1.Introdução

O terceiro trabalho prático tem por objetivo o estudo e concretização de agentes inteligentes.

O agente irá ser testado num meio com três tipos de elementos o agente, os alvos, e os obstáculos. O objetivo é o agente recolher os alvos sem colidir com os obstáculos.

2.Desenvolvimento

Para realização do trabalho consiste em quatro concepções diferentes de implementação do agente.

- Uma concepção de um agente reativo.
- Uma concepção tendo por base raciocínio automático com base em procura espaço estados, que minimiza a distância para os alvos.
- Uma concepção que tem como base os processos de decisão de Markov.
- Uma concepção tem por base os mecanismos de aprendizagem por reforço, capaz de aprender pela experiência.

2.1 Agente prospetor

A arquitetura implementada tem por base um agente prospetor. Este agente prospetor é um agente genérico que executa ações genéricas. O algoritmo de funcionamento deste base do agente é receber uma percepção, que é processada e posteriormente é feita uma atuação sobre a mesma. Em termos de implementação é concretizado pela classe AgenteProspetor que possui o método executar que executa o algoritmo referido anteriormente. Esta classe possui ainda alguns métodos privados que concretizam o algoritmo do agente. Este algoritmo está representado pelo seguinte diagrama.

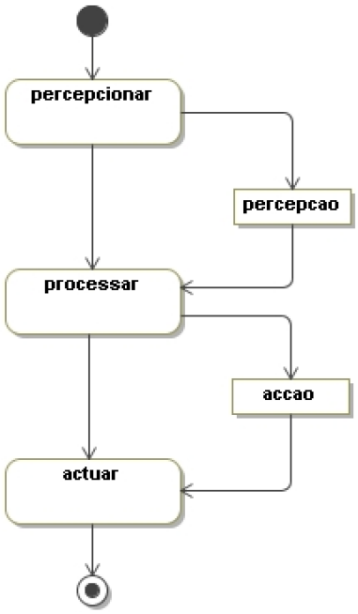

Figura 1 - Algoritmo base agente reativo

Na implentação foram utilizadas algumas funcionalidades da biblioteca psa para realização do mesmo algoritmo. A classe possui um atributo controlo, o controlo é uma interface que será utilizada para representar as várias concepções através da função processar.

2.2. Agente reativo

2.2.1 Controlo Reativo

A classe ControloReact é um controlo, este controlo possui a função processar que realiza a conexão entre percepções e a ações. A classe utiliza um comportamento, este comportamento é uma interface que faz a abstração entre o controlo e o comportamento do agente reativo feito pela biblioteca ecr.

2.2.2 Biblioteca ecr

O agente reativo desenvolvido é feito de acordo com uma arquitetura de subsunção, sendo um agente sem memória e onde o comportamento é dado pelo forte acoplamento entre percepção-reação. Esta implementação está depende das capacidades sensoriais e das

caracterísiticas do ambiente.

Na biblioteca existem duas classes que estendem de comportamento, a classe Reaccao e a classe ComportComp. A classe Reaccao é o comportamento responsável por comportamentos de deteção de estimulo e geração de resposta. Esta classe possui ainda a função ativar do Comportamento que devolve uma resposta a partir de uma percepção, utilizada na classe ControloReat.

A classe ComportComp define a seleção de ações do agente. Esta classe possui uma lista de comportamentos que serão organizados pelo tipo de seleção de ação. Organização esta implementada por um seletores de ações através de uma função abstrata. Visto que a classe também é um comportamento também possui o método ativar, que a partir da lista de comportamentos retirada a resposta de acordo com o tipo de seletor de ação.

A classe de seleção de ação são a Hierarquia e Prioridade que estendem de ComportComp e implementão a organização pela função selecionar resposta. A hierarquia organiza os comportamentos numa hierarquia fixa. A hierarquia é ordenada por ordem de chegada devolvendo sempre a primeira resposta. A prioridade é a seleção de acordo com uma prioridade, prioridade essa obtido pela classe Resposta.

