, especificou-se um controlo de aprendizagem, denominado sensorial, através de herança para reescrever o método de observação de estado.

Para analisar diferentes observações de estado possíveis, fez sentido definir, analogamente ao ocorrido com a função de reforço, o conceito de Observador de Estado como uma interface. Esta refactorização posterior, possibilita assim a especificação prévia da representação pretendida.

As ferramentas de visualização possibilitam obter informações relativas ao agente e ao seu progresso. A informação transmitida constitui um bem valioso para avaliação e melhoramento. Para uma representação bidimensional da posição do agente, a capacidade humana de raciocínio espacial é particularmente útil no sentido de acompanhar a evolução do espaço de estados sobre o ambiente. Numa situação cujo acompanhamento deixe de ser possível é necessário encontrar formas de visualizar algum tipo de informação.

O objectivo de uma representação de estado mais simples, recorrendo à informação sensorial dos sonares, visa a extrapolação da aprendizagem para zonas que representem situações, diga-se estado, idênticas para o agente. No entanto, o acompanhamento da evolução no espaço de estados pode não fornecer a informação útil ou relevante.

Assumindo este problema, uma pergunta poderá ajudar à análise do agente: “Considerando o estado do agente, quais as situações são semelhantes?”. Encarando as situações semelhantes como posições no ambiente, é possível atribuir a um conjunto de locais um estado “sensorial” representativo dos mesmos. Esta visualização consegue assim realizar a correspondência visual entre posições no ambiente cujo estado sensorial é igual.

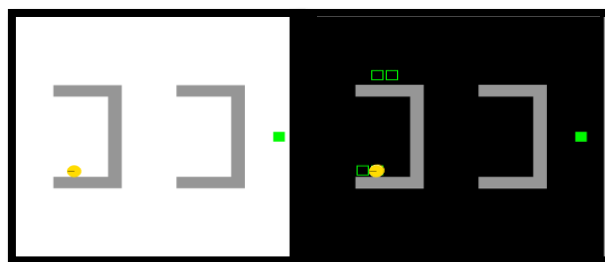


Figura 55 - Visualizador de estado semelhante. Nos quadrados verdes é possível observar quais os estados posição correspondem ao mesmo estado sensorial.

6.5.2 Vertente robótica

Toda a vertente robótica foi desenvolvida com base na ideia de um agente robótico simples e os componentes necessários nesse sentido. Encontrou-se uma solução de robô assente numa estrutura bimotora recorrendo a uma roda traseira de suporte. Os sensores, módulos e controlador dos motores foram escolhidos de forma a serem compatíveis com a unidade de processamento existente e em função do custo e suas capacidades.

Componente	Quantidade	Descrição
Estrutura Acrílica e Rodas	1	Base de suporte para os restantes componentes.
Roller Micro Switch	1	Micro Switch utilizado para detectar informação de colisão frontal.
Módulo ultra-sons HC-SR04	3	Módulos para detectar distância a obstáculos através de ultra-sons.
Sensor Luminoso TSL2561	2	Sensor Luminoso para medir a intensidade luminosa.
Motores DC	2	Motores de corrente continua com baixa potência para mover as rodas.
Controlador Motores (Motor Driver) TB6612FNG	1	Controla a activação dos motores através de sinalização.
Raspberry Pi 2 Model B	1	Unidade de Processamento responsável por realizar leituras dos sensores, processamento interno e actuação dos motores.

PowerBank (4000/2000 mA)	2	Fornecimento de energia para a unidade de processamento e controlador dos motores.
-----------------------------	---	--

Tabela 2 - Componentes integrantes do agente robótico.

Antes de proceder à montagem do robô atentou-se a uma abordagem mais cautelosa de todos os componentes, principalmente os sensores e módulos, recorrendo a testes unitários. Desta forma é possível adquirir conhecimento técnico sobre o funcionamento das partes e validar respostas.

6.5.2.1 Plataforma Robótica de Agente

A PRA segue as mesmas directrizes e forma de utilização da PSA. Nesse sentido, o ciclo de execução da plataforma contempla o uso de comandos para ser possível pausar, realizar um passo ou terminar a execução. O agente é executado no caso de não existir pausa ou existir um passo de execução a ser dado quando em pausa.

Sendo o processamento realizado exclusivamente por parte do próprio agente na unidade de processamento, a visualização de informação não é trivial quando comparada à PSA. A concepção de dois modos de funcionamento para a PRA permite colmatar este problema: um modo local e um modo remoto.

O modo local, contém uma opção “*verbose*” cuja activação permite visualizar informações do agente concreto na linha de comandos. Isto é possível através da adição do método “*mostrar_cli*” à interface do agente robótico. O utilizador da plataforma é assim livre de mostrar o que pretender e em função da sua implementação de agente.

O modo remoto, embora não concluído, consiste na existência de uma interface gráfica remota para visualização, em tempo de execução, das mais variadas informações e estado do agente. Para garantir a liberdade de acção do agente, o canal de comunicação necessita de ser sem fios, existindo duas tecnologias aptas para o fazer: *Bluetooth* e *Wifi (802.11)*. Para o propósito experimental, transmissão de dados via *Bluetooth* parece suficiente, no entanto o *Wifi* possibilita maior débito e maior alcance, o que poderá ser útil.

6.5.2.2 Agente Robótico

Mais uma vez para manter total compatibilidade com o ambiente de simulação, o agente robótico implementado contém um sensor múltiplo e um actuador. De igual forma, este sensor retornará uma percepção com a informação recolhida por os sensores. Quer a interface *Sensor*, quer a *Actuador* foram mantidas na integra para permitir a modularização e escalabilidade do agente. Nesse sentido, é possível implementar novos sensores e actuadores ou especificar apenas sensores pretendidos para o sensor múltiplo.

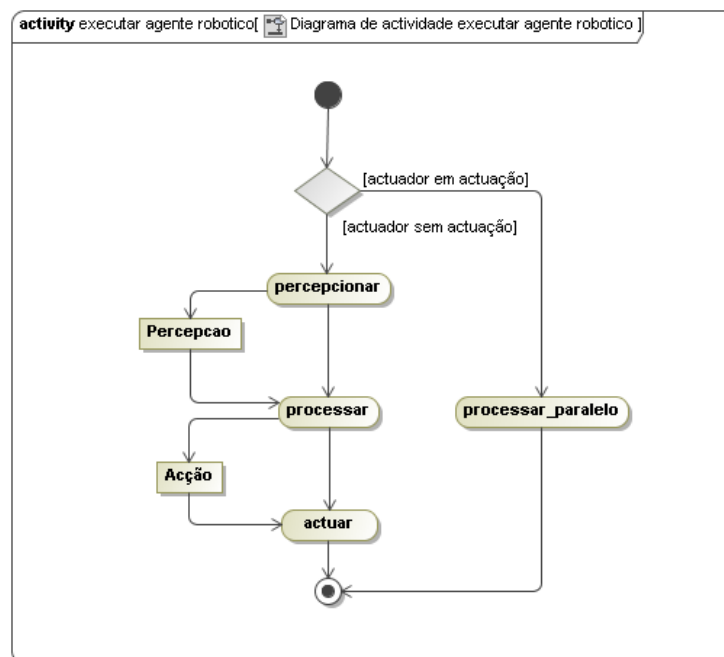


Figura 56 - Diagrama de actividade *executar* do agente robótico.

O agente robótico possui uma particularidade não existente na via de simulação. A realização de acções por parte dos actuadores requer um determinado período de tempo para serem concluídas. Nesse sentido, a perspectiva de execução contínua pode ser imitada através do bloqueio do processamento aquando da realização de acções.

Porém, tempo de execução útil é desperdiçado ao realizar estas interrupções. A solução simples passa por realizar um processar paralelo à actuação dos actuadores, de forma a aproveitar esse período de processamento de acordo com o agente em causa.

Para o robô desenvolvido foram concretizados os sensores: sonar, para os módulos de ultra-som, sensor de contacto, para o *micro switch*, e sensor

luminoso, para os sensores TSL2561. Para os motores, foi implementado o actuador de movimento com as acções anteriormente referidas na Tabela 12.

6.5.2.3 Agente Robótico Reactivo

Primeiramente à criação de uma abordagem robótica adaptativa recorreu-se à implementação de um agente robótico reactivo. Esta primeira iteração traduz-se na vantagem de testar os componentes e a implementação de agente e PRA realizada. As alterações para compatibilizar a biblioteca ECR com o agente robótico foram consequentemente mínimas, uma vez que, apenas existiu a necessidade de redefinir o esquema comportamental hierárquico e alterar as reacções de seguimento de potencial.

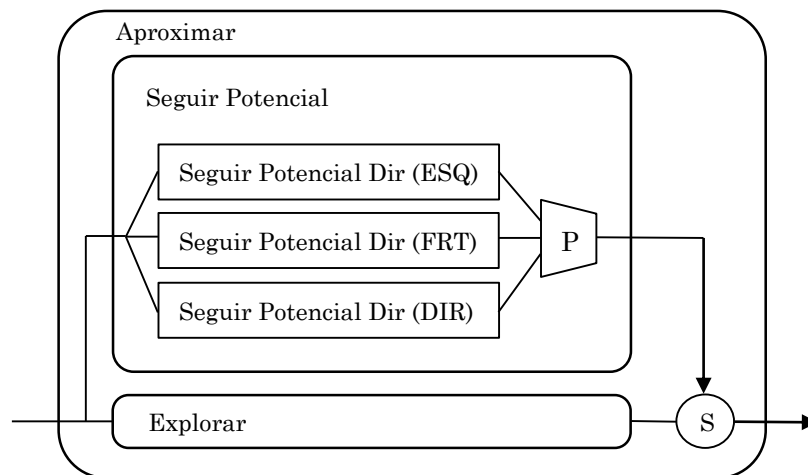


Figura 57 - Esquema reactivo do módulo comportamental do agente robótico.

A existência de apenas dois sensores físicos de luminosidade, traduz-se num problema quando da criação de reacções, uma vez que, o agente necessita eventualmente de seguir em frente quando o potencial obtido nos dois sensores luminosos não for diferente o suficiente. A solução encontrada para que o estímulo, e posterior resposta, na direcção frontal fosse activada baseia-se no cálculo da variação de potencial e posterior normalização pelo valor médio.

$$\begin{aligned}
P_{\text{médio}} &= \frac{P_{\text{esquerda}} + P_{\text{direita}}}{2} \\
\Delta P &= P_{\text{esquerda}} - P_{\text{direita}} \\
\Delta P_{\text{normalizado}} &= \frac{\Delta P}{P_{\text{médio}}}
\end{aligned}$$

Considerando um limiar ε , a prioridade da resposta frontal em relação às outras direcções será dada por:

$$\text{Prioridade} = \begin{cases} 100, & \text{se } |\Delta P_{\text{normalizado}}| < \varepsilon \\ 0, & \text{caso contrário} \end{cases}$$

A adição de outras reacções, como “evitar obstáculo” foi posteriormente realizada, uma vez que, relativamente aos sensores de distância, os mesmos existem e correspondem directamente às direcções.

6.5.2.4 Agente Robótico Adaptativo

A implementação de um agente adaptativo fica facilitada devido à modularização prévia da biblioteca da aprendizagem por reforço. Para transpor o mesmo para a vertente robótica, é necessário identificar os componentes que requerem definição específica: a função de reforço e a representação de estado. A função de reforço pode ser formulada da seguinte maneira:

$$R = \begin{cases} -1, & \text{custo movimento (por defeito)} \\ -R_{\text{max}}, & \text{caso "colisão" ou } \text{distancia}_{\text{percepcao}[FRT]} \leq \lambda_{\text{dist}} \end{cases}$$

Numa primeira consideração, é importante determinar em que contexto o agente receberá um reforço negativo. No caso de ocorrer uma colisão esse *feedback* é dado pelo sensor de contacto. Contudo, caso a colisão efectiva não seja o pretendido, é possível definir uma colisão como, o agente encontrar-se a uma distância inferior de um limiar desejado, quando o sensor de distância retorna um possível obstáculo. Importante notar que, para o último caso, um limiar elevado leva a que o agente evite contacto e também poderá não explorar as áreas adjacentes a esses obstáculos, independente das mesmas serem relevantes ou não.

A definição de um reforço máximo poderia ser eventualmente estudada através da avaliação das leituras dos sensores de potencial numa futura iteração. No entanto, devido aos problemas já assinalados da ambiguidade de

estado e tempo extenso de aprendizagem, o agente adaptativo terá apenas como objectivo inicial a evitar obstáculos através da aprendizagem de padrões reactivos.

A representação de estado agrega a informação dos sensores de distância à semelhança do estudo realizado sobre a limitação da representação de estado.

$$s_n = (distancia_{[FRT]}, distancia_{[ESQ]}, distancia_{[DIR]})$$

6.6 Conclusões

Este capítulo apresentou a concretização das abordagens apresentadas no capítulo 5. Foi concebida uma arquitectura de agente geral de forma a comportar as diferentes arquitecturas. Da mesma forma, as bibliotecas ECR e AprendRef foram concebidas para concretizar respectivamente, as componentes reactiva e adaptativa necessárias.

A modularização pensava permite usar de forma transparente as bibliotecas na definição de controlos para a integração de reactividade. De maneira semelhante, os métodos com abstracção de estado usufruem da modularização no sentido do desenvolvimento se ter apenas focado sobre a memória de aprendizagem, mantendo inalterado a restante estrutura e escolha de mecanismo de aprendizagem.

No contexto dos recursos muito limitados foi útil a refactorização da representação de estado para contemplar qualquer configuração de estado pretendida. Relativamente à vertente robótica, a PRA foi pensada para replicar o uso e funcionamento da PSA. Os detalhes e descrição da montagem técnica da plataforma física encontra-se presente nos anexos.

7 Resultados Experimentais

As abordagens e arquitectura proposta apresentadas nos capítulos anteriores são fundamentadas através da concretização experimental realizada e apresentada no presente capítulo.

Primeiramente são introduzidos o contexto e a metodologia criada para obtenção de resultados. Os ambientes relevantes, concebidos para demonstrar determinadas características dos agentes, também são caracterizados.

Secundariamente, os resultados obtidos são apresentados e sujeitos a comparações e análise. Por fim, algumas considerações são também feitas fruto dos comportamentos evidenciados nas diferentes abordagens.

7.1 Rotinas de Teste

Para avaliar o desempenho das soluções propostas, e comparar as mesmas com os métodos base e já consolidados, é necessário obter resultados coerentes e em carácter de igualdade.

A plataforma PSA, por si só, já reúne diversas informações e estatísticas à execução do agente. Nesse sentido, uma nova versão da PSA foi fornecida com vista a disponibilizar essas mesmas informações, definir um modo expedito de simulação sem visualização gráfica e, ainda, permitir definir o número de episódios a simular.

Com as informações, derivadas da simulação dos diferentes agentes, um módulo estatístico independente foi implementado com o intuito de ler os dados obtidos e disponibilizar gráficos estatísticos.

Encontrando-se o tempo de execução comprometido em função da máquina que corre a simulação, foram escolhidas unidades de passo de execução como medida para avaliação, devido ao seu carácter independente.

Para avaliar a solução, foi equacionada uma métrica do total de passos, assim como o número de passos a cada simulação. Considerando uma simulação com N episódios e registando a evolução do número de passos até atingir o objectivo. Esta medida permite avaliar e comparar a velocidade e tempo de convergência dos diferentes algoritmos.

7.2 Caracterização do Ambiente

O ambiente consiste em todo o meio fora do controlo directo do agente inteligente. Esta noção concretiza-se normalmente sob a forma do espaço físico ou virtual no qual o agente actua. Nesse sentido, em função do ambiente o agente poderá actuar de forma distinta consoante as características expostas e problemas consequentes.

No contexto da aprendizagem por reforço, um ambiente mais complexo, i.e., com um maior número de óptimos locais não acresce vantagens na obtenção de resultados comparativamente a um ambiente simplificado com apenas um óptimo local. O uso de um ambiente mais simplificado facilita e, até melhora, a execução de testes pois permite um maior número de execuções e amostras de dados.

Dos ambientes concebidos, e já existentes da PSA, o ambiente número 3 é o ambiente que satisfaz os requisitos de conter um óptimo local e apresentar uma estrutura simples na sua configuração, mas de dificuldade significativa na perspectiva da aprendizagem.

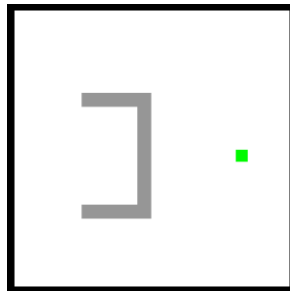


Figura 58 - Ambiente 3 da PSA.

Para testar e observar o comportamento dos agentes com representação sensorial foi concebido o ambiente 3b. Este ambiente duplica o obstáculo com o óptimo local do ambiente 3.

