INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

ÁREA DEPARTAMENTAMENTAL DE ENGENHARIA DE ELECTRÓNICA E TELECOMUNICAÇÕES E DE COMPUTADORES

Licenciatura em Engenharia Informática e Multimédia

Inteligência Artificial para Sistemas Autónomos

Documento Síntese

2018/2019

ALUNOS:

39275, Ana Sofia Oliveira

ÍNDICE

1 ENQUADRAMENTO TEÓRICO 1

1.1 ARQUITETURA DE AGENTES 1

1.1.1 Ambiente 1

1.1.1.1 Ambiente discreto 1

1.1.1.2 Ambiente contínuo 2

1.1.1.3 Ambiente determinístico 2

1.1.1.4 Ambiente estocástico 2

1.1.1.5 Ambiente estático 2

1.1.1.6 Ambiente dinâmico 2

1.1.1.7 Ambiente totalmente observável 2

1.1.1.8 Ambiente parcialmente observável 2

1.1.2 Agente Reativo 3

1.1.2.1 Reacção Simples 4

1.1.2.2 Reacção Composta (Comportamento) 5

1.1.3 Agente Deliberativo 6

1.1.3.1 Mecanismos de Selecção de Acção 7

1.1.3.2 Procura em Espaço de Estados 9

1.1.3.3 Processos de Decisão de Markov 14

1.2 APRENDIZAGEM POR REFORÇO 15

1.2.1 Algoritmo Q-Learning 16

2 ENQUADRAMENTO PRÁTICO 17

2.1 AGENTE REACTIVO 17

2.2 AGENTE DELIBERATIVO DE PROCURA EM ESPAÇOS DE DADOS 18

2.2.1 Puzzle A 18

2.2.2 Puzzle B 18

2.2.3 Puzzle – Analise de Resultados 19

2.3 AGENTE DELIBERATIVO 20

2.4 AGENTE DELIBERATIVO BASEADO EM PROCESSOS DE DECISÃO DE MARKOV E APRENDIZAGEM POR REFORÇO 20

3 CONCLUSÃO 22

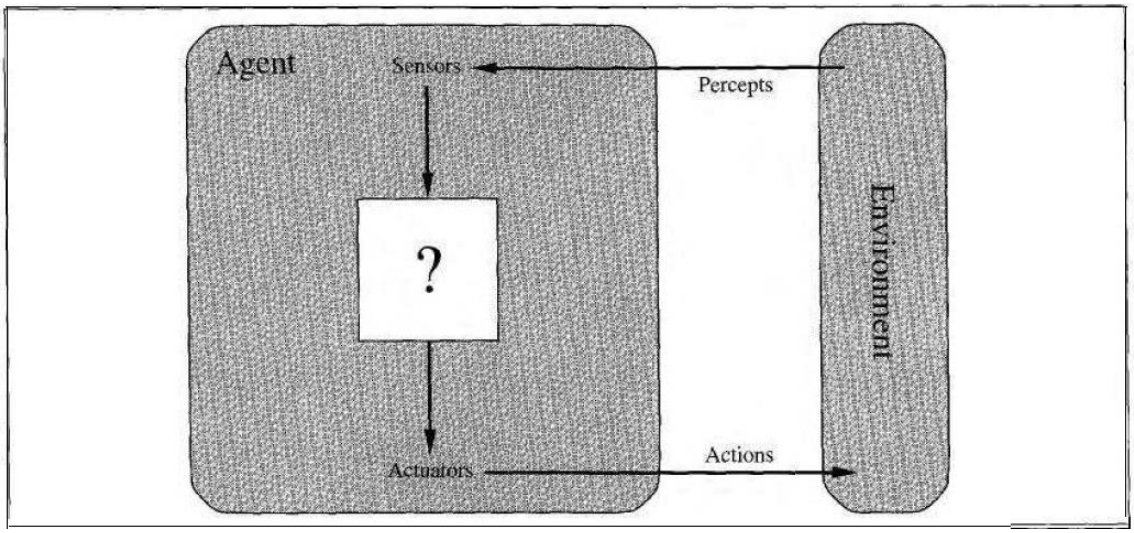
4 BIBLIOGRAFIA 24

1 Enquadramento Teórico

1.1 Arquitetura de Agentes

Um agente é uma entidade que funciona autonomamente num determinado ambiente, podendo ser visto como um perceptor do ambiente e um atuador no ambiente. A sua percepção é obtida através de sensores, enquanto que a sua actuação é executada através de actuadores.

Abaixo está ilustrada a arquitetura mais abstrata para qualquer agente. Dependendo do nível de funcionalidade exigido ao agente, esta arquitetura pode tornar-se mais completa e complexa.



Dependendo do tipo de agente, o módulo (?) será composto por diferentes partes.

1.1.1 Ambiente

O ambiente em que o agente opera pode ter diversas representações. Assim, é também importante o estudo das diversas variantes do ambiente.

1.1.1.1 Ambiente discreto

Um ambiente discreto é descrito como sendo um ambiente que contém um conjunto de estados finito. Por exemplo, num jogo de xadrez o conjunto de todas as configurações possíveis do ambiente é finito, pois descreve todas as possíveis configurações do tabuleiro.

1.1.1.2 Ambiente contínuo

Um ambiente contínuo representa um ambiente cujo conjunto de todas as configurações do ambiente não é finito, ou seja, podem sempre existir novas configurações de estado para o ambiente.

1.1.1.3 Ambiente determinístico

Um ambiente determinístico define a previsão de um sistema, ou seja, sempre que consigamos indicar o próximo estado apenas com a informação do estado corrente e a sua ação, conseguimos determinar o ambiente.

1.1.1.4 Ambiente estocástico

Um ambiente estocástico é contrário a um ambiente determinístico, ou seja, não conseguimos prever o próximo estado do ambiente apenas com a informação do estado corrente e da ação a tomar.

Por exemplo, um taxista sabendo que no ambiente não tem ninguém na estrada e cuja ação seria avançar não o pode fazer pois no instante (t+1) pode aparecer alguém na estrada.

1.1.1.5 Ambiente estático

Um ambiente estático, tal como o nome indica, é um ambiente que não sofre alterações ao longo do tempo.

1.1.1.6 Ambiente dinâmico

Um ambiente dinâmico sofre alterações ao longo do tempo, sejam por influência do próprio ambiente ou das ações do agente no ambiente.

1.1.1.7 Ambiente totalmente observável

Um ambiente é totalmente observável quando, na perspetiva do agente, o sensor deteta na integra a constituição do sistema.

1.1.1.8 Ambiente parcialmente observável

Um ambiente é parcialmente observável quando não é possível observar totalmente o ambiente através da observação do agente.