A classe Resposta é uma classe de leitura que permite obter a accao e a prioridade de uma resposta.

2.2.3 Reaccões

Para finalizar é necessário a implementação das reações.

A reação Aproximar realiza a funcionalidade de aproximar de um alvo caso detetado, a classe divide-se em três classes AproximarDir. A aproximação direcional aproxima-se de um

alvo de acordo com uma direção específica frente, direita ou esquerda. A classe Aproximar está no topo da hierarquia de reações.

As classes Evitar e Contornar permitem evitar obstáculos, o Evitar serve para obstáculos á frente e o contornar para a esquerda e direita.

A classe Explorar está no fim da hierarquia. Comportamento responsável por explorar o mapa por movimentos aleatórios.

A classe que cria a hierarquia entre todos estas reações é a classe Recolher.

2.2. Agente deliberativo

2.2.1 Controlo deliberativo

O agente deliberativo é um agente que age por antecipação. Para esta concepção, da mesma maneira que o agente reativo, possui uma classe de controlo designada ControloDelib. Esta classe implementa o algoritmo de processo de tomada decisão e ação. Inicialmente é observado o mundo e atualizadas as crenças. De seguida verificasse a necessidade de reconciderar, se sim deliberasse, planeiasse e posteriormente executasse, senão passasse diretamente ao passo executar que devolve uma ação. A classe possui funções privadas que ajudam na realização do algoritmo. Para o funcionamento do algoritmo é necessário um planeador realizado pela interface Planeador e uma representação interna do mundo feito pela classe ModeloMundo.

O raciciocínio orientado para a ação possui três representações necessárias para a sua realização. Uma representação dos objetivos a atingir, as ações realizáveis pelo agente e uma representação do mundo.

A classe ModeloMundo cria uma representação interna do mundo necessária para o agente. Para realizar esta funcionalidade existe uma lista com todos os elementos de cada coordenada. Possui também uma lista de operadores que representam os movimentos possíveis

de realizar pelo agente, um estado que representa a posição atual do agente valor de apenas leitura, os estados que devolve todas as coordenadas do mapa e por fim o atributo alterado que fica com o valor de verdade quando o mapa é alterado, no caso presente quando o agente recolhe um alvo. As funções permitem saber qual o elemento num determinado estado, atualizar os atributos quando se recolhe o alvo, quais os operadores que se podem aplicar e quais as posições do mapa.

A classe OperadorMover representa as ações realizáveis pelo agente. Para isso precisa de um modelo do mundo e do ângulo do movimento. A classe permite retornar a ação de um OperadorMover, simular o movimento do agente e obter qual o custo de um movimento.

2.2.2 Planeamento automático

Para criar um planeamento é necessário um planeador de modelo de caixa preta. Este planeador recebe um modelo de planeamento, um estado inicial e os objetivos e constrói um plano de ação. Este comportamento é feito pela realização da interface Planeador que obriga a implementação de métodos para obter o operador a partir de um estado, saber se existem planos pendentes e terminar o último plano pendente.

Como o Planeador necessidade de uma representação do modelo do planeamento surge a interface ModeloPlan que no caso do trabalho será o ModeloMundo.

2.2.2 Planeador utilizando pee

Para realizar o planeamento com base em procura de espaço de estados foi utilizada como ferramenta a biblioteca pee. Para isso foi criado um planeador para o pee designado PlanPEE. Esta classe recebe uma procura no espaço de estados, podendo ser a procura Melhor-Primeiro, ou procura A* ou procura Sofrega. Esta classe cria uma lista com um Plano com os nós que deve percorrer até á solução e um ProblemaPlan para acesso a conponentes chave para

resolução do problema. As funções são a concretização da interface Planeador por também ser um planeador.

A classe ProblemaPlan é a classe que permite á classe PlanPEE obter o objetivo do problema e a heurística.