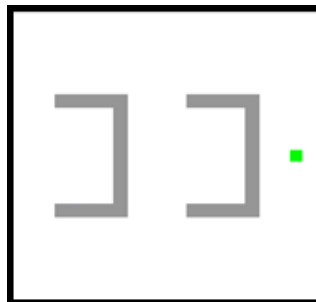


Figura 59 - Ambiente 3b da PSA.

Uma vez que, a representação de estado sensorial não detém uma representação posicional absoluta e única no espaço de estados, o mesmo ambiente serve para testar a extrapolação de aprendizagem para situações semelhantes em locais distintos. Estes ambientes servem assim os propósitos para testar os agentes.

7.3 Resultados das rotinas

Nesta secção são reunidos os resultados obtidos com base nas rotinas de teste descritas. As rotinas criadas não são referentes a todas as abordagens tomadas, tendo sido seleccionadas as abordagens mais relevantes para comparação.

7.3.1 Integração de Reactividade – Rotina 1

Para avaliar as vantagens da integração de reactividade em agentes adaptativos procedeu-se a uma rotina comparativa de métodos integrados, usando um coordenador de acção “*Existe Q*”, com os métodos gerais de aprendizagem por reforço.

Parâmetros		Valor
Aprendizagem	Reforço máximo	100
	Número de acções	8
	ϵ	0.05
	α	0.5
	γ	0.95
Simulação	Ambiente	3
	Detalhe	1
	Numero de Episódios	100

Tabela 3 - Parametrização dos agentes adaptativos para os métodos base e integração reactiva na rotina 1.

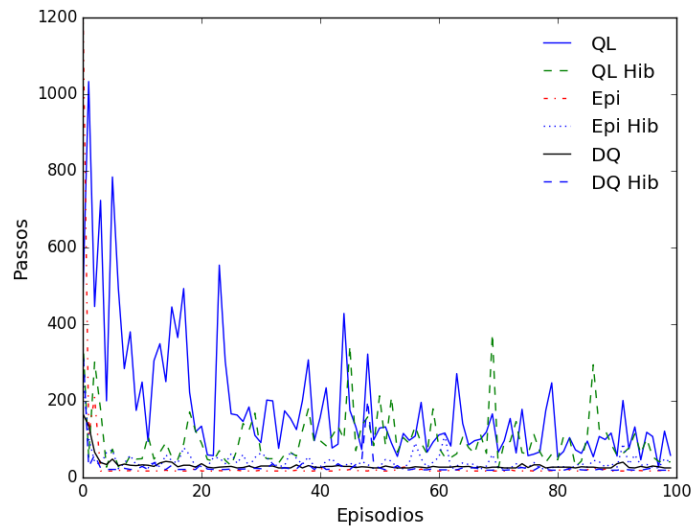


Figura 60 - Evolução dos métodos base em relação à integração de reactividade em simulação do ambiente 3 com grau de detalhe 1.

Atentando o gráfico da evolução na Figura 60, verifica-se que o método *Memória episódica* com integração destaca-se em relação ao mesmo método sem integração, com um desempenho pior. Uma explicação possível prende-se com a camada reactiva orientar o agente para estados em linha com o seguimento de potencial, o que resulta num início de propagação mais lenta devido apenas à exploração desses estados e provocando a escolha de episódios pouco diversificados quando comparado ao método sem integração.

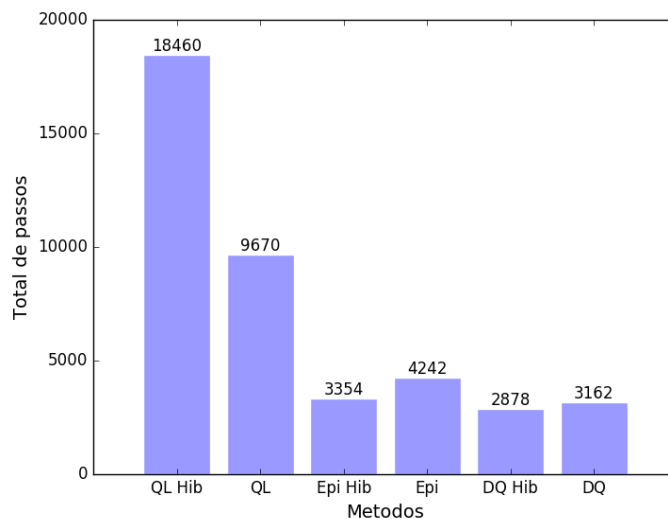


Figura 61 – Contabilização do total de passos dos métodos base em relação à integração de reactividade em simulação do ambiente 3 com grau de detalhe 1.

Por sua vez, o gráfico da Figura 61, realça a melhoria significativa obtida dos vários métodos, à excepção do método *Q-Learning*. Por conta da camada reactiva, a exploração dos estados é orientada no sentido do óptimo local para o ambiente 3 e, por consequente, a aprendizagem para evitar o obstáculo é mais rápida devido às simulações internas.

7.3.2 Integração de Reactividade – Rotina 2

A segunda rotina sobre a integração de reactividade prende-se com o estudo e desempenho quando o ambiente aumenta de escala, o que se traduz por um aumento do detalhe na simulação.

Parâmetros		Valor
Aprendizagem	Reforço máximo	100
	Número de acções	8
	ϵ	0.05
	α	0.5
	γ	0.95
Simulação	Ambiente	3
	Detalhe	2
	Numero de Episódios	100

Tabela 4 - Parametrização dos agentes adaptativos para os métodos base e integração reactiva na rotina 2.

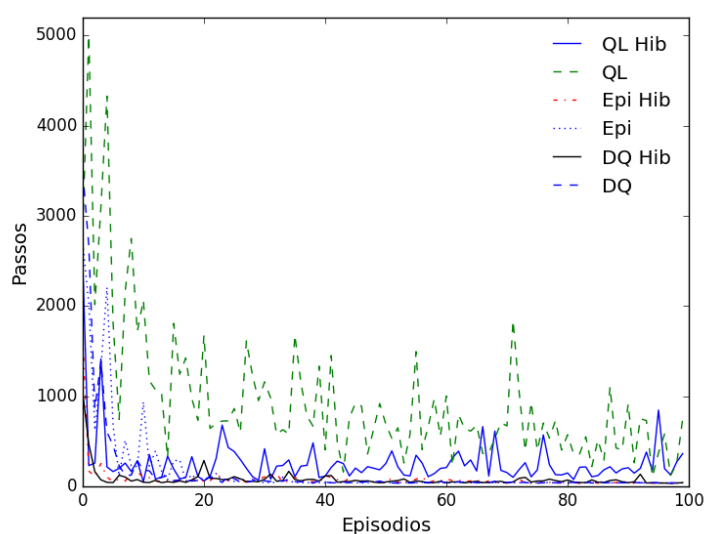


Figura 62 - Evolução dos métodos base em relação à integração de reactividade em simulação do ambiente 3 com grau de detalhe 2.

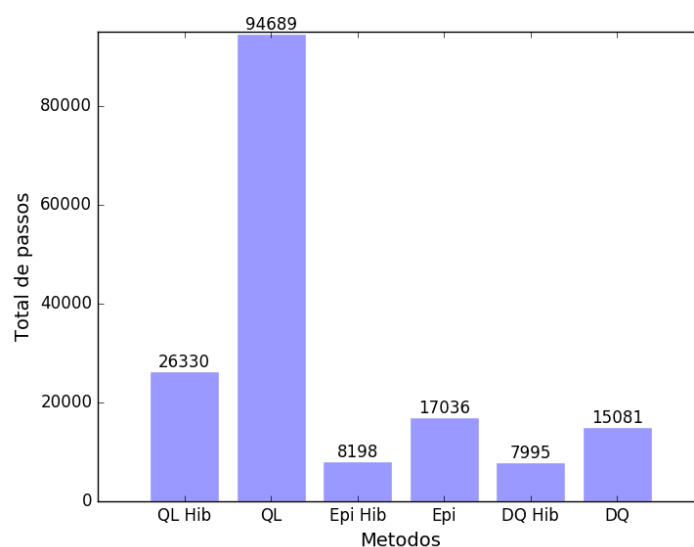


Figura 63 - Contabilização do total de passos dos métodos base em relação à integração de reactividade em simulação do ambiente 3 com grau de detalhe 2.

Observando os gráficos anteriores é visível que, o desempenho obtido pelas soluções com integração de reactividade é ainda mais expressivo quando comparado aos métodos base.

7.3.3 Aprendizagem hierárquica Quadtree – Rotina 1

Parâmetros		Valor
Aprendizagem	Reforço máximo	100
	Número de acções	4
	ϵ	0.05
	α	0.5
	γ	0.95
Simulação	Ambiente	3
	Detalhe	1
	Numero de Episódios	80

Tabela 5 - Parametrização dos agentes adaptativos para os métodos base e hierárquico *quadtree* na rotina 1.

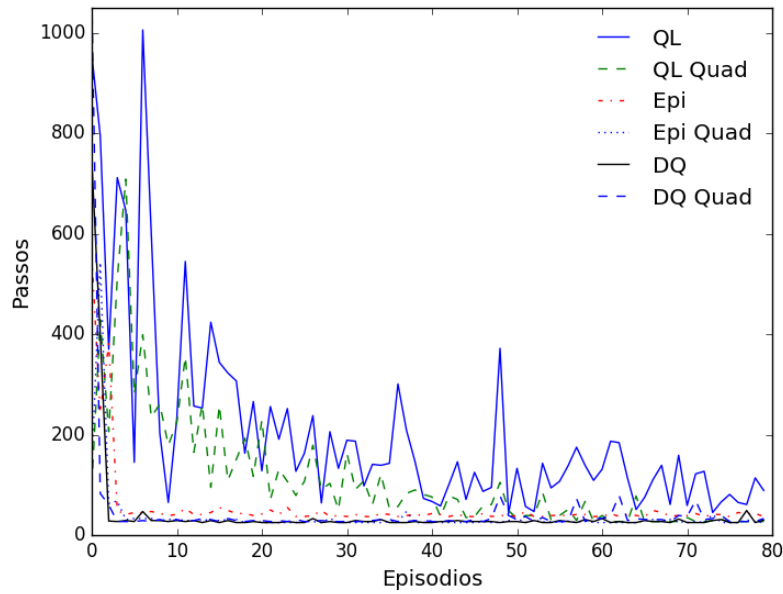


Figura 64 - Evolução dos algoritmos base e com memória *Quadtree* no ambiente 3 com grau de detalhe 1.

Observando a Figura 64 - Evolução dos algoritmos base e com memória *Quadtree* no ambiente 3 com grau de detalhe 1., é notável a evolução e convergência dos algoritmos com memória hierárquica. Esses mesmos algoritmos estabilizam para uma solução de forma expedita, surgindo apenas pequenos ressaltos no gráfico resultantes do diminuto factor de exploração da política ϵ -greedy, à semelhança dos métodos base. Também, o agente *Q-Learning* com *Quadtree* converge mais rápido que o mesmo sem esta estrutura.

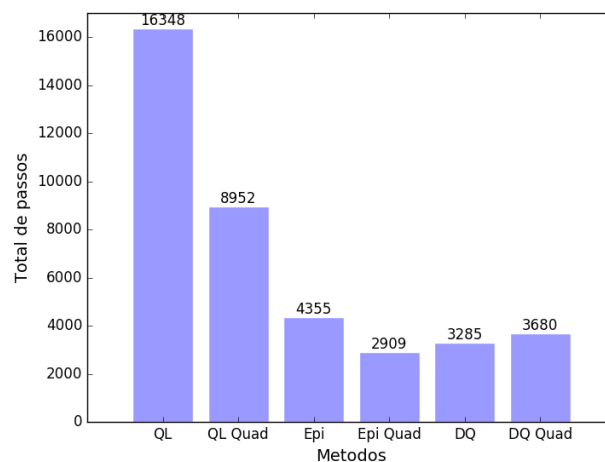


Figura 65 - Contabilização do total de passos de cada algoritmo na simulação do ambiente 3 com grau de detalhe 1.

Visando o gráfico de barras da Figura 65 - Contabilização do total de passos de cada algoritmo na simulação do ambiente 3 com grau de detalhe 1. podemos, igualmente, constatar que o número total de passos dos agentes é consideravelmente menor em relação aos mesmos agentes base. Esta diferença é mais notória no algoritmo *Q-Learning*, traduzindo-se numa redução de quase metade dos passos.

7.3.4 Aprendizagem Hierárquica Quadtree – Rotina 2

Parâmetros		Valor
Aprendizagem	Reforço máximo	100
	Número de acções	4
	ϵ	0.05
	α	0.5
	γ	0.95
Simulação	Ambiente	3
	Detalhe	2
	Numero de Episódios	160

Tabela 6 - Parametrização dos agentes adaptativos para os métodos base e hierárquico *quadtree* na rotina 2.

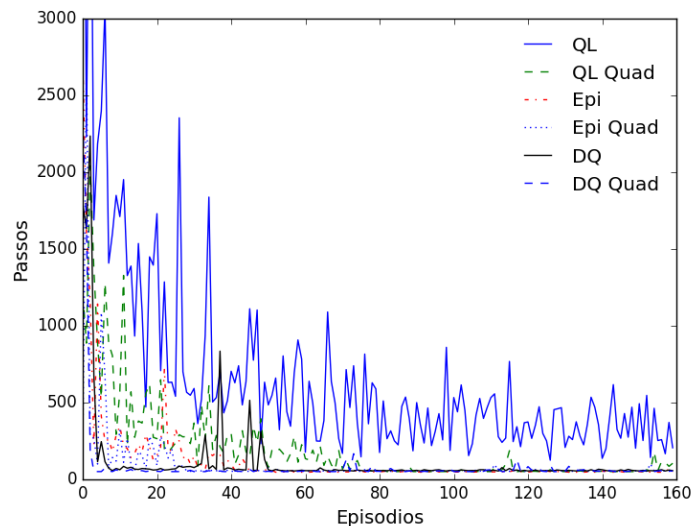


Figura 66 - Evolução dos algoritmos base e com memória *Quadtree* no ambiente 3 com grau de detalhe 2.

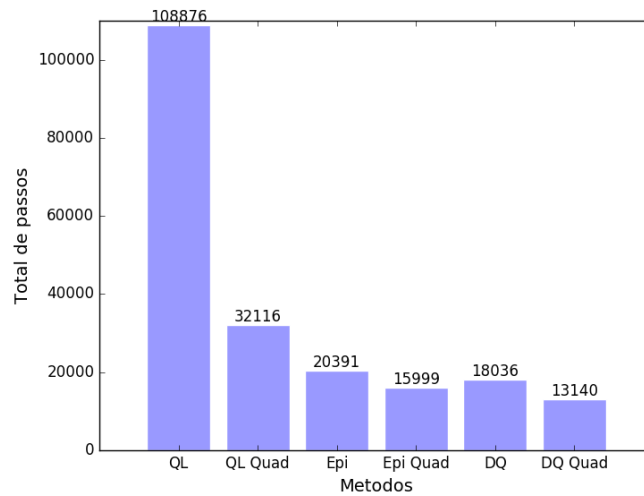


Figura 67 - Contabilização do total de passos dos algoritmos base e com memória *quadtree* no ambiente 3 com grau de detalhe 2.

A rotina 2 visa observar os resultados do agente hierárquico considerando o escalamento da dimensão do ambiente. Os resultados obtidos permanecem coerentes com os constatados na rotina 1. A diferença, na quantidade de processamento em termos de passos totais, entre os algoritmos *Q-Learning* é ainda mais evidente.

7.3.5 Aprendizagem com Memória Selectiva de Instâncias (MSI) – Rotina 1

Parâmetros		Valor
Aprendizagem	Reforço máximo	100
	Número de acções	8
	ϵ	0.05
	α	0.5
	γ	0.95
	Simulações internas (M. Episod. & Dyna-Q)	50
Simulação	Ambiente	3
	Detalhe	1
	Numero de Episódios	100
Relativos à memória selectiva de instâncias	Raio de detalhe	0.9
	Raio selectivo	$\sqrt{2}$
	Raio abstracto	3

Tabela 7 - Parametrização dos agentes adaptativos para os métodos base e com memória selectiva de instâncias da rotina 1.