1.1.2 Agente Reativo

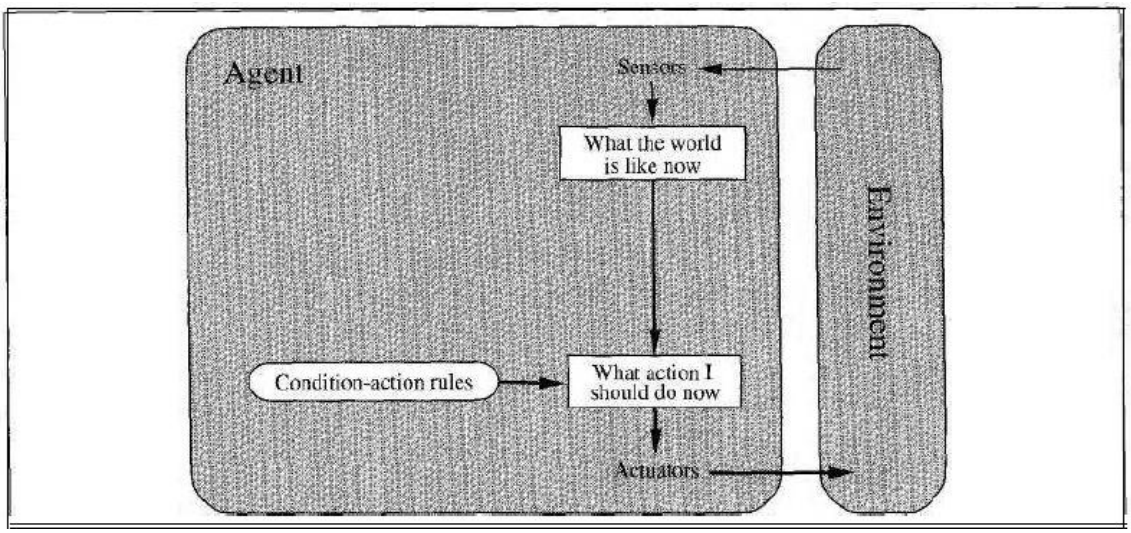
O agente reativo é aquele que percepciona o ambiente sob a forma de estímulo e produz uma ação associada a esse estímulo (comportamento). Por outras palavras, este agente perceciona o ambiente e executa uma ação apenas com a informação da perceção obtida.

> *"The choice of action at any given instant can depend on the entire percept sequence observed to date" – Artificial Intelligence: A modern approach*

Recorrendo à arquitetura geral ilustrada na figura abaixo, podemos concretizar o módulo (?) num módulo com regras de condições de acção. Estas regras de condições de acção serão responsáveis por definir as associações entre os estímulos percepcionados e as acções a serem tomadas, garantindo que o agente cumpre a sua função.

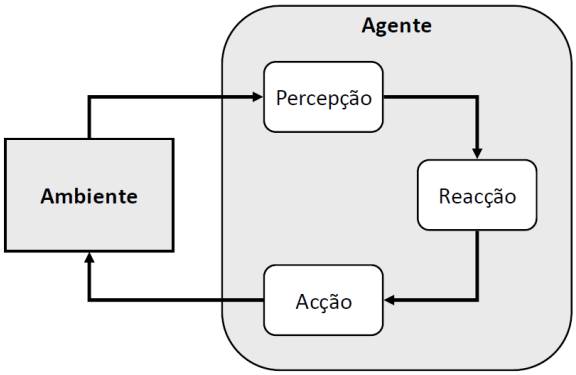
A estas associações denominamos reação/ comportamento do agente. Assim, mediante um determinado estímulo é gerada uma determinada ação.

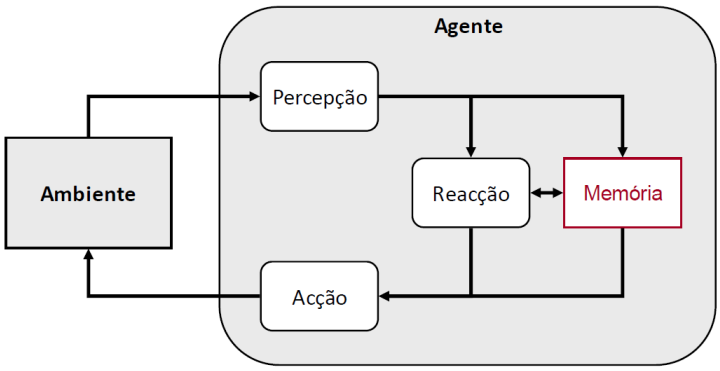
Complementando a arquitetura demonstrada na figura acima, obtemos a ilustração na abaixo.



O módulo (Regras de condições de ações) ficará então responsável por interpretar um estímulo percepcionado e indicar qual a ação a ser tomada. Para isso, existem 2 tipos de condições de ação, ou reações, que podem ser tomados.

Os agentes reativos podem também manter em memória o estado atual do ambiente, não dependendo assim apenas da informação da perceção para o instante (t). Quando esta informação é guardada internamente pelo agente, o esquema geral da arquitetura destes agentes evolui do apresentado na figura acima para o apresentado na figura abaixo.





1.1.2.1 Reacção Simples

A definição fisiológica determina que uma reação é uma resposta do organismo a um estímulo. Assim, podemos dizer que a reação é uma associação entre um determinado estímulo que irá gerar uma resposta.

Uma reação simples é uma associação direta o estímulo e a resposta.

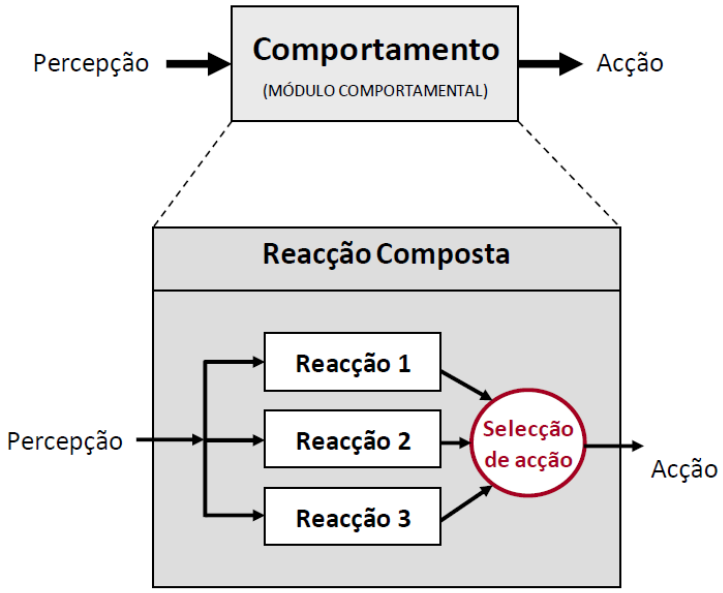
Por exemplo, para um robô de limpeza conseguimos construir a tabela abaixo que associa os estímulos percecionados às respostas a serem tomadas pelo agente.

Estímulo	Resposta
Sujo	Limpa
Limpo	Avança
Obstáculo	Desviar

A arquitetura de um agente cujas reações são diretas é a anteriormente apresentada na figura acima.

1.1.2.2 Reacção Composta (Comportamento)

Uma reação composta, ou comportamento, corresponde a um conjunto de reações simples sendo que produz apenas uma ação a ser aplicada ao ambiente, independentemente dos estímulos percepcionados.