2.2.2 Planeador utilizando pdm

O planeamento PDM tem por base a propriedade probabilísta de Markov. Toda a inteligência por de trás do planeamento é feito através de modelos utilizando uma política e utilidade por base. Para criar um planeador com base em pdm criou-se a classe PDM. O PDM possui dois atributos o gama e o delta max, o gama representa a taxa desconto temporal e o delta max é o valor mínimo da utilidade para continuar a aprender. Esta classe permite saber qual a utilidade do plano, qual a utilidade da ação, a politica, e qual a utilidade e a política de um modelo. A utilidade é o efeito cumulativo da evolução da situação representado por um double. A política é a representação da ação que deve ser realizada em cada estado.

O modelo é representado por meio de uma interface designada ModeloPDM que simboliza o modelo do mundo na forma de um PDM. Onde a função S é o conjunto de estados do mundo, o A o conjunto de ações possíveis no estado s pertentencente ao conjunto de estados S, T a probabilidade transição de um estado(s) para um outro estado (s') através de uma ação e R o retorno esperado na transição de (s) para (s') através de uma ação.

Surge então a necessidade de criar um plano de ação para executar o plano através dos valores obtidos no pdm. A classe PlanPDM, que á semelhança de outras classe de planeamento possui os mesmos métodos com a diferença de possuir um atributo gama, delta max, utilidade e política precisas para utilização do PDM.

Para finalizar é criada a classe ModeloPDMPlan que estende de ModeloPDM e possui também um ModeloPDM que implementa as funções S, A, T e R referidas anteriormente e também os estados e operadores do ModeloPDM no seu interior.

2.2.2 Aprendizagem por reforço

Para se fazer a aprendizagem por reforço é necessário uma memória de aprendizagem com o carácter comportamental, onde a aprendizagem é feita a partir da interação com o ambiente através de um um estado, ação e reforço. Assim como o planeador por pdm o reforço também utiliza uma política e utilidade. Para isso primeiramente é criada a AprendRef que aprende a partir de uma memória de aprendizagem e de uma seleção de ação.

Desta forma surgem duas interfaces a MemoriaAprend para representar a memória de aprendizagem e SelAccao para a seleção da ação.

Para concretizar a MemoriaAprend é criada a MemoriaEsparsa. Esta memória serve para atualizar o estado e a ação e permite saber o valor do reforço de um par estado ação.

Para a concretização da seleção de ação é feita pela classe SelAccaoEGreedy esta estratégia de seleção é feita por um valor epsilon que dita a taxa de aprendizagem do agente. A seleção de ações é feita com base no que o agente aprende e numa seleção de ações aleatórias. Como a seleção é E-Greedy o valor que determina quando o valor é aleatório ou não é o epsilon, sendo que a procura greedy tem probabilidade 1-epsilon, caso este valor seja 0 passa a ser uma procura Greedy. As funções representam este mesmo comportamento onde a seleção de ação é feito com base num valor aleatório onde para valores inferiores a epsilon é aletório senão é sôfrega, e os respetivos métodos para procura sôfrega e aleatória.

Para concluir o mecanismo de aprendizagem falta somente uma classe que implemente o método aprender da classe AprendRef. Posto isto a classe AprendQ implementa a função aprender com o algoritmo Q-Learning.

Desta maneira já temos o mecanismo de aprendizagem todo concluído, falta somente juntar todas as componentes numa única que é a classe MecAprend que cria uma memoria esparsa, seleção ação E-Greedy e uma aprendizagem por Q-Learning. Sendo então abstração à implementação das funções e espondo apenas as funções de contrato com as interfaces.

Por fim falta o controlo para o agente prospetor realizar o comportamento por reforço. A classe ControloAprendRef sendo o controlo implementa a função processar e utiliza a classe MecAprend para aprender e selecionar uma ação. A classe também implementa um passo importante do algoritmo que é o geramento de reforço.