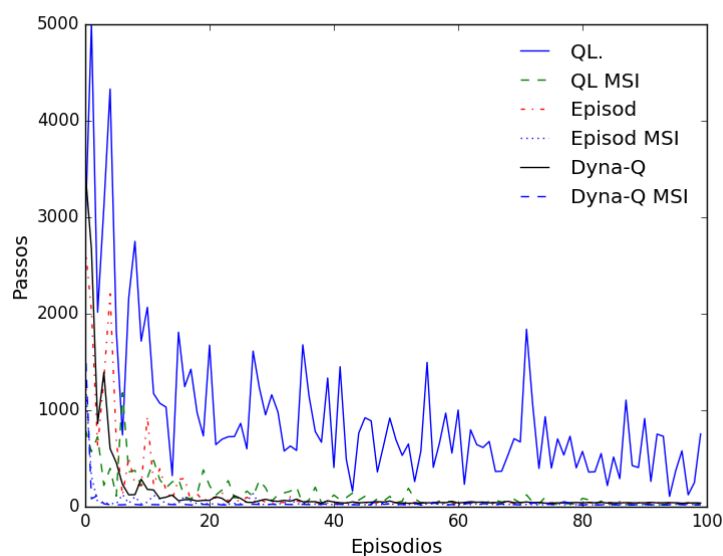


Figura 68 - Evolução dos algoritmos base e com memória selectiva de instâncias no ambiente 3 com grau de detalhe 1.

Os dados obtidos da comparação entre o uso de uma memória selectiva de instâncias (*MSI*) e uma memória de aprendizagem geral perspectivam o potencial teórico previsto. Os tempos de convergência diminuíram todos, com este tipo de memória a permitir uma aprendizagem mais rápida.

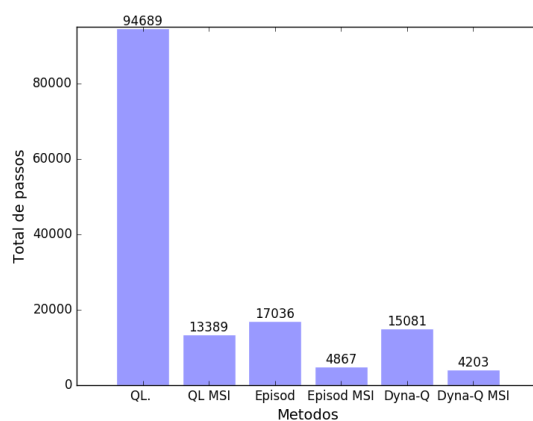


Figura 69 - Contabilização do total de passos dos algoritmos base e com memória selectiva de instâncias no ambiente 3 com grau de detalhe 1.

Analisando a Figura 69, também se verifica uma poupança muito relevante em relação aos métodos base no processamento total. O método Q-Learning com memória selectiva consegue superiorizar-se em relação à memória episódica e Dyna-Q base. Sendo que os restantes conseguem um desempenho que faz prever alguma robustez à escala do ambiente.

7.3.6 Aprendizagem MSI & Hierárquica Quadtree – Rotina 1

A comparação dos dois tipos de memória de abstracção criados requer o uso de apenas quatro acções devido às limitações de transição de estado diagonal da *quadtree*.

Parâmetros		Valor
Aprendizagem	Reforço máximo	100
	Número de acções	4
	ϵ	0.05
	α	0.5
	γ	0.95
	Simulações internas (M. Episod. & Dyna-Q)	100
Simulação	Ambiente	3
	Detalhe	1
	Numero de Episódios	80
Relativos à memória selectiva de instâncias	Raio de detalhe	0.9
	Raio selectivo	$\sqrt{2}$
	Raio abstracto	3

Tabela 8- Parametrização dos agentes adaptativos para os métodos com memória selectiva de instâncias e *quadtree* da rotina 1.

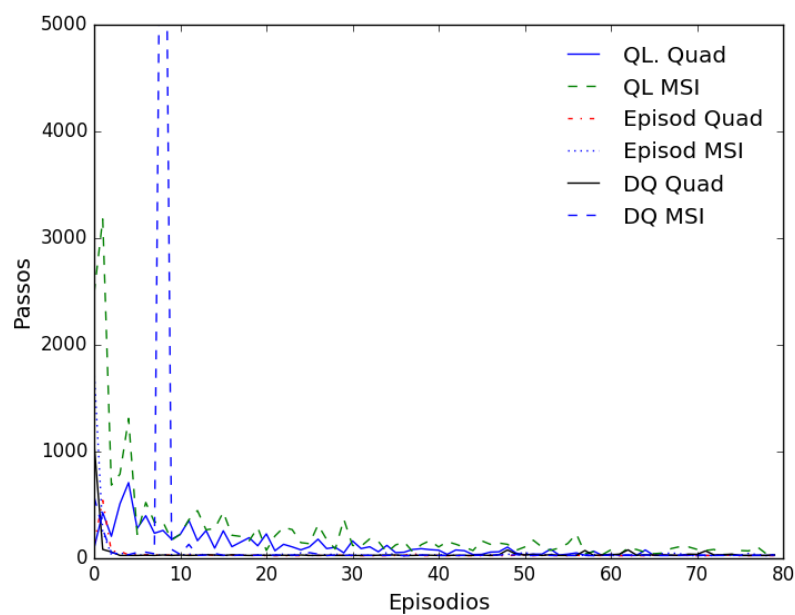


Figura 70- Evolução dos algoritmos com memória selectiva de instâncias e *quadtree* no ambiente 3 com grau de detalhe 1.

Realizando a análise à Figura 70, verifica-se alguns problemas que ocorreram com a memória selectiva de instâncias, tendo obtido um desempenho e tempo de convergência piores que a memória hierárquica *quadtree*.

Esta falta de desempenho prende-se com as transições entre nível abstracto e concreto, onde com apenas quatro acções se torna ainda mais difícil a criação detalhe e definição de novas instâncias. A *quadtree* obtém vantagem no sentido em que suaviza esse aprofundar de detalhe ao realizar a divisão recursivamente para aquela região.

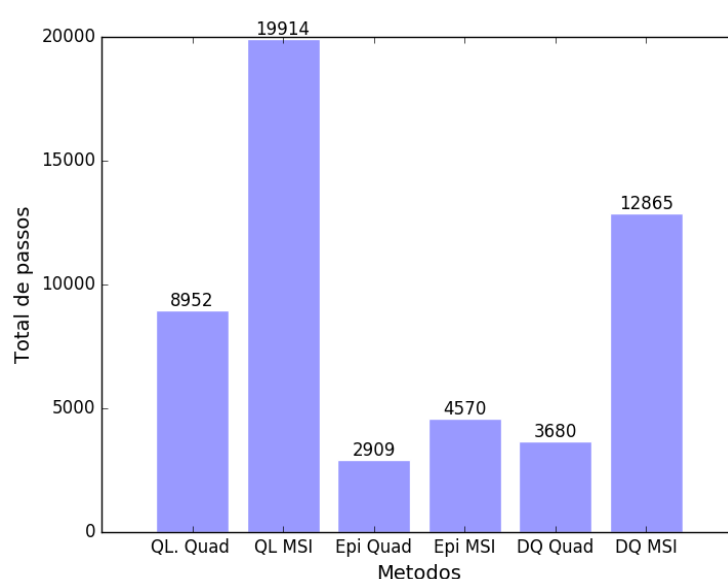


Figura 71 - Contabilização do total de passos dos algoritmos com memória selectiva de instâncias e *quadtree* no ambiente 3 com grau de detalhe 1.

Em termos de processamento total, a *quadtree* obtém melhores resultados, devido sobretudo aos problemas já referidos da memória selectiva. Contudo, é importante não esquecer o caracter geral da memória de instâncias quanto à representação de estado, o que não acontece com a *quadtree*.

7.3.7 Aprendizagem MSI Integr. react. & Integr. react. simples – Rotina 1

A criação desta rotina específica visa comparar a integração de reactividade simples com a integração de reactividade e a memória selectiva de instâncias.

Parâmetros		Valor
Aprendizagem	Reforço máximo	100
	Número de acções	8
	ε	0.05
	α	0.5
	γ	0.95
	Simulações internas (M. Episod. & Dyna-Q)	100
Simulação	Ambiente	3
	Detalhe	2
	Numero de Episódios	100
Relativos à memória selectiva de instâncias	Raio de detalhe	$\sqrt{2}$
	Raio selectivo	3
	Raio abstracto	5

Tabela 9 - Parametrização dos agentes adaptativos para os métodos com integração reactiva simples e integração reactiva com memória selectiva de instâncias da rotina 1.

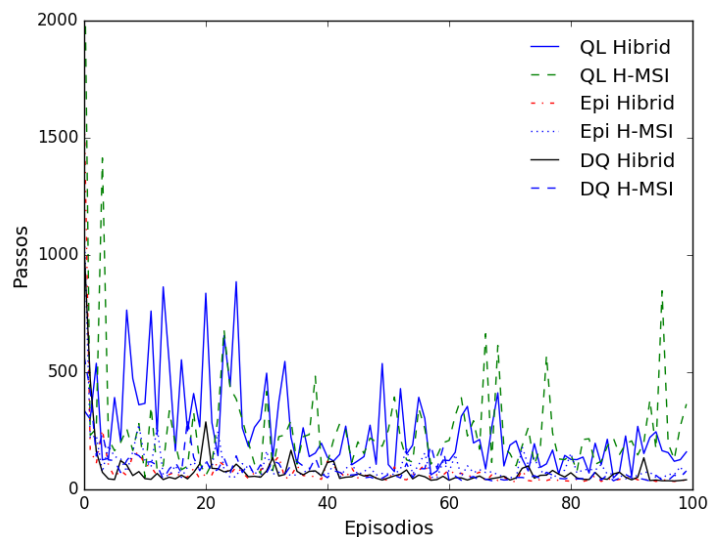


Figura 72 - Evolução dos algoritmos dos algoritmos com integração reactiva simples e integração reactiva com memória selectiva de instâncias no ambiente 3 com grau de detalhe 2.

Analisando a evolução dos métodos, é visível o desempenho próximo, mas pior, com uso de memória selectiva. Este desempenho inferior, prende-se possivelmente com os problemas existentes entre transições do nível abstracto para o concreto.

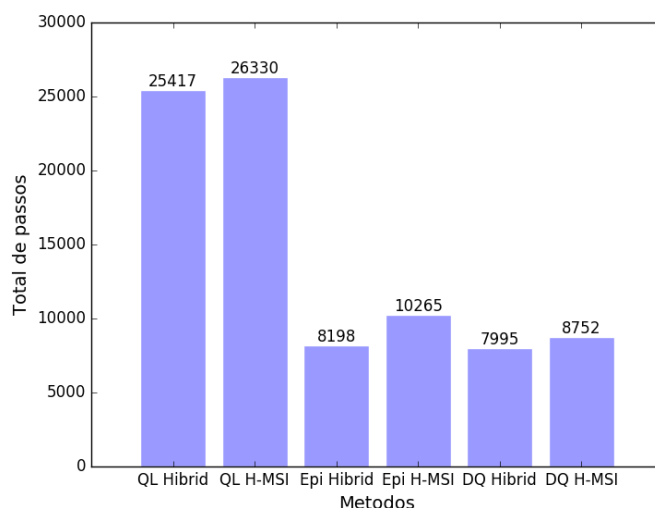


Figura 73 - Contabilização do total de passos dos algoritmos dos algoritmos com integração reactiva simples e integração reactiva com memória selectiva de instâncias no ambiente 3 com grau de detalhe 2.

7.3.8 Aprendizagem MSI Integr. react. & Integr. react. simples – Rotina 2

Estando-se consciente do potencial da memória selectiva para a capacidade de abstrair, foi realizado uma rotina adicional para continuar a comparação destas soluções em função da evolução do detalhe do ambiente.

Contudo, ao proceder com o aumento de detalhe do ambiente verifica-se uma inversão, com o potencial da técnica de abstracção a emergir. Devido ao tempo consumido em simulação pelos métodos devido ao detalhe elevado, apenas foram usados os mecanismos com memória episódica.

Parâmetros		Valor
Aprendizagem	Reforço máximo	100
	Número de acções	8
	ε	0.05
	α	0.5
	γ	0.95
	Simulações internas (M. Episod.)	100
Simulação	Ambiente	3
	Detalhe	3
	Numero de Episódios	150
Relativos à memória selectiva de instâncias	Raio de detalhe	$2\sqrt{2}$
	Raio selectivo	5
	Raio abstracto	7

Tabela 10 - Parametrização dos agentes adaptativos para os métodos com integração reactiva simples e integração reactiva com memória selectiva de instâncias da rotina 2.

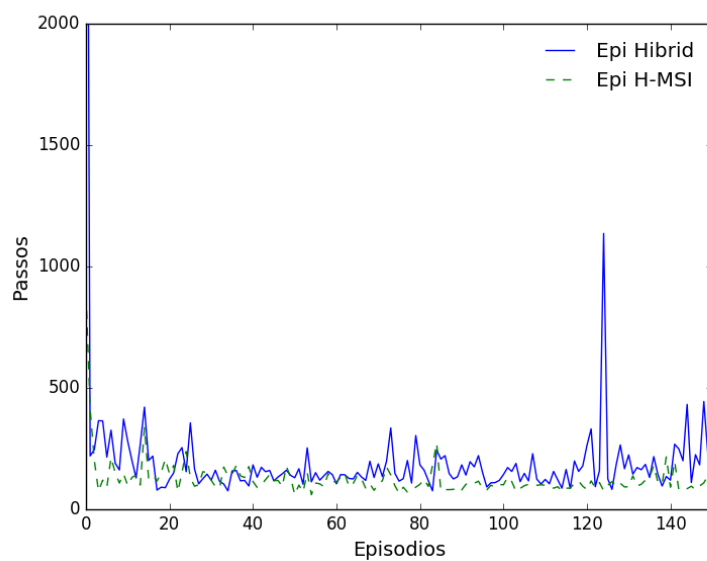


Figura 74 - Evolução dos algoritmos dos algoritmos com integração reactiva simples e integração reactiva com memória selectiva de instâncias no ambiente 3 com grau de detalhe 3.

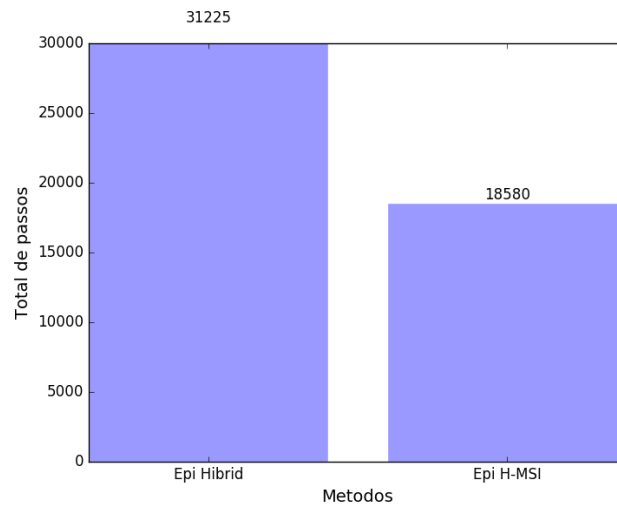


Figura 75 - Contabilização do total de passos dos algoritmos dos algoritmos com integração reactiva simples e integração reactiva com memória selectiva de instâncias no ambiente 3 com grau de detalhe 3.

7.4 Considerações sobre os resultados experimentais

Esta secção procede com algumas considerações sobre os resultados experimentais que não puderam ser observadas pelas rotinas efectuadas, sendo que, nem todas as vertentes e consequentes resultados podem ser comparados entre si.

7.4.1 Diferenças entre os métodos *Dyna-Q* e *Memória Episódica*

Observando os resultados das rotinas destaca-se a diferença de desempenho entre o método *Dyna-Q* e o método baseado em *Memória episódica*. Embora seja diminuta, a verdade é que a *Memória episódica* obtém resultados de desempenho ligeiramente piores do que o *Dyna-Q*, quando deveriam ser muito idênticos.

Esta diferença está possivelmente relacionada com o facto de no método *Dyna-Q* não existir episódios repetidos para um determinado par (s, a) , ao passo que, no método baseado em *Memória episódica* podem existir, na memória de repetições, experiências com os mesmos pares estado-acção (s, a) . Desta forma a propagação de valor não será tão eficiente.

Uma forma de resolver esta questão passou por testar o uso de uma Memória episódica com uma memória de repetição com unicidade de pares. Eliminando os pares repetidos, os resultados obtidos já corresponderam ao esperado.

7.4.2 Integração de reactividade

Na abordagem relativa à integração de reactividade, com o coordenador que verifica existência de valor na função Q obteve-se o resultado expectável. No entanto, é relevante destacar duas fases de comportamento distinto na obtenção do objectivo. Na primeira iteração, onde não existe pares estado-acção na função valor Q , o agente dirige-se para o óptimo local, aprende a sair deste com base na política gerada na camada adaptativa, e por fim, alcança o alvo através da política comportamental da camada reactiva, devido ao seguimento de potencial.

Contudo, em iterações posteriores, o caminho até ao alvo já contém pares estado-acção na função valor pelo que o valor é propagado devido ao mecanismo de aprendizagem subjacente. Esta propagação, após convergir e estabilizar, gera uma política sub-óptima, mas satisfatória, para alcançar o alvo, o qual o agente irá sempre seguir devido ao critério de selecção de acção do coordenador.