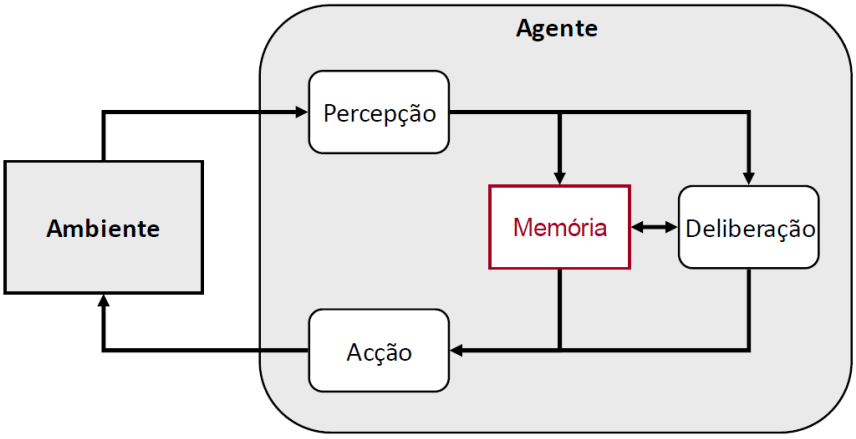


1.1.3 Agente Deliberativo

Agente deliberativo é aquele que possui pouca autonomia e possui um modelo simbólico explicito do ambiente. Baseado numa interpretação onde o agente faz parte de um sistema com conhecimento. A decisão das ações a tomar são baseadas num raciocínio lógico, seleção de ação, que irá ser explicado com maior pormenor mais à frente.

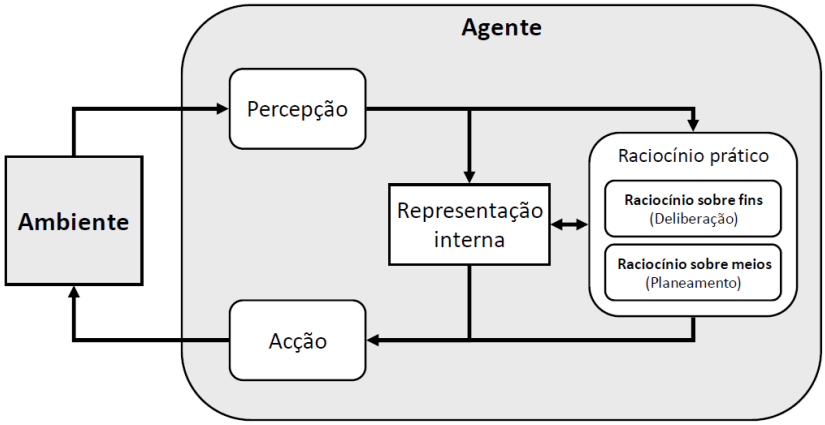
Estes agentes, após obterem a perceção proveniente do ambiente, recorrem a esta informação para manter atualizada uma representação interna, usualmente simbólica, do estado do mundo. É com este estado do mundo em conjunto com os objetivos do agente que são geradas as possíveis ações a executar pelo agente e selecionar as mais apropriadas a serem executadas por esse mesmo agente.

A figura abaixo ilustra o modelo geral de um agente deliberativo.



Na construção de agentes com uma arquitetura deliberativa, colocam-se questões importantes no processo:

- Como traduzir o mundo real para uma descrição simbólica e utilizar a perceção para manter essa estrutura simbólica atualizada? - Deliberação
- Como raciocinar utilizando essa informação simbólica de forma a decidir as ações a executar em cada instante? - Planeamento

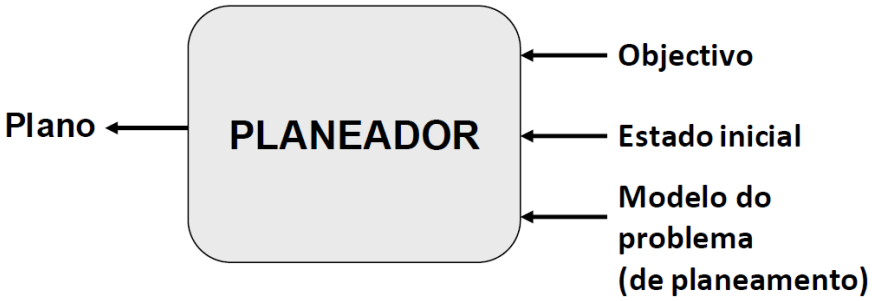


O raciocínio sobre os fins decide o que fazer com o propósito de chegar aos objetivos do agente, ou seja, delibera.

O raciocínio sobre os meios decide como fazer para que o objetivo seja cumprido, ou seja, planeia.

A arquitetura geral de um agente deliberativos encontra-se ilustrada na figura acima.

O modelo geral do planeador encontra-se ilustrado na figura abaixo.



1.1.3.1 Mecanismos de Selecção de Acção

Conforme descrito anteriormente, sempre que um agente possa ter mais que uma acção para o cumprimento da sua função é necessário selecionar a melhor ação a ser executada pelo mesmo. Assim, surge o conceito de mecanismo de seleção de acção.

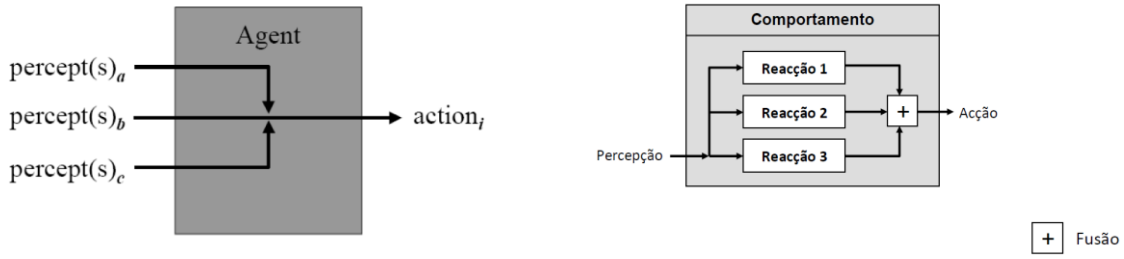
Estes mecanismos podem assumir diversas formas, sendo que o nosso estudo recaiu nos mecanismos de carácter paralelo, combinatório, hierarquico e com prioridade.

1.1.3.1.1 Ações paralelas

Este mecanisco, tal como o nome indica, consiste em gerar diversas ações. Apenas podemos usar este mecanismo quando o agente consigue executar mais que uma ação ao mesmo tempo. Cada uma das ações geradas corresponde a um estímulo anteriormente percepcionado. Assim, o agente percepciona diversos estímulos e gera as respostas correspondentes para cada estímulo, sendo que as suas ações iram ser executas em paralelo.