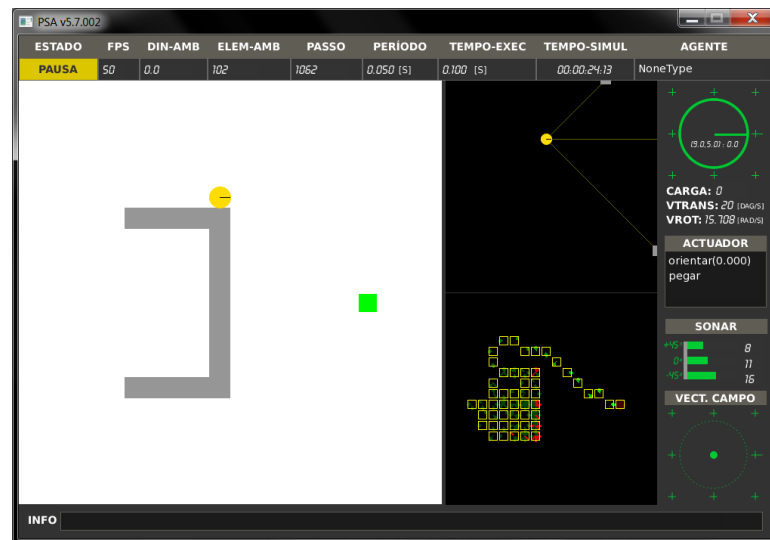


Figura 76 – Visualização do primeiro episódio, devido a seguir o potencial da camada reactiva o agente alcançou rapidamente o alvo.

As arquitecturas híbridas apresentam uma solução eventualmente mais viável em função da coordenação de acção existente. No entanto, este tipo de soluções estabelece uma abstracção entre as camadas existentes levando à eventual não interacção directa entre camadas.

Adicionalmente, para o coordenador de acção simples “*existe Q*” implementado, a camada reactiva torna-se inútil a partir do momento em que já existe conhecimento obtido por parte da camada adaptativa. Esta situação traduz-se numa preferência do aproveitamento da aprendizagem em detrimento das capacidades reactivas.

7.4.3 Agente adaptativo hierárquico *quadtree*

A abordagem hierárquica acarreta vantagens em lidar com um espaço de estados abrangente, na medida em que, a capacidade de abstracção permite extrapolar acções e aprendizagem para estados diferentes. Essa característica é concebida através da definição de um macro estado que agrega estados com características similares e que não necessitam de maior detalhe.

Contudo, a principal vantagem destes métodos revela fraquezas relativamente à transição entre os diferentes níveis de abstracção existentes. Não existem garantias de uma solução óptima, dado a existência de estados macro, resultando em possíveis perdas de detalhe sobre situações relevantes por descobrir.

Adicionalmente, é importante notar que, estes tipos de estrutura de memória para aprendizagem, funcionam exclusivamente para uma forma de espaço de estados bidimensional, sendo necessário outras soluções em espaços com diferentes formatos.

7.4.4 Agente adaptativo com memória selectiva de instâncias

O método de aprendizagem com memória selectiva de instâncias padece do mesmo tipo de problema em lidar com a mudança do nível de detalhe. Se por um lado existe vantagem em abstrair e conseguir abranger regiões com a mesma política, por outro, quando é necessário reduzir o grau de detalhe, transitando de um nível superior de abstracção para o concreto, existe o problema de como esse tipo de transição é realizada.

Esta técnica de abstracção não é indiferente à situação descrita anteriormente e, conseqüentemente, a mesma traduz-se em perda de

qualidade na solução obtida comparativamente ao que seria a solução óptima. A tradução de situações deste tipo pode ser observada na figura seguinte.

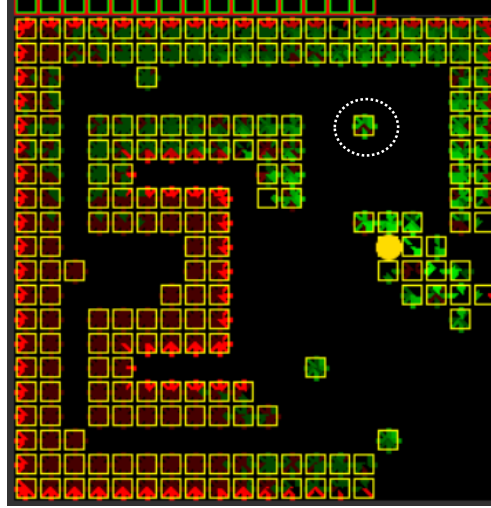


Figura 77 - Função acção-valor gerada por uma simulação recorrendo à memória selectiva. A branco destaca-se uma memória de instância.

O círculo branco evidencia um estado abstracto que corresponde a uma memória de instância criada. A propagação do reforço positivo da recolha do alvo pode gerar, para o estado referido, valores maiores para as direcções que apontam a posição do alvo. Contudo, tais direcções não estão em linha com o estado objectivo. A criação de instâncias perto do alvo é necessária para aumentar o detalhe na região e diminuir o efeito da transição entre níveis de detalhe.

7.4.5 Agente *Deep-Q* adaptado e treino *offline*

Os problemas inerentes nesta área relativamente ao tempo e consumo de recursos computacionais na convergência de uma solução, são de igual forma, transportados na aplicação de soluções que recorrem a funções de aproximação. Numa rede neuronal, quanto maior for a complexidade do problema a treinar, maior o tempo de uma eventual convergência e minimização da função de custo considerada.

Usando um ambiente bastante simples (Figura 48) provou-se que de forma *offline* é possível treinar com sucesso uma rede para que esta aprenda uma função de acção-valor previamente aprendida.

Parâmetros		Valor
Ambiente Gravado (2x2)	Reforço máximo	100
	Número de acções	4
	E	0.05
	A	0.5
	Γ	0.95
Rede	<i>Input neurons</i>	2
	<i>Hidden neurons</i>	128
	<i>Output neurons</i>	4
Treino	Método Rprop	epochs=50000, show=1000, goal=0.001, lr = 0.0001, adapt = False, rate_dec = 0.5, rate_inc = 1.2, rate_min = 1e-9, rate_max=50

Tabela 11 - Parametrização da rede neuronal para teste *offline*.

Foram testadas várias configurações de rede e métodos de treino para a função valor do ambiente simplista previamente gravada. O método *Rprop*, a versão não estocástica ao *rmsprop*, foi o método cujo desempenho e rapidez foi superior aos restantes. A normalização dos dados de entrada é também importante de forma a escalar os dados para serem propagados pela rede e funções de activação.

$$Y_{norm} = \frac{Y - Min(Y)}{Max(Y) - Min(Y)}$$

Porém, colocar uma rede a ser treinada iterativamente, i.e., de forma *online*, revela-se um desafio árduo devido ao limite de processamento disponível entre cada passo de execução. Este problema agrava-se sobretudo em função da dimensão do espaço de estados tomado.

Métodos de treino mais eficientes e que melhor explorem o espaço dimensional da função de custo de modo a minimizar o erro podem revelar-se escassos quando o próprio treino a cada iteração é limitado.

À data desta dissertação não foi possível obter uma solução *online* viável para este tipo de métodos, sendo plausível considerar que seja

impossível dado a representação de estado cartesiano e a limitação de recursos computacionais existente. Considerando a forma de treino com episódios (passos de execução) e a escassa janela temporal para o fazer, verifica-se que, o contexto de utilização actual assenta num cenário muito diferente quando comparado ao uso de estruturas distribuídas, que detêm enorme robustez em processamento.

7.4.6 Método *HSL* e Agente com camada reactiva com memória

A arquitectura híbrida com camada reactiva com memória e o método HSL foram usados predominantemente para investigação com a representação limitada de estado.

O HSL não é usado em comparações de rotinas anteriores devido a diferenças entre heurísticas. A explicação para tal, deve-se à possibilidade de usar dois tipos de heurística com este método: heurística de potencial e a heurística de estado final. A heurística de estado final encontra-se melhor adaptada aos ambientes de simulação, mas, não corresponde à heurística de potencial usada pelos restantes métodos com uso de heurística.

Por sua vez, o uso de potencial com este método apenas é útil com uma representação de estado sensorial, ou cartesiana com apenas quatro acções. Este factor explica-se devido aos sensores de potencial que, sendo apenas três, não associam variações de potencial às restantes direcções possíveis de actuar no HSL, resultando na criação de memórias esparsas não conexas.

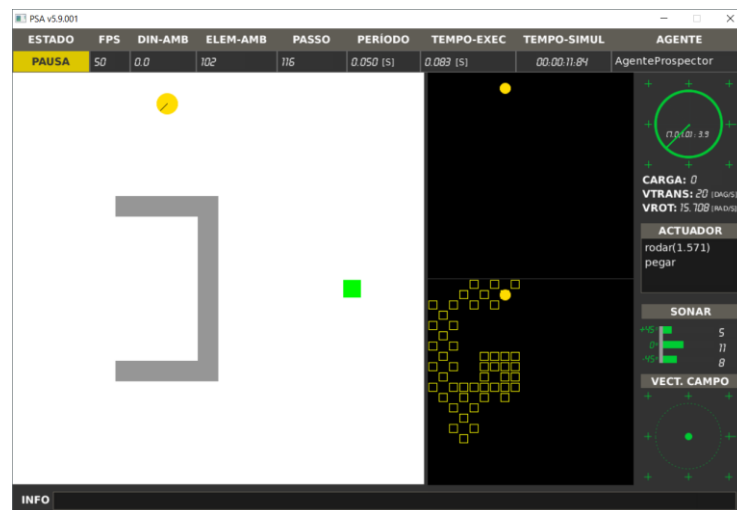


Figura 78 - Agente adaptativo HSL com heurística de potencial utilizando 8 acções. É visível a memória de aprendizagem desconexa.

O mesmo não acontece com a heurística de estado final, que funciona correctamente para uma representação de estado cartesiano, onde é possível calcular o valor heurístico máximo de todas as acções para um estado.

Em relação à comparação entre as diferentes representações de estado, as mesmas não são passíveis de comparação. Os seus propósitos e informação discriminativa são diferentes.

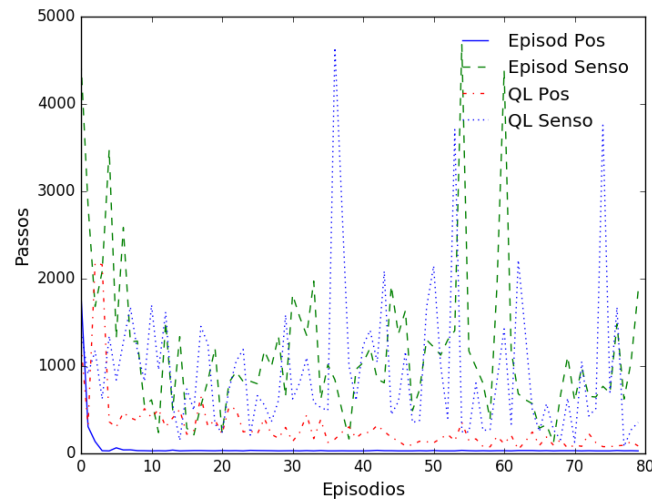


Figura 79 - Gráfico da evolução dos algoritmos base com representação de estado diferente no ambiente 3 com grau de detalhe 1.

Uma vez que não existe modelo interno para a representação sensorial apenas os algoritmos *Q-Learning* e *Memória episódica* são usados. Como é possível observar, o tempo que a representação sensorial leva para convergir é muito superior à representação cartesiana, utilizando ambas 4 acções.

7.4.7 Representação limitada de estado

A capacidade de adaptar, e encontrar, uma representação de estado que permita deter um espaço de estados adequado aos recursos disponíveis é um factor importante para o desempenho do processo de aprendizagem. Uma representação de estado mais simples potencia a diminuição do número de estados a explorar.

Verifica-se que a mudança de representação no estado acarreta mudanças no comportamento do agente quando comparado ao uso de estado posicional. No entanto, visualizando o comportamento do agente não é possível afirmar se este convergiu para uma política óptima, pois, a função de

transição modificou-se juntamente com o espaço de estados. Embora se verifique um comportamento semelhante e coerente ao fim de algum tempo e iterações, é visível que o caminho aprendido diverge consideravelmente em relação ao caminho óptimo obtido pela representação posicional. Este resultado advém principalmente pela, consideravelmente menor, informação discriminativa da representação de estado sensorial.

Sendo a política aprendida bastante distinta consoante a representação, é, porém, observável a capacidade de extrapolar a aprendizagem por parte da representação sensorial no ambiente 3b. Esta aptidão traduz-se de imediato numa “poupança de aprendizagem” em relação a um agente com estado posicional após ambos aprenderem a contornar o primeiro óptimo local do ambiente.

De forma complementar, é interessante estudar o parâmetro γ para esta representação. A aprendizagem modifica-se em função do parâmetro γ . Com um γ nulo, o agente aprende padrões de comportamento reactivo. Esta característica é interessante, pois permite obter um agente reactivo através de uma representação simples e sem recorrer a uma arquitectura reactiva pré-definida.

8 Conclusão

O constante desenvolvimento da aprendizagem por reforço, fruto de esforços de percursores como *Watkins* ou *Sutton*, levam-nos a encarar a A.R. como um domínio unificado. Esta vertente não consiste numa colecção de métodos individuais, mas sim, num conjunto de ideias e conceitos coerentes que atravessa cada um desses métodos [5].

Esta visão unificada permite identificar três princípios relevantes para qualquer potencial modelo de inteligência. Todos estes métodos exploram a estimação de funções de valor, operam na base da retenção de valores ao longo do espaço de estados para delinear trajectórias, e mantêm uma estratégia baseada na iteração de política generalizada. Esta iteração de política generalizada (*GPI - generalized policy iteration* [5]), traduz-se na manutenção de uma função valor e de uma política aproximadas, ambas em constante evolução e melhoramento com base na outra.

O tempo e as capacidades de raciocínio disponíveis para uma dada problemática são factores limitativos que reflectem a complexidade inerente num cenário de operação real. As visões de racionalidade, conjuntamente com os conceitos destacados no problema da limitação de recursos, fornecem algumas abordagens e uma nova perspectiva sobre a definição de racionalidade.

No âmbito desta dissertação propôs-se o estudo de propostas relativas à aprendizagem por reforço que explorem soluções com potencial de enquadramento e relacionadas com a escassez de recursos. Nesse sentido, a aprendizagem por reforço com recursos limitados representa uma vertente com grande potencial para diversas áreas de aplicação e necessidades.

O estudo da integração de reactividade em agentes adaptativos constituiu um ponto de entrada na investigação e experimentação no sentido de perceber que vantagens uma solução híbrida sortiria. A utilização de uma arquitectura horizontal de camadas acarreta um ponto vital: a coordenação de acção. A solução primordial de integração revelou-se demasiado simples pois, embora estabeleça uma abstracção entre as camadas, não existe interacção directa entre as mesmas. Contudo, esta primeira abordagem permitiu identificar que, estas arquitecturas, disponibilizam uma robustez e viabilidade no desempenho em função da coordenação de acção considerada.

Adicionalmente, o conceito de integração permite associar a qualquer agente adaptativo uma componente reactiva deste tipo. Esta possibilidade remete a integração de reactividade para não sendo só uma solução, mas também uma componente importante de outras possíveis soluções.

De forma similar à integração de reactividade, também as técnicas de abstracção detêm um ponto critico: o critério de abstracção. A decisão de quando abstrair é importante na medida em que define quando deve ou não manter detalhe, reflectindo-se no desempenho e entrando no domínio da satisfação da racionalidade limitada.

Embora as técnicas abordadas tenham apenas sido testadas com um espaço de estados bidimensional, é possível identificar um factor muito importante a nível geral: a capacidade deste tipo de métodos lidar com as transições entre graus de abstracção. Assim, é importante perceber os contornos do critério de definição de um evento relevante e ainda as especificidades de cada solução, como os raios de influência presentes na memória selectiva de instâncias. Soluções, como a aprendizagem feudal [10], disponibilizam uma navegação mais detalhada com camadas de abstracção à custa de maior complexidade computacional.

A abordagem recorrendo a uma aprendizagem multinível (*deep learning*) para abstracção de estado, não se relevou prática relativamente ao método *Deep-Q* utilizado. Constatou-se que o treino de uma rede neuronal substituta de uma função valor é uma possibilidade. Embora a utilização de recursos limitados seja evidente, a rápida convergência revela-se um problema para este tipo de soluções.

Seguindo os princípios da racionalidade limitada, os métodos heurísticos baseiam-se no auxilio de heurísticas para melhorar o processo de aprendizagem, quer em orientação em relação ao objectivo, quer em rapidez de convergência e aliviar o tempo de processamento.

Estes métodos sofrem, no entanto, de um problema inerente e limitativo. Para cada problema, em função do ambiente e do próprio sistema inteligente, poderá ser difícil formular uma heurística, mesmo se possível. Sendo a aprendizagem por reforço aplicável, de forma geral, a diferentes tipos de problemas, a representação de estado e a definição de uma heurística admissível são necessários em cada caso para este tipo de métodos.

A consideração de uma representação limitada de estado permitiu perceber os contornos e dificuldades em conseguir realizar aprendizagem num

contexto de operação real. A informação presente numa representação limitada pode traduzir-se em ambiguidade de estado, dificultando o processo de aprendizagem e até inviabilizar aprendizagens em situações pertinentes, mas percebidas de forma ambígua.

Uma representação limitada como o designado estado sensorial, aliado a uma arquitectura híbrida com camada reactiva com memória consegue demonstrar a possibilidade de abordagens deste tipo. Contudo não pode ser expectável um desempenho e tempo de aprendizagem que rivalize com uma representação de estado mais detalhada.

A vertente robótica permitiu perceber os contornos e dificuldades identificados em cenários de operação real. A implementação do agente robótico adaptativo, por sua vez, encontra-se limitada pela plataforma física, uma vez que, as representações de estado estão não só condicionadas pelas características do ambiente, mas também, devido a restrições intrínsecas ao sistema.

8.1 Trabalho Futuro

O trabalho de investigação e desenvolvimento constitui sempre um desafio difícil de terminar. Ao trabalho realizado foi necessário limitar o seu âmbito e respectiva investigação no sentido de culminar com uma abrangência significativa e suficiente das vertentes seleccionadas.

Fora da estruturação criada para o presente trabalho, vários caminhos e soluções possíveis ficam por estudar e concretizar, remetendo as mesmas para desenvolvimento futuro. Estas direcções futuras permitem obter uma visão de eventuais perspectivas relevantes e promissoras em relação à temática e base apresentada.

A abordagem baseada em técnicas de abstracção (secção 5.1.2) oferece uma base interessante uma vez que, permite a visualização da estrutura da memória de aprendizagem e verificar as vantagens deste tipo de técnicas.

Considerar um agente fora de um ambiente fechado e controlado, remete a sua própria noção de estado para uma maior complexidade, consoante a existência de diferentes indicadores. Um agente com estado sensorial, como o

anteriormente utilizado, implica no contexto de abstracção, encontrar os próprios agregados de informação dos quais seja possível abstrair.

Na sequência dos aspectos realçados, seria interessante testar e, se necessário, adaptar a memória selectiva de instâncias com uma representação de estado diferente ao espaço cartesiano. Esta técnica permite o uso de um estado como um vector multidimensional com diferentes características, sendo por isso motivo de uma exploração mais aprofundada.

As heurísticas, sendo bastante rápidas, contêm informação escassa do mundo. Investir tempo para entender de que forma seria possível encontrar e demonstrar heurísticas com maior eficácia e capacidade para descrever o ambiente seria importante.

Relativamente às funções de aproximação, apenas uma adaptação do algoritmo *Deep-Q* foi realizada e considerando o estado cartesiano. Uma vez que a biblioteca *ConvJS* [17] demonstrou potencialidades com um agente efectivamente treinado, seria aliciante transpor o mesmo algoritmo numa perspectiva de agente com estado sensorial e observar os resultados.

Ainda dentro do domínio envolvendo redes neuronais, existem outros tipos de arquitectura, como a arquitectura *CMAC* (*Cerebellar Model Articulation Controller*). A exploração deste tipo de arquitecturas poderá ser um bom foco de estudo e posterior investigação.

No contexto da arquitectura híbrida com camada reactiva com memória, seria interessante explorar a utilização de sequências de acções de carácter reactivo e colocar uma camada adaptativa a realizar aprendizagem sobre a sua utilização. Esta abordagem poderia suavizar a carga da camada adaptativa, reagindo com sequências de acções pré-concebidas em soluções já aprendidas.

Uma outra possibilidade na continuidade de investigação sobre esta arquitectura prende-se com equacionar uma camada adaptativa que aprendesse a seleccionar, de um conjunto de heurísticas, qual a melhor em determinada situação. Para compreender melhor esta perspectiva pode-se considerar uma heurística de potencial e uma heurística designada de “*campo aberto*”. Esta última define que, quanto mais longe de obstáculos o agente se encontra, mais fácil é a sua componente reactiva seguir o potencial e obter comportamento óptimo. Esta heurística pode ser fornecida, a título de exemplo, por distâncias captadas pelo agente. Por consequente, numa situação sem obstáculos próximos, o agente aprenderia a recorrer à típica

heurística de potencial; junto a obstáculos, a aprendizagem levaria a escolher a heurística de campo aberto.

Participando activamente no processo de selecção de acção, quer na camada reactiva, quer na camada adaptativa, qualquer que fosse a heurística seleccionada, orientaria o agente no sentido certo e esperado.

No domínio da robótica, a investigação e restante implementação de mais arquitecturas adaptativas seria vantajoso para perceber a sua capacidade de aplicação no mundo real. Adicionalmente, testar e criar arquitecturas adaptativas com o uso de um microcontrolador em mente, ao invés de algo mais robusto como o *Raspberry Pi*, permitiria definir melhor os contornos de possibilidades relativamente à escassez de recursos computacionais.

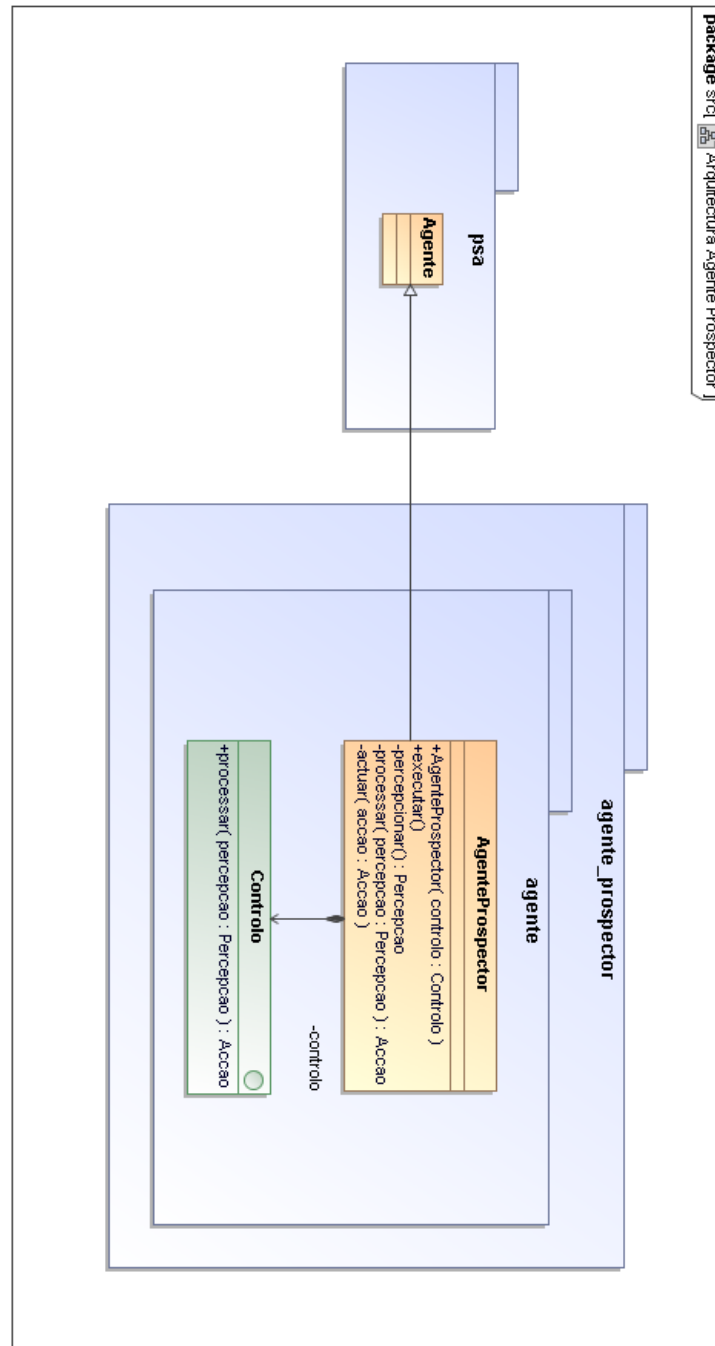
Bibliografia

- [1] S. J. Russell e P. Norvig, Artificial Intelligence: A Modern Approach, 2nd Ed., Prentice Hall, 2003.
- [2] R. A. Brooks, A robust layered control system for a mobile robot, Massachusetts Institute of Technology, 1986.
- [3] R. C. Arkin, Behavior-Based Robotics, MIT Press, 1998.
- [4] L. F. G. Morgado, Complementos de Inteligência Artificial - Textos de Apoio, 2015.
- [5] R. Sutton e A. Barto, Reinforcement Learning: An Introduction, MIT Press, 2012.
- [6] C. J. C. H. Watkins, Learning from Delayed Rewards, University of Cambridge, 1989.
- [7] D. Shiffman, "The Nature of Code - Chapter 10. Neural Networks," [Online]. Available: <http://natureofcode.com/book/chapter-10-neural-networks/>. [Acedido em 2016].
- [8] W. McCulloch e W. Pitts, "A logical calculus of the ideas immanent in nervous activity," em *Bulletin of Mathematical Biophysics Volume 5*, The University of Chicago Press, 1943, pp. 115 - 133.
- [9] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Cornell Aeronautical Laboratory, Psychological Review*, vol. 65, pp. 386-408, 1958.
- [10] S. Ruder, "An overview of gradient descent optimization algorithms," 26 06 2016. [Online]. Available: <http://sebastianruder.com/optimizing-gradient-descent/index.html#gradientdescentoptimizationalgorithms>. [Acedido em 12 2016].
- [11] P. Dayan e G. E. Hinton, Feudal Reinforcement Learning, 1992 .
- [12] R. A. C. Bianchi, R. Ros e R. L. d. Mántaras, Improving Reinforcement Learning by using Case Based Heuristics, 2009.
- [13] K. K. D. S. A. G. I. A. D. W. M. R. Volodymyr Mnih, "Playing Atari with Deep Reinforcement Learning," em *NIPS Deep Learning Workshop 2013*, 2013.
- [14] P. M. T. a. t. A. R. G. Gerd Gigerenzer, Simple heuristics that make us smart, Oxford University Press, 1999.
- [15] H. A. Simon, "Rational choice and the structure of environments," *Psychological Review*, vol. 63, pp. 129-138, 1956.
- [16] G. Gigerenzer e R. Selten, "Bounded Rationality The adaptive toolbox," The MIT Press, Cambridge, Massachusetts; London, England, 2001.

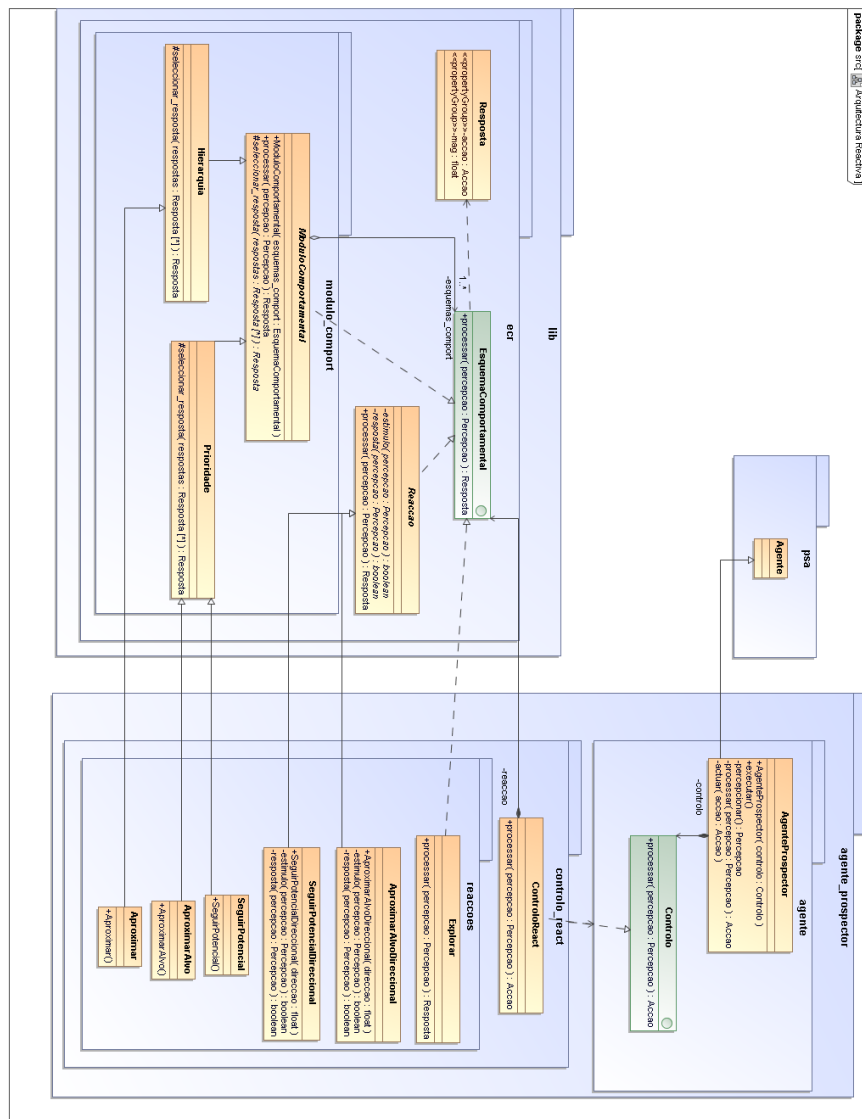
-
- [17] J. W. Payne, J. R. Bettman e E. J. Johnson, The Adaptive Decision Maker, New York: Cambridge University Press, 1993.
- [18] H. Tulleken, “Quadrees: Implementation,” Dev.Mag - A game development magazine, 2011. [Online]. Available: <http://devmag.org.za/2011/02/23/quadrees-implementation/>.
- [19] A. Karpathy, “ConvNetJS Deep Q Learning Demo,” [Online]. Available: <https://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html>. [Acedido em 2016].
- [20] M. Li, T. Zhang, Y. Chen e A. J. Smola, Efficient Mini-batch Training for Stochastic Optimization, 2014.
- [21] R. Sutton, Learning to predict by the methods of temporal differences, 1988.
- [22] R. Rojas, Neural Networks: A Systematic Introduction, Berlin: Springer, 1996.
- [23] G. Tesauro, Temporal Difference Learning and TD-Gammon, ACM, 1995.
- [24] M. Kesson, “Algorithmic - Quadtree (Python),” 2002. [Online]. Available: <http://www.fundza.com/algorithmic/quadtree/index.html>.
- [25] “Physical computing with Raspberry Pi - Buttons and Switches,” [Online]. Available: https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/buttons_and_switches/. [Acedido em 2016].
- [26] J. L. McClelland, “Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises - Chapter 9 Temporal-Difference Learning,” Dezembro 2015. [Online]. Available: <https://web.stanford.edu/group/pdplab/pdphandbook/handbookch10.html>. [Acedido em 2016].
- [27] V. Braitenberg, Vehicles: Experiments in synthetic psychology, Cambridge, MA:: MIT Press, 1984.