1.1.3.1.2 Combinação

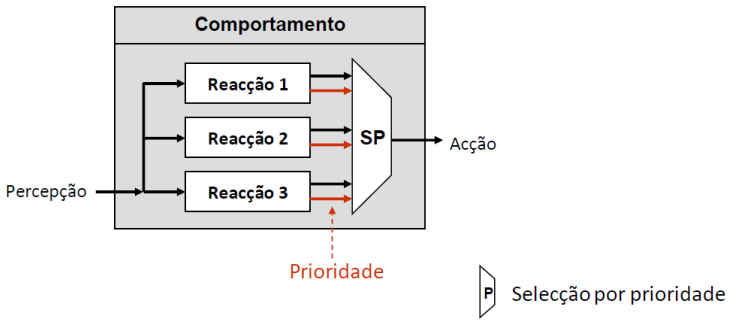
O carácter combinatório deste mecanismo é obtido através da aplicabilidade de operações aritméticas sobre as ações anteriormente geradas para cada reação (Fig. XXX). A combinação dessas ações sobre uma operação aritmética resulta numa única ação que irá ser executada pelo agente, apresentada na figura abaixo.



1.1.3.1.3 Por prioridade

Este mecanismo resulta de uma ordem de prioridade, ou seja, existem comportamentos mais prioritários que outros. A ação do comportamento com maior prioridade será aquela que irá ser gerada para o agente executar.

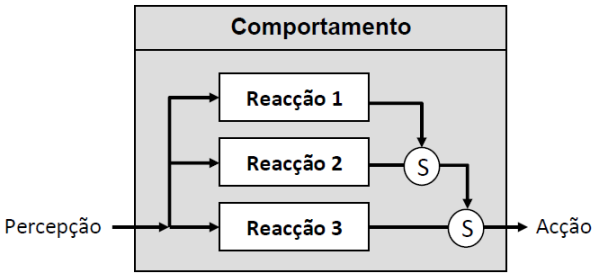
O esquema geral para o gerar desta ação está ilustrado na figura abaixo.



1.1.3.1.4 Por hierarquia

Este mecanisco resulta de um encadeamento hierarquico, ou seja, existem comporamentos com uma ordem hierárquica superior a outros.

Sempre que um comportamento com uma maior ordem hierárquica seja ativo a ação por si codificada será aquela que o agente irá executar.



1.1.3.2 Procura em Espaço de Estados

A procura em espaço de estados consiste em assumir que um agente capaz de executar uma ação que altere o estado corrente do ambiente. Assim, e tal como vimos, o agente gera planos de ação com base na transição entre estados. Para gerar um plano de ação necessita de efetuar uma procura num espaço de estados de forma a atingir o objetivo.

Para implementar este tipo de procura recorremos às seguintes definições:

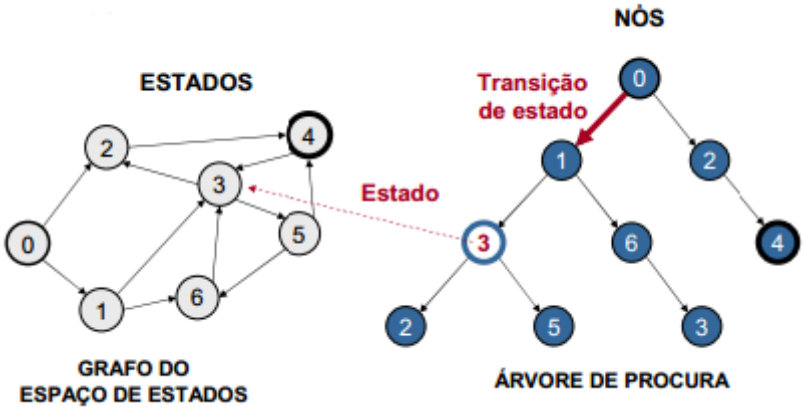
- Estado: Configuração do mundo possível no domínio do problema.
- Operador: Transformação de gera uma alteração de estado.
- Problema: Configuração que integra o modelo do mundo, objetivos, operadores e função de avaliação do custo.
- Solução: Conjunto de configurações capaz de atingir o objetivo através da aplicação dos operadores.

Esta procura pode ser representada como uma árvore de procura.

A árvore de procura é gerada pelas sucessivas aplicações de operadores sobre o estado corrente. O nó raiz da árvore corresponde ao estado inicial no domínio do problema.

Para facilitar a interpretação definimos os seguintes conceitos:

- Nó: Estado do sistema.
  - Estado: Configuração do estado.
  - Nó pai: Referencia para o nó que lhe deu origem.
  - Acão: ação que foi aplicada ao nó pai, dado origem ao nó corrente.
  - Custo do caminho: Custo do caminho desde o nó raiz até ao nó corrente.
  - Profundidade: profundidade na árvore de procura a que se encontra o nó corrente.
- Operador: Acão que gera uma transição de estado.
- Fronteira de Exploração: fronteira que dispões de todos os nós a serem explorados.



O algoritmo geral da procura em espaço de estados é dado pela sucessiva validação de nó objetivo à medida que se vai desenvolver a pesquisa na árvore de procura. Sempre que o nó corrente não seja um nó objetivo é necessário progredir na árvore de procura, ou seja, expandindo o nó corrente por aplicação do operador, onde iram ser gerados novos nós. Com a entrada destes novos nós conseguimos recursivamente avançar na pesquisa, no entanto a ordem de entrada dos nós gerados na fronteira de exploração representa um fator importante da concretização do algoritmo.

Assim, ilustramos o algoritmo de forma generalizada nas figuras abaixo.

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  inputs: percept, a percept
  static: seq, an action sequence, initially empty
          state, some description of the current world state
          goal, a goal, initially null
          problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then do
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

```
function TREE-SEARCH(problem, fringe) returns a solution, or failure

  fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
      if EMPTY?(fringe) then return failure
      node ← REMOVE-FIRST(fringe)
      if GOAL-TEST[problem] applied to STATE[node] succeeds
          then return SOLUTION(node)
      fringe ← INSERT-ALL(EXPAND(node, problem), fringe)

function EXPAND(node, problem) returns a set of nodes

  successors ← the empty set
  for each (action, result) in SUCCESSOR-FN[problem](STATE[node]) do
      s ← a new NODE
      STATE[s] ← result
      PARENT-NODE[s] ← node
      ACTION[s] ← action
      PATH-COST[s] ← PATH-COST[node] + STEP-COST(STATE[node], action, result)
      DEPTH[s] ← DEPTH[node] + 1
      add s to successors
  return successors
```

Os mais diversos mecanismos de procura em espaço de estados definem a forma como a fronteira de exploração é construída. Iremos, portanto, detalhar estes mecanismos.

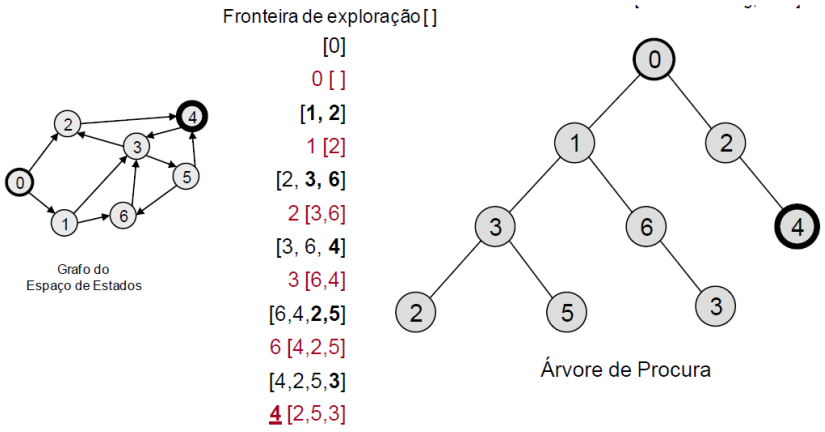
1.1.3.2.1 Mecanismos de Procura

1.1.3.2.1.1 Procura não informada

Este tipo de procura consiste na exploração do espaço de estado sem informação do domínio do problema.