Apêndice A – Detalhe de Arquitectura

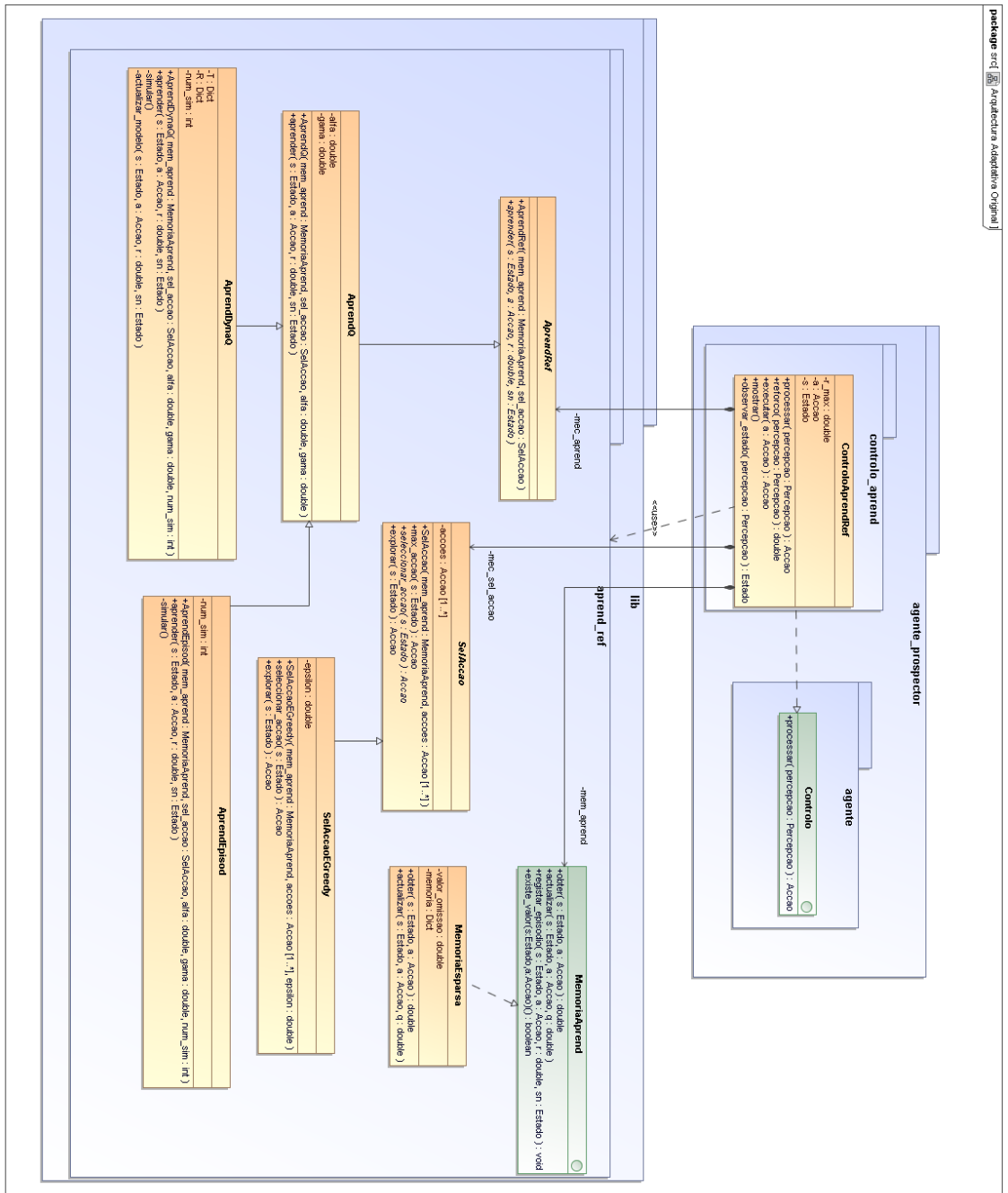
Agente Prospector



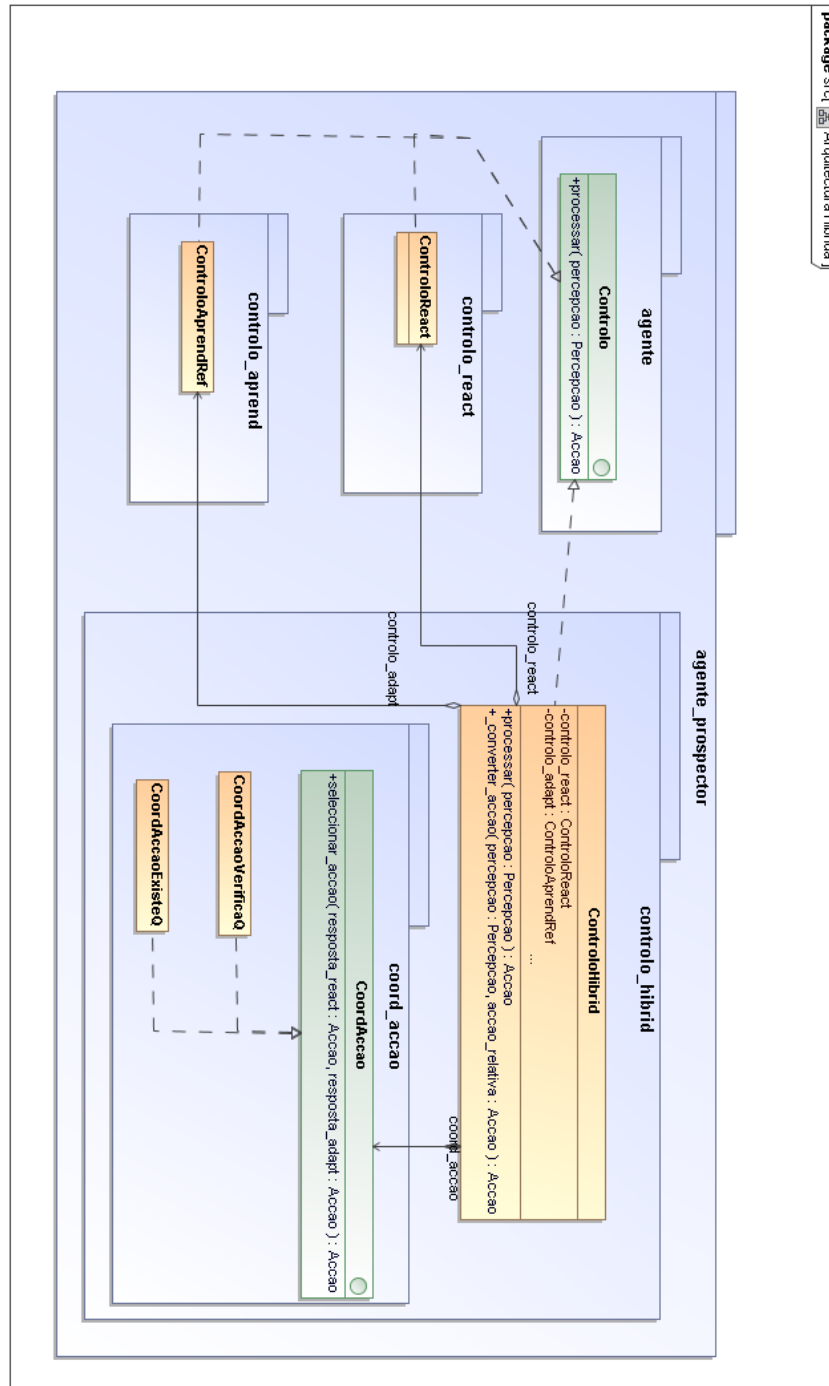
Arquitectura Reactiva



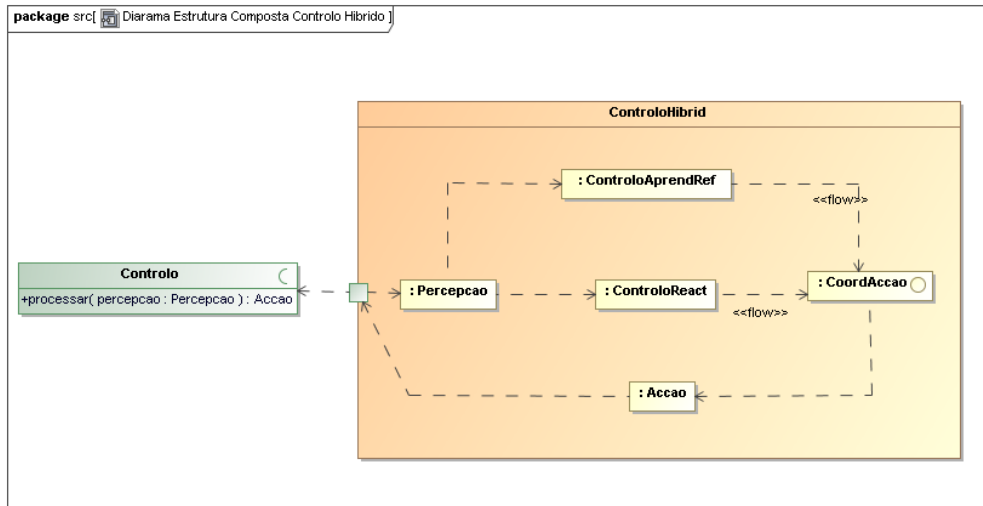
Arquitectura Adaptativa Base



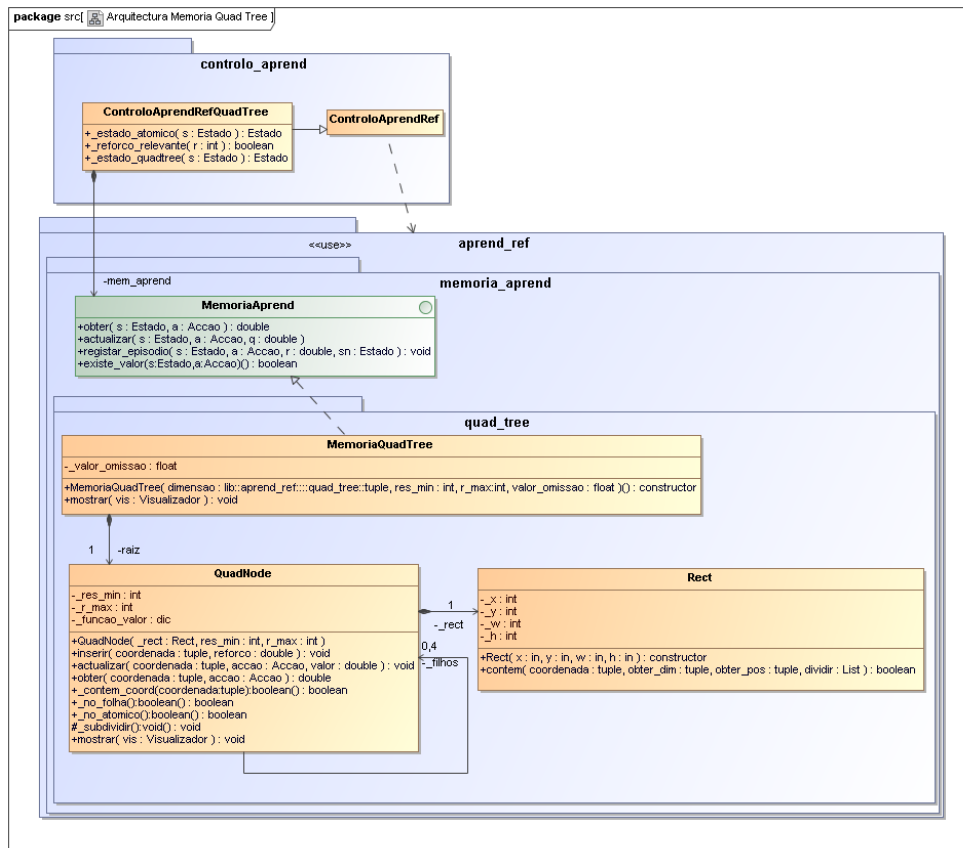
Arquitectura Híbrida Simple con integración de reactividade



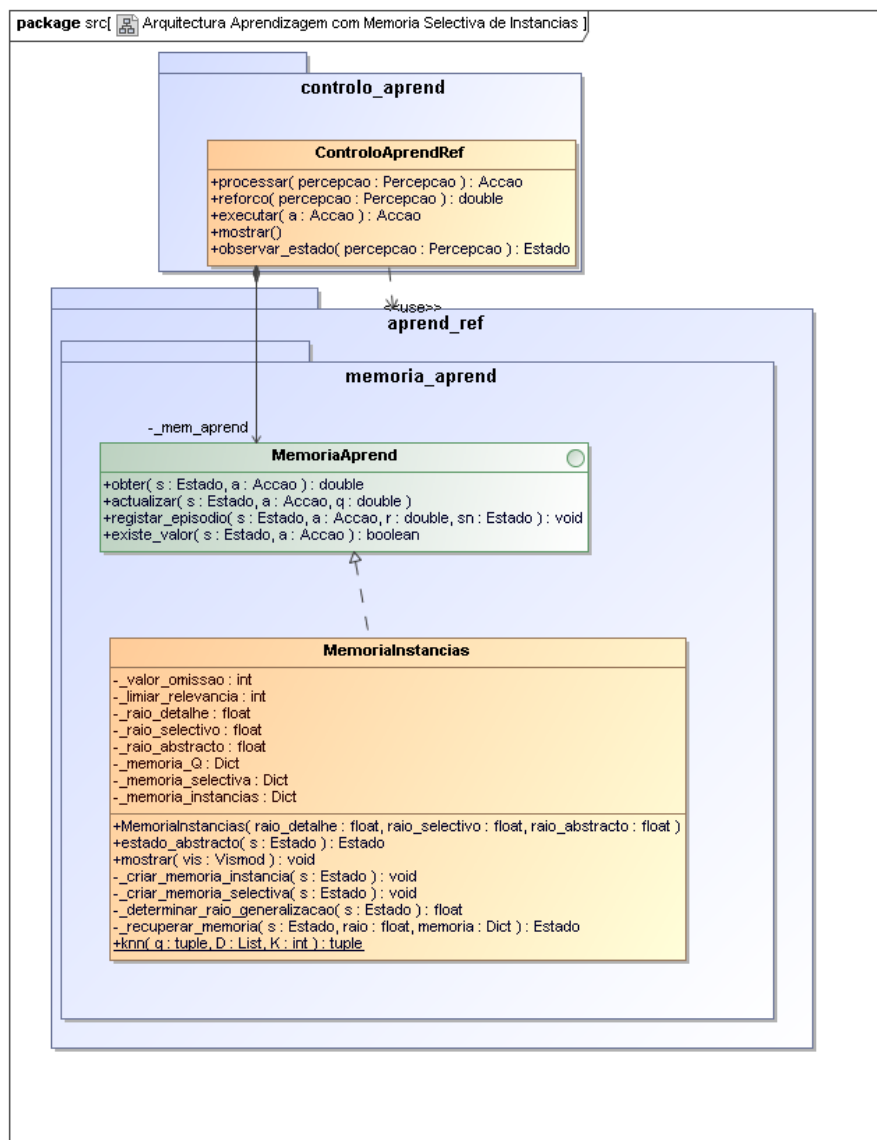
Estrutura Composta da Arquitectura com integração de reactividade



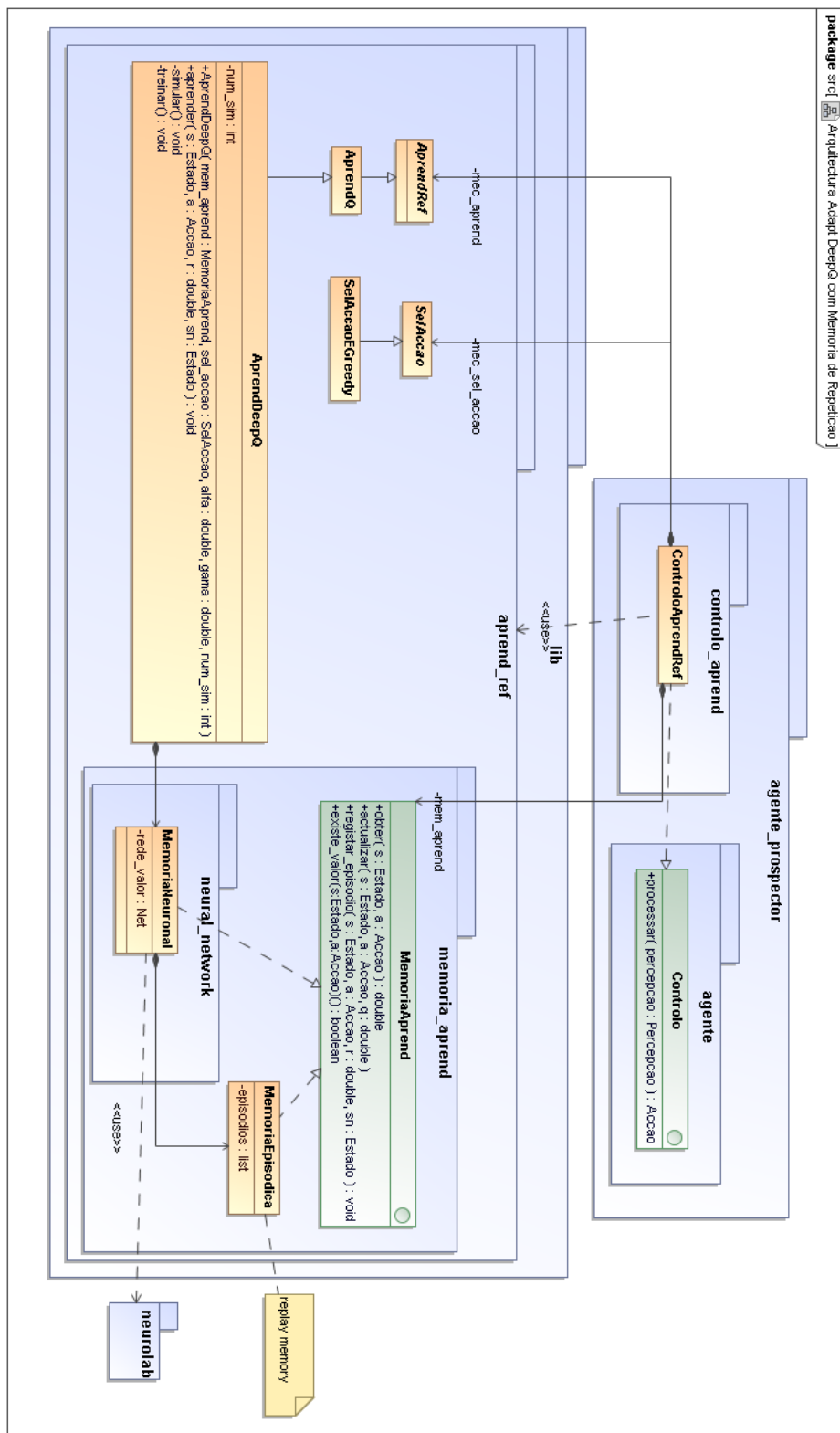
Arquitectura Adaptativa Hierárquica *QuadTree*



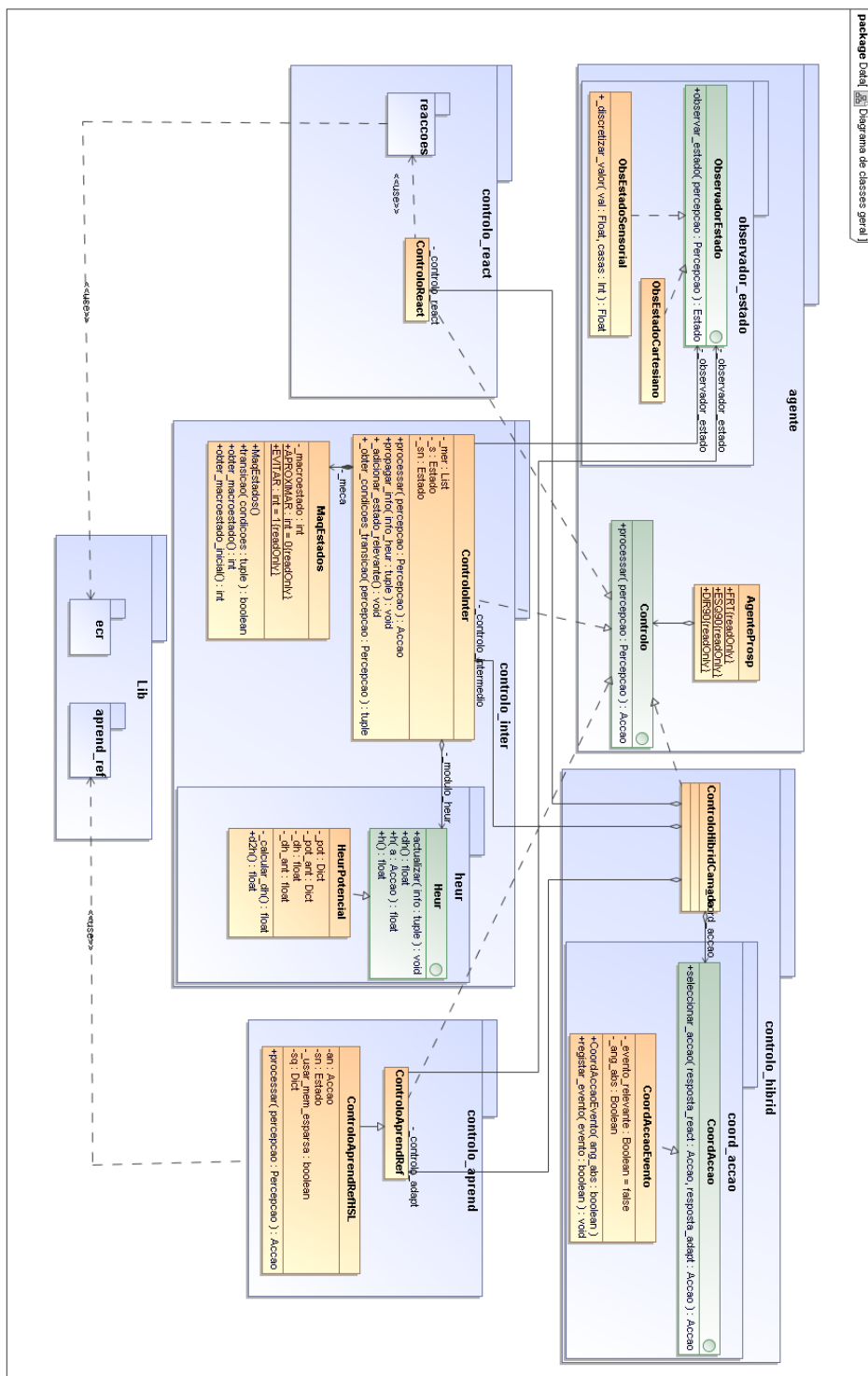
Arquitetura Adaptativa com Memória Selectiva de Instâncias



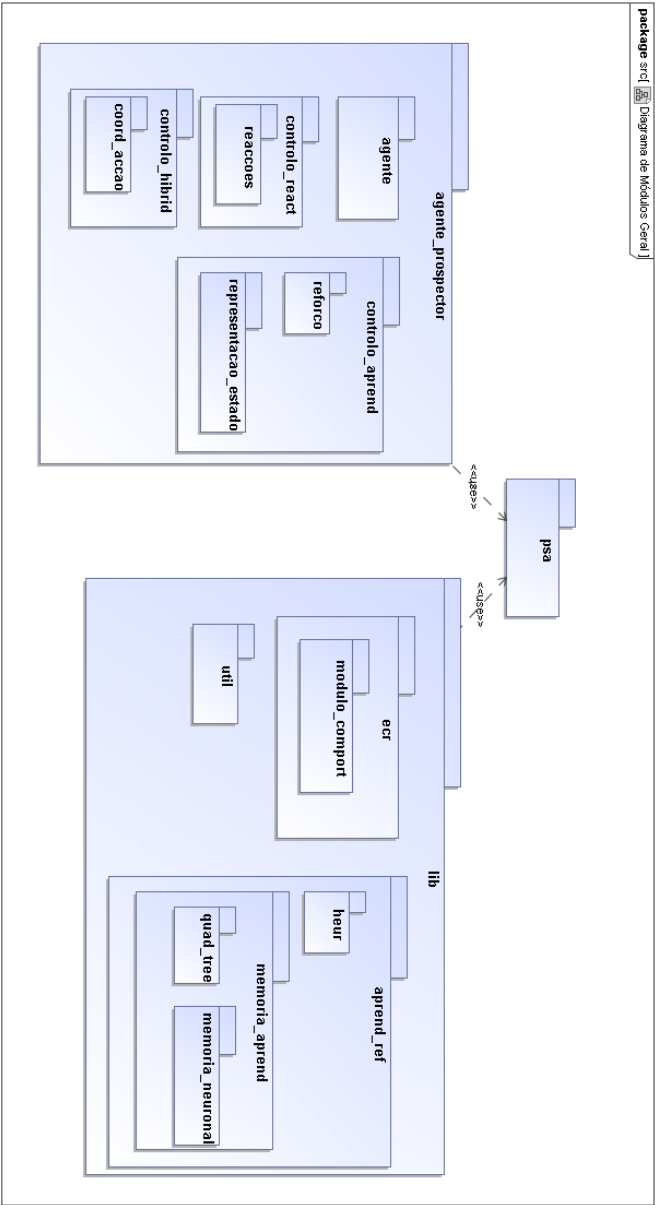
Arquitectura Deep-Q



Arquitetura híbrida de camada reactiva com memória



Organização Geral



Apêndice B – Vertente Robótica

Montagem técnica

Nesta secção são apresentados em detalhe todos os passos e informação técnica relativamente à montagem do agente robótico.

Micro Switch

O *Raspberry Pi*, sendo a unidade de processamento, disponibiliza uma interface de pins, designada por GPIO (*general purpose input/output*), com o propósito de facilitar a comunicação com qualquer outro componente electrónico compatível.

No caso deste interruptor de colisão, a interface GPIO é usada, não para comunicação, mas para detectar flutuações de tensão. Nesse sentido, é configurado um pin de entrada de forma a proceder à leitura da tensão enviada de uma fonte e, associar posteriormente uma lógica de detectar colisão através do fecho do circuito. A ideia base do circuito é apresentada na Figura 80.

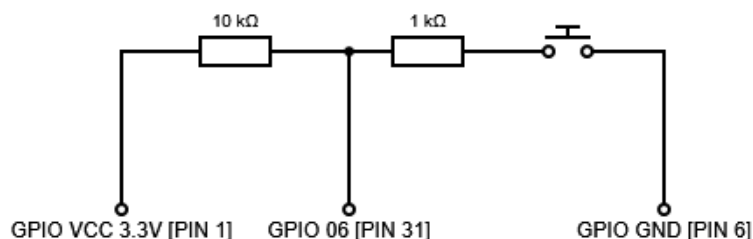


Figura 80 – Esquema do circuito MicroSwitch.

Relativamente à implementação, o teste inicial consistiu na configuração do pin de entrada e inicialização de uma variável para guardar o valor lido. O contacto é detectado no caso de ainda não existir uma leitura de tensão alta previamente e ocorresse no momento uma leitura positiva. Para intervalar leituras e obter uma maior credibilidade na detecção, um atraso foi adicionalmente introduzido.

Por fim, para incluir esta, ou qualquer outra, lógica de sensores, foi especificada uma interface sensor com o método detectar. Desta forma é

possível encapsular todo o processamento individual aos componentes sensoriais do agente robótico.

Módulo ultra-sons HC-SR04

O módulo de ultra-sons HC-SR04 consiste num sensor de distância de baixo custo e relativamente fácil de usar. Em termos de comunicação, existem quatro conexões: a alimentação (VCC), a ligação terra (GND), a ligação de activação (TRIG) e a ligação de resposta (ECHO).

O seu funcionamento consiste num processo simples. Primeiramente, é enviado um sinal de activação (TRIG). O HC-SR04 necessita de um pulso com duração de 10µs para activar, colocando o sensor em modo de detecção de alcance. São emitidos oito pulsos ultra-sónicos a 40kHz de forma a obter um eco de resposta. A chegada do eco é notificada através da sinalização pela ligação de resposta (ECHO). Com base no tempo de activação e tempo de captação do eco é possível determinar a duração. A mesma é utilizada para obter a distância até à superfície detectada. Como a duração engloba o envio e retorno do sinal sonoro, está implícito que a distância pretendida foi percorrida duplamente. A velocidade do som à temperatura média (343 m/s) considera-se suficiente para o propósito pretendido.

$$Velocidade = \frac{Distância}{Duração}$$

$$Distância = 34300 \times \left(\frac{Duração\ Total}{2} \right)$$

O alcance do sensor precisa igualmente de ser equacionado. O intervalo de alcance encontra-se entre os 2 cm e os 400 cm, traduzindo-se em leituras incorrectas qualquer valor calculado fora do intervalo. O próprio programa de medição incorporado no sensor também envia uma resposta por excedência de tempo, sinónimo de inexistência de eco, correspondente à distância máxima.

Este módulo opera e responde com uma tensão de 5V, tensão esta que, pode trazer problemas aos pins na leitura. A fonte usada corresponde ao próprio *Raspberry Pi* que disponibiliza um pin a 5V. Tal constitui um problema pois todos os pins de entrada encontram-se preparados para receber uma tensão 3.3V. Dada esta questão, um divisor de tensão foi enquadrado para obter a tensão de 3.3V desejada.

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2} \Leftrightarrow \frac{V_{out}}{V_{in}} = \frac{R_2}{R_1 + R_2}$$

Atribuindo um valor de resistência a uma das duas resistências do divisor, calculou-se a resistência restante necessária. Com esta determinação adicionou-se o divisor de tensão ao circuito dos sensores.

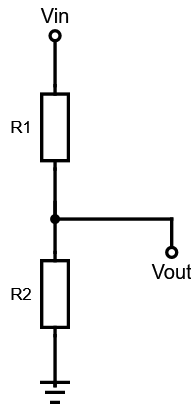


Figura 81 - Esquema do divisor de tensão.

Adicionalmente, é importante notar que o uso de divisor de tensão recorrendo a resistências pode não ser ideal, existindo outras soluções recorrendo, por exemplo, a transístores.

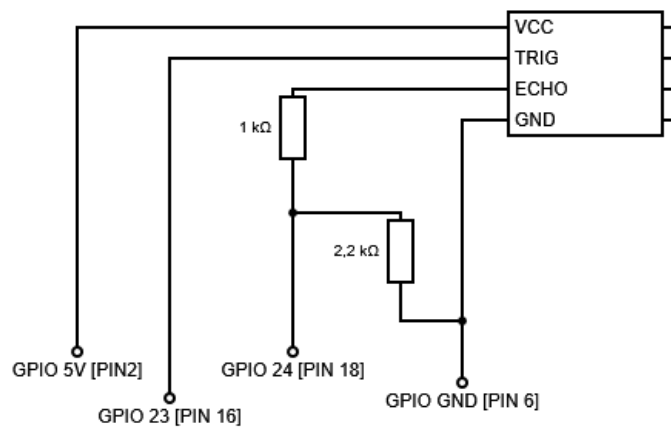


Figura 82 - Esquema de circuito de um módulo HC-SR04.

A linguagem de *Python* também não é muito eficiente ou precisa na leitura de medidas temporais com exactidão devido ao seu alto nível. Porém,

todos estes factores não invalidam a obtenção de resultados satisfatórios para o contexto de utilização pretendido.

Módulo luminoso TSL2561

O sensor de luminosidade TSL2561 é um sensor de luz digital e razoavelmente robusto na precisão. Essa característica torna-o ideal para utilização em diversas situações de luz, sendo mais preciso e permitindo o cálculo exacto de Lux com diferentes ganhos. A detecção da intensidade varia entre 0.1 a 40.000+ Lux e contém díodos para o espectro infravermelho e visível.

A interface de comunicação I2C (*Inter-integrated Circuit*) é utilizada para facultar acessos de leitura ao sensor. Este protocolo consiste num modelo de comunicação master-slave, onde circuitos integrados suportados utilizam dois canais. Cada barramento I2C é composto por dois sinais: SCL e SDA. O SCL consiste no sinal de relógio, gerado pelo master, e o SDA o de transmissão de dados. A maior vantagem deste protocolo consiste precisamente no uso exclusivo destes dois canais para estabelecer transmissões, identificando os slaves através de endereços únicos. Tal, permite a partilha do canal de dados até um máximo de 127 elementos, considerando um endereço a 7 bits.

Em termos do circuito, ambos os sensores partilham as linhas. No entanto, o facto destes sensores serem iguais traduz-se na utilização do mesmo endereço. Para ultrapassar este conflito, estes módulos possuem uma saída (ADDR) à qual é possível ligar a terra ou fonte, alterando o endereço e possibilitando o uso até três módulos no mesmo canal. Ligado à terra é definido o endereço de 0x29, ligado à fonte 3.3V (VCC) o endereço muda para 0x49 e, por fim, sem ligação adquire o endereço padrão 0x39.

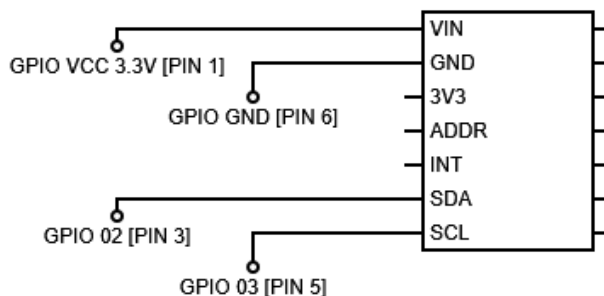


Figura 83 - Esquema de circuito de sensor TSL2561.

Finalizado o circuito, é importante configurar o protocolo I2C na unidade de processamento e verificar o reconhecimento dos endereços. Utilizando a linha de comandos do Raspberry Pi procedeu-se à activação do suporte I2C no *kernel*, este último recorrendo à configuração nativa existente no *Raspberry Pi*, e à instalação das ferramentas necessárias.

```
sudo raspi-config  
sudo apt-get install python-smbus  
sudo apt-get install i2c-tools
```

Para verificar todos os dispositivos ligados via I2C é executado o comando:

```
sudo i2cdetect -y 1
```

Com o reconhecimento dos sensores é assim possível aceder a estes e proceder à leitura dos parâmetros necessário ao cálculo da intensidade luminosa. Para uma maior robustez em relação a possíveis erros e complicações foi utilizada uma biblioteca com alguns dos métodos de acesso e de cálculo já implementados. A biblioteca *Adafruit_TSL2561* consiste numa adaptação para *Python* da mesma criada na linguagem *C++* em conjunto com a biblioteca *Adafruit I2C*, usada para as operações de escrita e leitura.

Com base nesta camada de abstracção, o funcionamento do sensor passa pela inicialização especificando o endereço e parâmetros possíveis, como o auto ganho, e proceder à chamada do método para calcular o Lux. Este, converte os valores em bruto fornecidos pelo sensor em unidades Lux.