1.1.3.2.1.1.1 Procura em Largura

Neste tipo de procura é dada prioridade à exploração dos nós com menor profundidade, ou seja, os primeiros nós a serem expandidos correspondem aos primeiros nós que deram entrada na fronteira de exploração em primeiro lugar (FIFO – First In First Out). Assim, garantimos que dada profundidade (d) todos os nós na profundidade (d-1) já foram explorados.

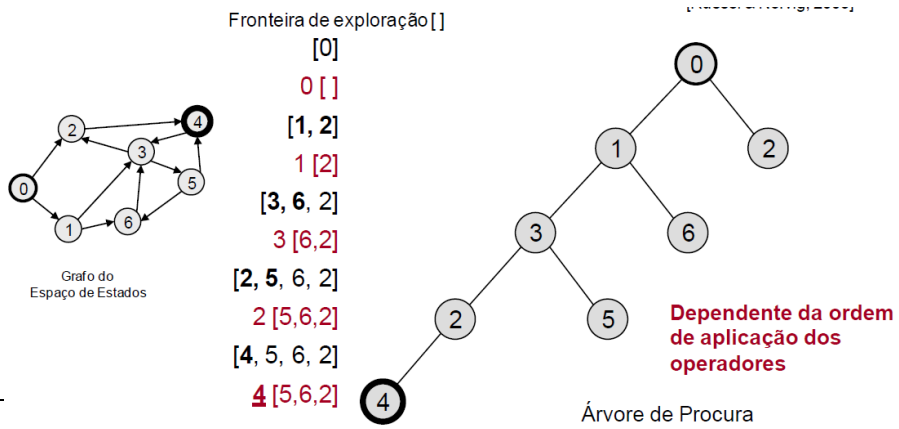


É um método de procura completo pois avalia todos os níveis de profundidade na procura garantindo que só termina quando encontra uma solução. Por este mesmo motivo é também ótimo pois vai sempre encontrar a solução garantindo que esta se encontra no menor nível de profundidade.

1.1.3.2.1.1.2 Procura em Profundidade

A prioridade dada a esta procura é sobre os nós com maior profundidade, ou seja, os últimos nós a serem gerados correspondem aos primeiros nós a saírem da fronteira de exploração (LIFO – Last In First Out). Desta forma, garantimos que quando seja expandido um segundo nó filho toda a ramificação do primeiro nó filho já foi explorada.

É um mecanismo de procura relativamente limitado por não consegue gerir ciclos de execução.

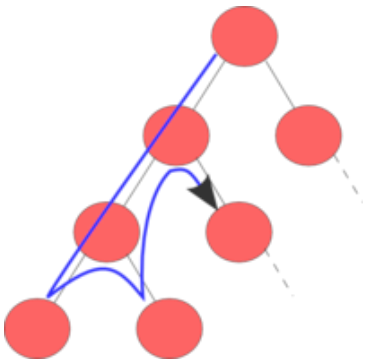


Não é um método de procura completo pois na eventualidade do nó expandido no instante (t) já ter sido expandido num instante anterior, este não é detetado, dando origem a loops. Por este mesmo motivo também não é um método ótimo.

1.1.3.2.1.1.3 Procura em Profundidade Limitada

Este mecanismo de procura é uma evolução do mecanismo de procura em profundidade com a nuance da limitação da profundidade de procura.

Através dele conseguimos garantir que a procura efetuada é completa sempre que a solução esteja a uma profundidade igual ou inferior à profundidade máxima de procura. Conseguimos garantir também que este mecanismo de procura é ótimo se o custo de cada transição for o mesmo, pois eliminando os loops irá encontrar a solução com menor custo.



1.1.3.2.1.1.4 Procura em Profundidade Iterativa

A procura em profundidade iterativa é uma variante mecanismo de procura de profundidade limitada, sendo que sempre que a solução não seja encontrada a uma profundidade (d) irá ser efetuada uma nova procura com profundidade (d+1), e assim sucessivamente.

Este mecanismo é, portanto, completo podendo não ser ótimo sempre que seja necessário um incremento à profundidade máxima da procura.

1.1.3.2.1.2 Procura Informada

Este tipo de procura consiste na exploração do espaço de estado com informação do domínio do problema para ordenação da fronteira de exploração. Esta informação está descrita na função f(n).

A função f(n) é uma função que avalia o custo do nó corrente, ou entre o nó inicial e o nó corrente ou entre o nó corrente o nó objetivo, heurística. A heurística estabelece uma relação válida de custo entre o nó corrente e o nó objetivo, sendo que para isso tem de ser otimista e admissível.

Uma heurística admissível é otimista, sendo sempre a sua estimativa de custo inferior ou igual ao custo efetivo mínimo. Em geral, esta é obtida através da remoção de restrições associadas ao problema, sem afetar o mesmo.

A heurística, para além de ótima e admissível, pode também ser consistente sendo que os valores de f(n) não irão diminuir ao longo do tempo. Assim, qualquer nó explorado irá permanecer no caminho ótimo pois os restantes caminhos terão um custo mínimo superior ou igual. Uma heurística consistente é também uma heurística admissível, mas uma heurística admissível não é necessariamente uma heurística consistente.

1.1.3.2.1.2.1 Procura Melhor Primeiro

Este mecanismo de procura organiza a fronteira de exploração apenas com a informação de custo entre o nó inicial e o nó corrente. Assim,

$$f(n) = g(n) , f(n) > 0$$

1.1.3.2.1.2.2 Procura Sôfrega

A fronteira de exploração deste mecanismo de procura é organizada pela informação do custo entre o nó corrente e o nó objetivo. Assim,

$$f(n) = h(h), f(n)>0 \text{ e } h(n) \text{ admissível}$$

Esta procura não considera o percurso já explorado, minimiza o custo local e obtém soluções sub-optimas.

1.1.3.2.1.2.3 Procura A*

A fronteira de exploração deste mecanismo de procura é organizada pela informação do custo do nó inicial até ao nó corrente e do custo entre o nó corrente e o nó objetivo. Assim,

$$f(n) = g(n) + h(n), f(n)>0 \text{ e } h(n) \text{ admissível}$$

1.1.3.2.1.3 Avaliação dos mecanismos de procura

Os mecanismos de procura podem ser avaliados nas seguintes categorias:

- Completo: Garante que caso exista solução, esta será encontrada
- Ótimo: Garante que se existirem várias soluções, a primeira solução encontrada é a melhor
- Complexidade:
    - Temporal: Tempo necessário para encontrar uma solução
    - Espacial: Memória necessária para encontrar uma solução
    - Computacional: Número de nós a expandir para explorar até uma profundidade m

	Largura	Custo Uniforme	Profundidade	Profundidade Limitada	Profundidade Iterativa	Bi-direccional
Completo	Sim (x)	Sim (x,y)	Não	Não	Sim (z)	Sim (z,w)
Tempo	$O(b^d)$	$O(b^{[C^*/\varepsilon]})$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Espaço	$O(b^d)$	$O(b^{[C^*/\varepsilon]})$	O(bm)	O(bl)	O(bd)	$O(b^{d/2})$
Óptimo	Sim (z)	Sim	Não	Não	Sim (z)	Sim (z,w)

Legenda:

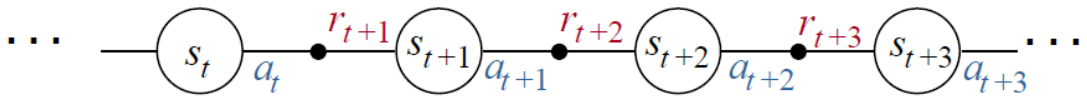
(x) – Completa se b infinito
(y) – Completa de custo de cada ramo > 0
(z) – Ótima se todos os ramos têm o mesmo custo
(w) – Se ambas as direções executam procura em largura primeiro
b – Fator de ramificação
d – Dimensão da solução
m – Profundidade da árvore de procura
l – Limite de profundidade
C* – Custo da solução ótima
Ɛ – Custo mínimo de uma transição de estado (Ɛ > 0)

1.1.3.3 Processos de Decisão de Markov

Os processos de decisão de Markov representam um processo de decisão sequencias em que, ao longo do tempo, são escolhidos estados distintos. Para transição existem ganhos ou perdas, representados na forma de recompensas. Estas recompensas nem sempre têm um efeito cumulativo.

Quando a previsão de um estado depende apenas do estado atual dizemos que este processo tem propriedade de Markov. Neste contexto temos a noção de:

- s, estado.
- a, ação.
- s', estado seguinte.



Estado num determinado estado, s, e executando a ação, a, temos uma determinada probabilidade de atingir o próximo estado, s', que nos é dada através da função T(s, a, s').

Igualmente, Estado num determinado estado, s, e executando a ação, a, temos uma recompensa que pode ser positiva ou negativa. Esta recompensa encontra-se explicita na função R(s, a, s'). O valor das recompensas é descontado ao longo do tempo, ou seja,

temos um fator multiplicativo (gama) que faz com que as recompensas do passado sejam menos importantes que as do presente.

A utilidade, U, é dada pelo efeito cumulativo das consequências das escolhas.

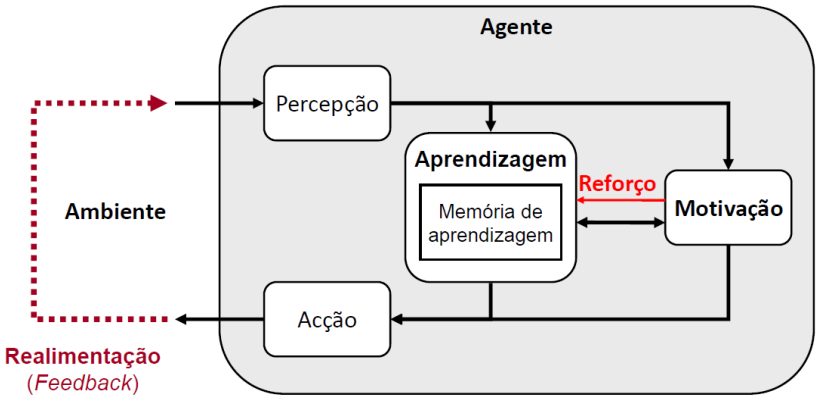
A política comportamental é o que define que ação se deve realizar em cada estado, e qualidade da política é medida pela utilidade esperada das ações geradas pela política. As transições e recompensas constituem o modelo do mundo.

O propósito é maximizar a recompensa total a longo prazo (uma política que maximiza a utilidade).

1.2 Aprendizagem por Reforço

A melhoria de desempenho de um determinado agente pode ser otimizada através do processo de aprendizagem.

O agente não possuir nenhuma representação do ambiente que o rodeia, este vai ter de o explorar e classificar com base na recompensa.



Na aprendizagem por reforço o agente aprende através da interação com o ambiente de uma forma continua e mantendo em memória as consequências que os comportamentos geraram.

O modelo do mundo é desconhecido pelo agente e é através da exploração que o agente aprende que ações dão mais valor. O agente passa a ter motivação para a realização de determinados comportamentos, e a tomada de decisão ocorre a partir da avaliação das opções possíveis de ação.

A motivação do agente é maximizar a recompensa a longo prazo. O agente pode ter uma vertente de aprendizagem de exploração do ambiente que tem por objetivo explorar todos os estados possíveis de modo a ter mais experiência ou pode usar uma estratégia de aproveitamento onde usa as suas aprendizagens para obter o máximo de recompensa.

O agente restringe-se às ações que conhece, é chamada de política sôfrega ou greedy.

1.2.1 Algoritmo Q-Learning

Escolhe a ação futura tendo em conta o próximo estado que produz o melhor Q seguinte.

O gama é um fator de desconto temporal e alfa um parâmetro de esquecimento.

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

2 Enquadramento Prático

2.1 Agente reativo

No decorrer do cumprimento dos objetivos propostos para este trabalho prático, foi crescentemente notável a importância de um software bem pensado e bem desenhado como forma de conseguir estender as funcionalidades do programa de forma rápida, simples e funcional, sem ter de proceder a alterações estruturais do projeto. Foi também notória a relação entre a boa estruturação e o reduzido número de linhas de código necessárias para conceber uma aplicação. Recorrendo a uma boa engenharia de software é assim possível simplificar o processo de programação, reduzindo a complexidade do código.

Passa-se assim a explicitar as conclusões retiradas relativamente aos problemas aos quais fomos expostos aquando a realização do primeiro trabalho.

Após conseguido o pedido no primeiro ponto verificou-se o bom funcionamento do agente na procura dos alvos, contudo, verificou-se ainda a baixa eficiência dessa procura. Isto verificou-se devido ao facto de o agente apenas percecionar alvos situados em frente ou nas direções que fazem 45º com a sua frente. Segundo esta arquitetura é comum o agente vaguear enquanto ainda existem diversos alvos no ambiente, não sendo este o modo mais eficiente de os recolher.

Perante este problema, passou-se assim ao cumprimento do segundo ponto, com o qual se acrescentou a reação de seguir campos de potencial. Sendo que seria de esperar que a eficiência do sistema aumentasse na recolha de alvos, na medida em que o agente teria algo que o aproximaria do alvo, a realidade experimentada não correspondeu com a expectativa. Com a nova implementação deparamo-nos com a existência de zonas ótimas nas quais o agente ficava retido. Ao entrar nestas, era impossível ao agente proceder à procura de alvos, dado que não conseguia deslocar-se para fora das zonas em questão. Em conclusão, era comum o agente permanecer estático antes da recolha de todos os alvos, bloqueado numa posição.