Controlador Motores TB6612FNG

Os motores de corrente contínua necessitam de ser activados individualmente, mas sincronamente. O controlador, ou drive, de motor TB6612FNG permite satisfazer esse requisito e controlar dois motores DC a uma corrente constante de 1.2A (com pico de 3.2A). Com base nas ligações IN1 e IN2 é possível controlar o motor especificando um de quatro modos de operação: rotação no sentido do relógio, contra rotação, travagem e paragem. A entrada PWM (*Pulse Width Modulation*) serve para controlar a velocidade de rotação dos motores até a uma frequência de 100kHz.

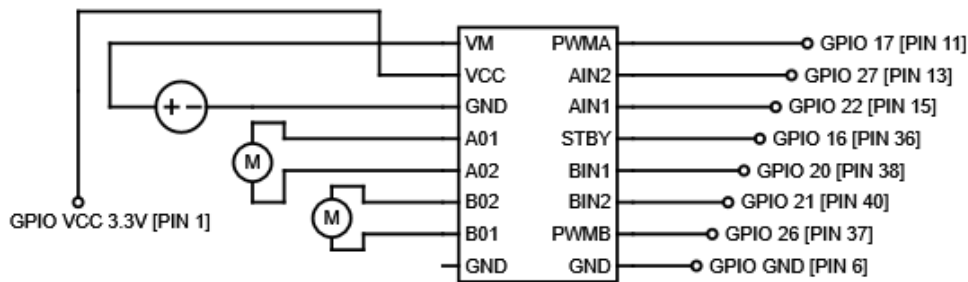


Figura 84 - Esquema do circuito do controlador e respectivos motores.

A implementação e criação dos movimentos é realizado de forma intuitiva através de sinalização por parte da unidade de processamento para o controlador. Sinalizando as entradas IN, STBY e PWM existentes é possível activar um dos modos de funcionamento descritos. A codificação de acções pode ser assim elaborada combinando a saída dos pins de acordo com a

Acção	Entrada / Pin						
	AIN1	AIN2	BIN1	BIN2	STBY	PWMA	PWMB
Rodar Esquerda	HIGH	LOW	LOW	HIGH	HIGH	HIGH	HIGH
Rodar Direita	LOW	HIGH	HIGH	LOW	HIGH	HIGH	HIGH
Avançar Frente	HIGH	LOW	HIGH	LOW	HIGH	HIGH	HIGH
Parar / Desligar	LOW	LOW	LOW	LOW	LOW	LOW	LOW

Tabela 12 - Codificação das acções em sinalização.

Enquanto as entradas estiverem configuradas com uma codificação, os motores executam esse modo de funcionamento, sendo sempre necessário recorrer à sinalização de paragem para cessar movimento.

Impressão 3D de suportes

O posicionamento e orientação dos sensores na estrutura do robô representa um desafio, uma vez que, sendo circuitos integrados, encontram-se desprotegidos e sem suporte. A solução encontrada residiu no uso das recentes técnicas de impressão 3D através de filamento plástico.

Para além de prototipagem e aquisição rápida, este tipo de impressão permite modelar qualquer modelo a três dimensões e proceder à sua concretização a baixo custo. Nesse sentido, recorreu-se ao domínio da internet para encontrar modelos de suportes adequados aos sensores. Tendo sido apenas encontrado modelos para o módulo *HC-SR04*, procedeu-se à criação dos restantes suportes em *Blender*. Os suportes criados foram os seguintes:

- Suporte para sensor luminoso TSL6125.
- Suporte de altura para módulo HC-SR04.
- Edição e adaptação de suporte de fixação para módulo HC-SR04.
- Base de suporte à unidade de processamento e respectivas fontes de alimentação portáteis (*powerbanks*).

A impressão 3D requer sempre um ficheiro específico que contenha os dados de impressão. Sendo a impressora usada um modelo do tipo *Prusa i3*, o *software Cura* foi utilizado para converter o ficheiro de modelação, criado no *Blender*, para um ficheiro de extensão “*gcode*” compatível com a impressora.

O *Cura* é particularmente útil pois, permite especificar os parâmetros de impressão como, a velocidade e densidade no preenchimento, além da espessura do filamento a ser usado na impressão. Estas especificações conjuntas permitem obter uma duração estimada do processo e qualidade do produto final.



Figura 85 - Peça 3D finalizada de suporte ao sensor HC-SR04.

Infelizmente, a impressão 3D não está longe de problemas e defeitos que possam ocorrer. Má aderência do plástico à superfície de impressão ou mudanças de temperatura no aquecimento da ponteira podem provocar defeitos ou até mesmo fracassar a impressão. A criação de peças pequenas, como os suportes de sensores, consegue obter resultados satisfatórios, sendo apenas necessário limar pequenas imperfeições para encaixar correctamente.



Figura 86 - Suporte impresso para o sensor TSL6125.

Porém, com a impressão do suporte da unidade de processamento, as dimensões, consideravelmente maiores, aumentam a probabilidade de falha devido ao tempo de impressão. Tendo falhado a impressão da mesma por várias vezes, optou-se pela impressão de uma simples base, de forma a aglomerar os elementos, em detrimento de uma estrutura completa.

Montagem do agente robótico

A montagem do robô seguiu o enquadramento especificado na Figura 32, fazendo uso de duas *breadboards* de dimensões reduzidas para criar as ligações. O esquema de ligações completo foi criado através da aplicação *fritzing*, sendo apresentado na Figura 88,

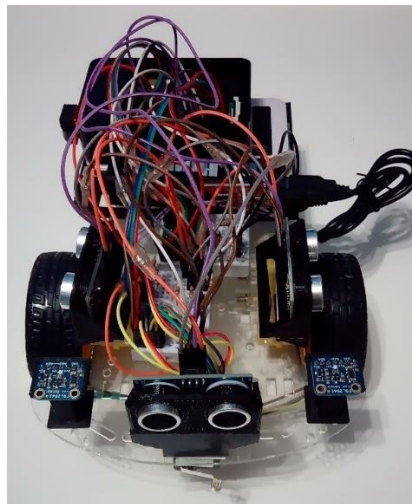
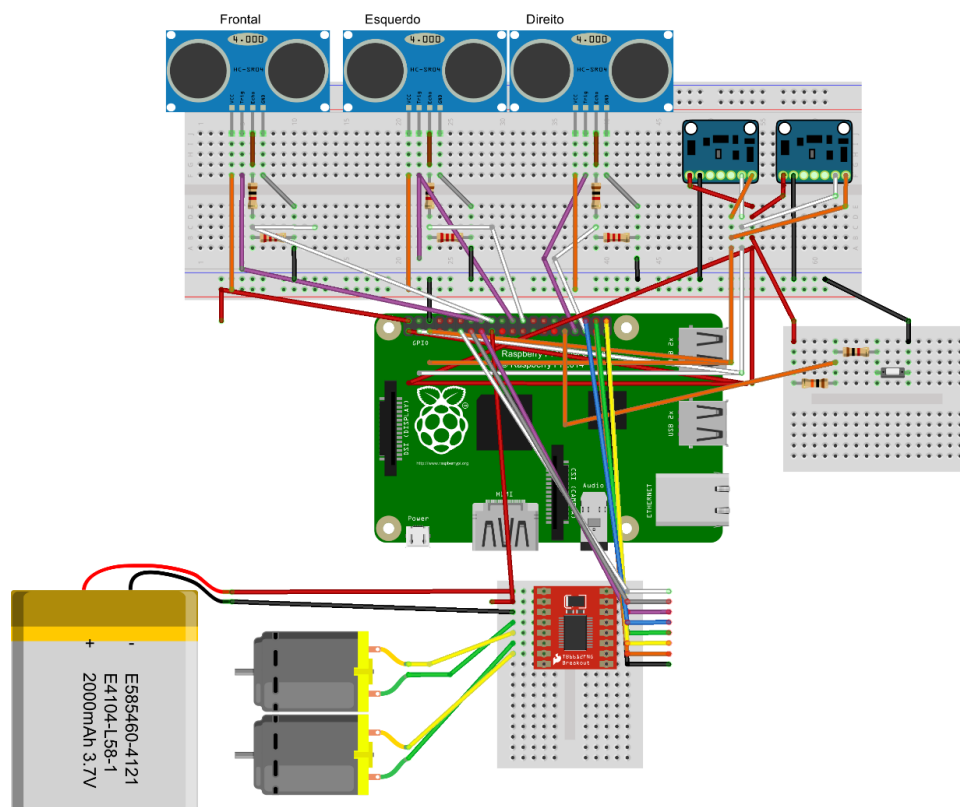


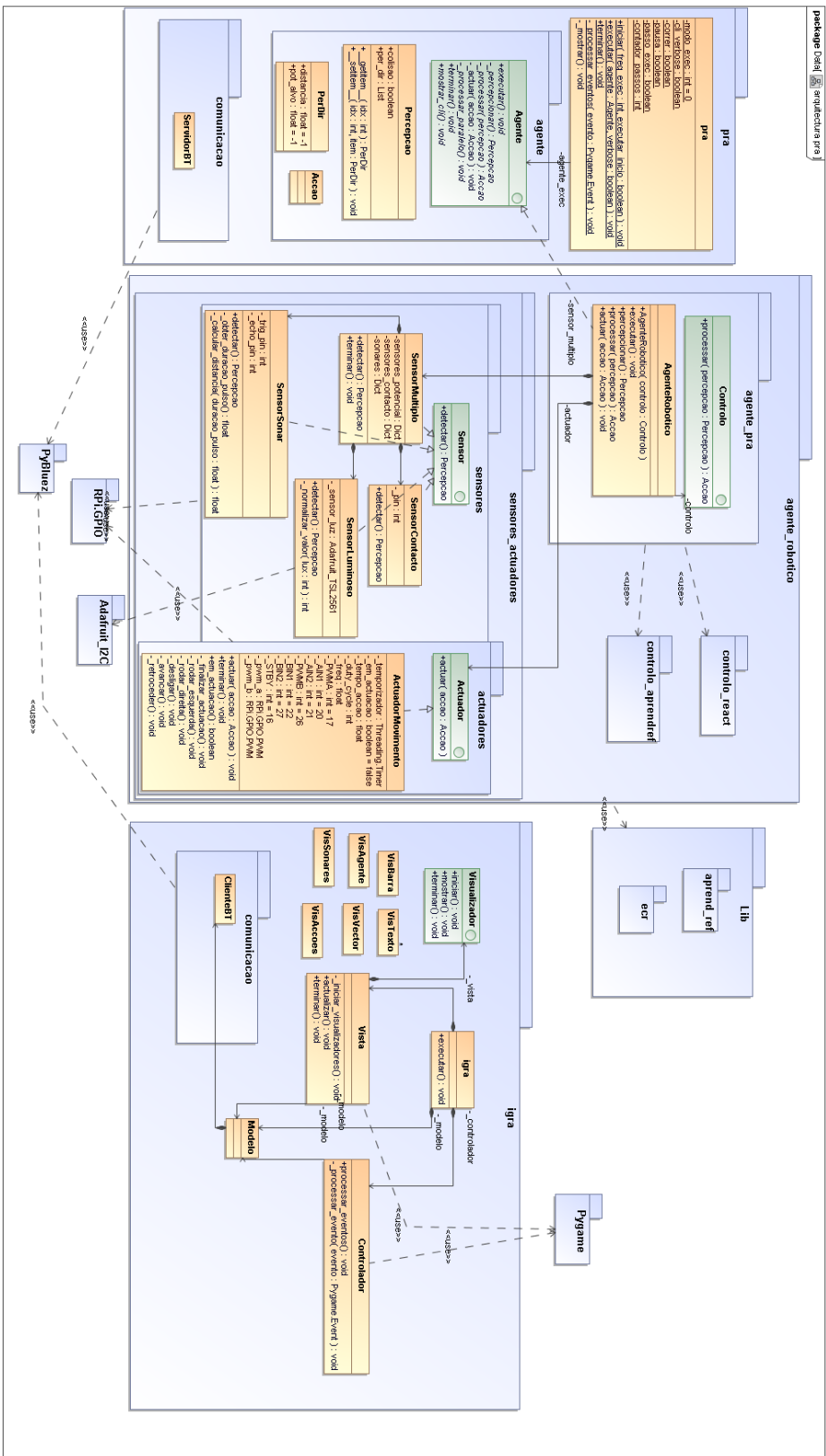
Figura 87 - Agente robótico finalizado.



fritzing

Figura 88 - Esquema de ligações completo.

Arquitectura PRA - Plataforma de Robótica de Agente



Apêndice C – Ferramentas

Neste apêndice são apresentados os recursos tecnológicos utilizados para a concretização do trabalho prático, nomeadamente a linguagem de programação *Python*, o módulo *Pygame* e a plataforma PSA.

Linguagem Python

Python é uma linguagem de programação lançada por *Guido van Rossum* em 1991. É uma linguagem de programação de alto nível, interpretada e orientada a objectos que permite trabalhar com rapidez e integrar sistemas de forma eficaz. A linguagem foi projectada com a filosofia de enfatizar a importância do esforço do programador em detrimento do esforço computacional, combinando uma sintaxe concisa e clara através dos recursos poderosos da sua biblioteca padrão, dando prioridade à legibilidade do código. A linguagem *Python* possui diversos módulos adicionais criados pela comunidade e concebidos para diferentes funcionalidades.

Pygame

O *Pygame* consiste numa biblioteca focada para o desenvolvimento rápido e simples de interfaces gráficas e jogos. Este módulo adiciona um conjunto de funcionalidades que permite criar todo o tipo de interfaces gráficas com recurso à linguagem *Python*.

De entre as funcionalidades encontram-se uma biblioteca completa de forma a criar toda a componente gráfica 2D, detendo métodos de leitura, manipulação de imagem, assim como métodos para desenhar todo o tipo de formas geométricas, que podem ser depois agregadas para criar elementos gráficos.

Ambiente de desenvolvimento

A observação do comportamento de um sistema autónomo inteligente é importante no sentido de avaliar o desempenho e resultados produzidos.

A Plataforma de Simulação de Agentes, ou PSA abreviadamente, é uma plataforma criada e disponibilizada pelo docente da unidade curricular. Esta plataforma permite observar o comportamento de um agente na sua evolução temporal para um determinado ambiente. Contendo várias vistas, como observação de campos de potencial e funções de valor com política comportamental, esta plataforma fornece um ambiente adequado a testes.

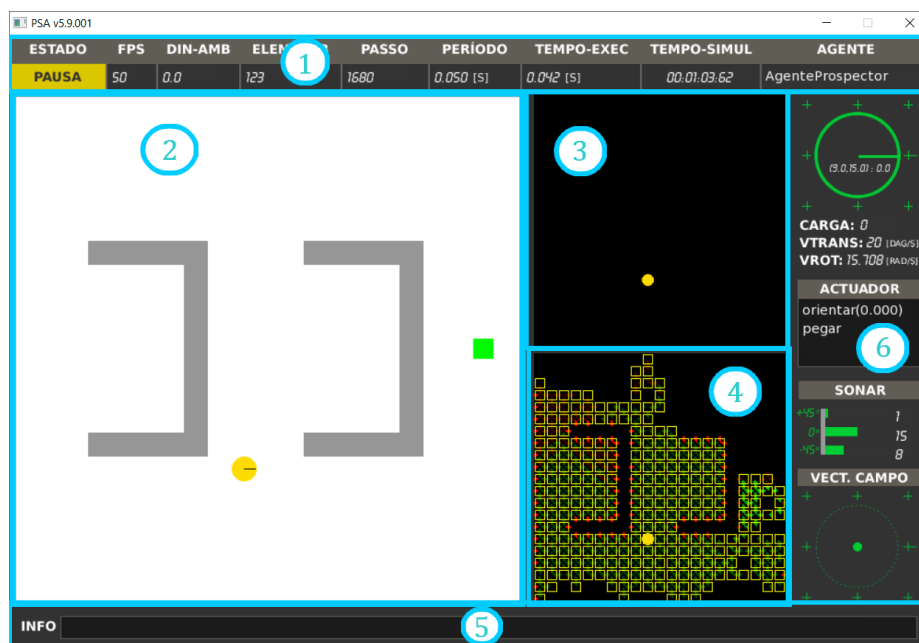


Figura 89 - Interface de visualização da PSA.

A interface da PSA é constituída por 6 componentes visuais:

1. Barra de informação da simulação;
2. Visualizador do ambiente;
3. Visualizador de percepção do agente;
4. Visualizador de modelo;
5. Barra de informação auxiliar;
6. Coluna de informação relativa ao agente;

Na perspectiva de desenvolvimento do trabalho as características da PSA são incrivelmente úteis uma vez que permite criar uma camada de abstracção, resultando num foco total em relação às arquitecturas e algoritmos desenvolvidos.