Foi na medida de contornar a situação descrita que se implementou o ponto 3 presente no enunciado do trabalho. Ao implementar sobre o agente a noção de memória, é-lhe possível sair de locais onde se recorde ter estado. Assim, quando se encontra numa zona ótima, verificando a existência dessa posição em memória, o agente procura deslocar-se para fora da mesma.

Embora tenha sido implementado com sucesso os objetivos do trabalho, verificou-se em alguns casos situações indesejadas, nomeadamente a ineficiência da fuga de ótimos locais em algumas situações. Embora se verifique a saída de zonas ótimas, quando em redor desta zona existem objetos que levem o agente de regresso às mesmas, este acaba por tomar um movimento cíclico do qual não consegue sair. Denotou-se ainda a importância do ajuste da longevidade da memória ao problema em questão, de modo a otimizar a relação necessidade de memória, complexidade espacial.

2.2 Agente deliberativo de procura em espaços de dados

Sendo que no âmbito do segundo trabalho prático foram desenvolvidos métodos de procura em espaços de estados, demonstra-se importante a análise dos diferentes mecanismos de procura tendo em conta:

- **O custo da solução;**
- **A complexidade temporal** (tempo necessário para encontrar a solução);
- **A complexidade espacial** (memória necessária, considerando-se a fronteira de exploração e os estados explorados em memória).

Para tal apresentam-se de seguida tabelas demonstrativas dos resultados obtidos.

2.2.1 Puzzle A

			Complexidade Espacial	
Método de Procura	Custo	Complexidade Temporal	Fronteira de Exploração	Mem Estados Explorados
Largura	14	0,4	2401	6836
BF- Custo Uniforme	14	0,41	2401	6836
Prof Limitada **	15000	6,86	11357	104955
Prof Iterativa**	5002	3,26	3859	57684
BF- Sôfrega	18	0,02	40	92
BF- A*	14	0,03	57	139

2.2.2 Puzzle B

			Complexidade Espacial	
Método de Procura	Custo	Complexidade Temporal	Fronteira de Exploração	Mem Estados Explorados
Largura	26	16,7	24050	172803
BF- Custo Uniforme	26	17,43	24050	172803
Prof Limitada **	15000	14,1	11357	153720
Prof Iterativa**	5002	5,48	3859	86683
BF- Sôfrega	56	0,16	357	895
BF- A*	26	1,26	2276	6099

Primeiramente há que considerar que apenas os métodos de procura de Profundidade Limitada e Iterativa não são completos. Dado que é objetivo obter valores que permitam comparar os diversos mecanismos, foi de nossa responsabilidade considerar limites de profundidade de procura para os quais fosse possível obter solução. Relativamente aos resultados obtidos, todos eles foram expectáveis.

2.2.3 Puzzle – Analise de Resultados

Para os testes efetuados com o problema Puzzle, verificaram-se soluções ótimas para os seguintes mecanismos de procura:

- Largura (devido ao facto de o custo ser unitário, logo, ao dar prioridade à expansão nós de menor profundidade, garante-se o menor custo);
- Custo Uniforme;
- A*.

Sendo o custo unitário para o problema Puzzle, verificou-se que a procura em largura teve comportamento idêntico à procura de Custo Uniforme, dado que nesta situação ambas garantem que são explorados os nós de menor custo. Há a apontar a diferença entre a complexidade temporal, devendo-se o menor valor da procura em largura possivelmente devido ao facto de dispensar o cálculo do custo do percurso percorrido.

Comparando os mecanismos de procura referidos no parágrafo anterior com a procura A*, verificou-se nom esta última quer menor complexidade quer temporal quer espacial. Deve-se este facto ao uso da função heurística que direciona a procura rumo à solução de menor custo, através da ponderação tanto do custo do percurso percorrido, com a estimativa do custo do percurso a percorrer. Assim, evita-se a expansão de nós que nos levem a percursos que terão maior custo que o ótimo.

Considerando a procura sôfrega, comparativamente com os outros mecanismos de Procura Melhor Primeiro, verificou-se uma menor complexidade temporal e espacial, em detrimento do custo. O próprio nome é indicativo deste resultado, sendo que tem como função de avaliação apenas a estimativa do custo de atingir o alvo. Faz sentido assim que explore menos estados até atingir o objetivo, demorando menos tempo nessa procura. Deste modo é também lógico que o custo seja superior dado que para cada estado a explorar não se pondere o custo tido para o atingir.

Debruçando-nos no presente parágrafo na análise dos resultados resultantes do uso dos métodos de procura em profundidade verificámos o maior custo, complexidade temporal e espacial deste. Poderia facilmente intuir-se este resultado, sendo que para que este mecanismo de procura tivesse melhores resultados a solução teria de se encontrar no primeiro ramo a ser explorado. Não sendo assim, despende-se imenso tempo e memória com a exploração de ramos sem solução, cada qual atingindo níveis de profundidade enormes.

Podemos, contudo, verificar os melhores resultados da pesquisa em Profundidade Iterativa comparativamente com a Profundidade limitada. Sendo que com o mecanismo referido primeiramente são efetuadas pesquisas de Profundidade Limitada com limite menor, é óbvio que o percurso obtido tenha menor custo. Intui-se também que a complexidade espacial seja menor, dado que apenas no pior dos casos se atingiria o limite de profundidade a explorar. Relativamente à complexidade temporal, esse valor depende da quantidade de pesquisas que são descartadas por falta de solução para a profundidade respetiva. No nosso caso, verificou-se bastante frutuoso utilizar a pesquisa em Profundidade limitada em detrimento de iterativa, dado que o limite de exploração desta última era enorme em relação à profundidade na qual existiam soluções.

Relativamente à complexidade destes tipos de procura e à importância da escolha de limites de exploração adequados, torna-se relevante referir que ao tentar efetuar a pesquisa em Profundidade limitada com limite 29596 o mecanismo demorou cerca de 2

horas a percorrer o espaço de estados sem, contudo, encontrar solução, tendo sido nossa opção abortar a procura.

2.3 Agente Deliberativo

Tendo atingido todos os objetivos propostos no enunciado do segundo trabalho prático com sucesso, estamos neste momento aptos a tirar diversas conclusões, algumas das quais já expostas sumariamente nos subtópicos anteriores.

Primeiramente há que ter em consideração o aumento da eficiência na busca de alvos do agente desenvolvido aquando a realização deste trabalho prático, comparativamente com o trabalho anterior. Denota-se, no entanto, o aumento da carga computacional da arquitetura deliberativa implementada. Foi assim necessário sacrificar espaço de memória e tempo de processamento a fim de obter um agente com uma capacidade deliberativa que permitisse uma melhor recolha de alvos em relação ao anterior agente reativa.

Tendo agora em conta a arquitetura deliberativa desenvolvida, poder-se-ia diminuir ainda a carga computacional do agente implementado, dado que para o ambiente estático no qual opera não torna necessário o agente ser cauteloso, denotando-se a necessidade de reconsiderar apenas quando o agente recolhe algum alvo. Deste modo, aumenta-se a complexidade tanto espacial como temporal em buscas de novas soluções quando a anteriormente encontrada é também válida.

Considerando os métodos de procura implementados, verificamos também que a eficiência e a eficácia do agente dependiam preponderantemente destes. Dados os resultados obtidos nos testes de mecanismos de procura, quer para a resolução de puzzles, quer para a busca de alvos, consideramos que os primeiros mecanismos de procura implementados, neste contexto, são meramente académicos, e que foram implementados por constituírem a base de outros mecanismos mais complexos, na medida em que tanto o peso computacional por eles requeridos como o tempo despendido na procura são bastante elevados, comparativamente com outros mecanismos posteriormente desenvolvidos. Quanto a este aspeto referíamo-nos aos métodos de procura em largura e profundidade. Denotou-se assim o benefício do uso de procuras informadas, assim como o aumento da eficiência através da utilização de heurísticas, denotando-se, contudo, a necessidade de ter conhecimento acerca do domínio do problema. Sempre que sejam de possível implementação, estas últimas pesquisas vêm de certo modo descartar o uso de outros métodos de procura não informada, dado que diminuem tanto a complexidade espacial como temporal.

Verificou-se assim em última instância a importância do uso de heurísticas como caminho para encontrar percursos de resolução de problemas em curtos intervalos de tempo, com relativamente pequena complexidade temporal e conseguindo soluções ótimas ou aproximadamente ótimas.

2.4 Agente deliberativo baseado em processos de decisão de Markov e aprendizagem por reforço

Comparando o mecanismo de procura de alvos recorrendo a processos de Decisão de Markov, em relação ao recurso a procuras em espaços de estados, verifica-se que com o primeiro não se torna necessário esperar pelo traçar duma rota de busca de alvos, contudo, a movimentação do agente torna-se bastante mais lenta, devido ao facto de

estarem envolvido no processo de recolha de alvos cálculos pesados computacionalmente na medida em que se procura maximizar a recompensa a longo prazo e há a necessidade de calcular a recompensa total obtida a longo prazo.

Relembra-se ainda que num modelo PDM é necessário conhecer-se o modelo do mundo e a função de recompensa que tem como imagem a recompensa associada a dois estados e a ação entre eles. Foi assim bastante lucrativo implementar aprendizagem por reforço, na medida em que nos pudemos libertar da necessidade de conhecer A priori quer o mundo, quer a função de recompensas. Com a aprendizagem por reforço, tornou-se assim possível ter como ponto de partida um ambiente completamente desconhecido. Deste modo o agente é essencialmente reativo, utilizando experiência como suplemento ou em substituição de modelos predefinidos. Assim, à medida que o agente toma determinadas ações, aprende e atualiza o valor de estado e o valor de estado-ação com a conhecimento adquirido após a tomada da ação, de modo a em iterações futuras sobre esse mesmo estado poder fazer a optar pela melhor ação dadas as experiências já vividas.

Verificou-se ainda o caracter exploratório do agente, através da implementação da Política e-greedy, na medida em que este não se cinge apenas aos conhecimentos já adquiridos nem somente à exploração de novos rumos. Deste modo, à medida que se itera sobre o mundo, é possível a aproximação ao caminho ótimo, devido ao carácter exploratório do agente, tornando-se possível a saída de Mínimos/máximos locais. Sendo que à medida que o agente explora o mundo o seu conhecimento sobre o mesmo é crescente, logo a necessidade de o explorar decrescente, seria uma boa prática diminuir gradualmente o valor de épsilon.

Relativamente aos algoritmos implementados, SARSA e Q-Learning, com o ambiente desenvolvido não verificámos diferenças significativas. Sabemos, contudo, que o algoritmo Q-Learning maximiza a recompensa esperada ao seguir para determinado estado, comparativamente com o SARSA. Contudo o primeiro é menos cauteloso relativamente à possibilidade de executar uma ação que não seja a esperada, dado que não considera a possibilidade de infortunadamente cair num local de recompensa muito prejudicial. Sendo assim o algoritmo SARSA mantém-se a maior distância dos locais cuja recompensa é menor, enquanto o Q-Learning é menos cuidadoso.

Neste caso torna-se lucrativo usar-se o SARSA, pois a probabilidade de o agente fazer a ação determinada pelo software quando o agente tem experiência é de 100%, não havendo a possibilidade de infortunadamente fazer a ação indesejada e obter uma recompensa indesejada. Em outros casos há que ponderar o algoritmo a utilizar consoante a aplicação do sistema.

3 Conclusão

No término deste trabalho podemos considerar a necessidade de existência dos quatro tipos de agente implementados dado que, embora uns atinjam o objetivo de modo mais eficiente que outros, nenhum é substituível. Cada um é assim aplicável a contextos distintos, tendo em consideração o hardware e software disponíveis, assim como o intervalo de tempo dispensado à resolução do problema, a necessidade de otimização da solução, a possibilidade de guardar memória e ainda o conhecimento que se tem do mundo.

É notável assim a grande distinção de ramos de aplicação de cada agente, sendo que cada um requer diferentes recursos e insere-se em diferentes contextos, os quais se enumeram de seguida:

- Agente Reativo:
  - Agente sem memória;
  - Sem representações internas do mundo;
  - Respostas fixas e predefinidas aos estímulos;
  - Bastante propensos a ótimos locais;
  - Talhados para cumprir objetivos simples, sem necessidade de grandes recursos;
  - Incapaz de produzir solução ótima;
  - Fraco desempenho.
- Agente Deliberativo:
  - Capazes de ponderar sobre opções e delinear planos;
  - Necessitam de representação interna do mundo;
  - Necessidade de conhecer o mundo A priori;
  - Agente com memória;
  - Maior complexidade espacial e temporal * (adaptáveis ao domínio do problema);
  - Capaz de gerar solução ótima;
  - Melhor desempenho *.

    *Comparativamente com agentes reativos.
- Processo de decisão Sequencial:
  - Necessitam de representação interna do mundo;
  - Necessidade de conhecer o mundo A priori e ações possíveis;
  - Necessidade de conhecimento dos modelos de transição e recompensas;
  - Previsão dos estados futuros dependente apenas do estado presente;
  - Elevado peso computacional dedicado a cálculos;

    - Problemas com a dimensão do espaço de estados.
- Processo de adaptação e aprendizagem:
    - Pode não existir conhecimento do mundo A priori nem a função de recompensa;
    - De relevante importância a combinação das vertentes explorar e aproveitar;
    - Problemas com a complexidade de espaço de estados e tempo de convergência.

No termino da unidade curricular estamos, portanto, aptos a, considerando os fatores enumerados anteriormente, os recursos e as necessidades de cada problema, implementar a solução mais vantajosa das enumeradas anteriormente.

4 Bibliografia

- [Russel & Norvig, 2003] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 2nd Edition, Prentice Hall, 2003.
- [Anabela Simões & Ernesto Costa] Inteligência Artificial, 2008.
- [Sérgio Guerreiro] Introdução à Engenharia de Software, 2015.