Introdução à Inteligência Artificial

Trabalho de grupo

Ano Lectivo de 2013/2014

12 de Dezembro de 2013

I Observações importantes

1. Este trabalho deverá ser realizado por grupos de 2 a 3 alunos.
2. Este trabalho deverá ser submetido até ao dia 6 de Dezembro de 2013. Prazo extendido para 13 de Dezembro de 2013.
3. Os trabalhos poderão ser entregues para além do prazo, mas serão penalizados em 5% por cada dia adicional
4. Em cada módulo Python submetido, inclua um comentário com o nome e número mecanográfico dos autores
5. Juntamente com o código Python, cada grupo deverá submeter uma apresentação Powerpoint (formato pdf) com no máximo 3 páginas resumindo a concepção subjacente à implementação desenvolvida
6. Pode discutir o trabalho com colegas de outros grupos, mas não pode copiar programas, ou partes de programas, qualquer que seja a sua origem
7. Se discutir o trabalho com colegas de outros grupos, inclua um comentário com o nome e número mecanográfico desses colegas
8. Se recorrer a outras fontes, identifique essas fontes também
9. Em caso algum passe o seu código a colegas de outros grupos, pois, poderá ser utilizado indevidamente
10. Todo o código submetido deverá ser original; embora confiando que a maioria dos alunos fará isso, serão usadas ferramentas de detecção de copianço
11. Alunos envolvidos em casos de copianço terão os seus trabalhos anulados
12. Os trabalhos serão avaliados tento em conta a qualidade da concepção e da implementação, originalidade (incluindo evidência de trabalho autónomo) e desempenho

II Tema do trabalho

Este trabalho envolve a aplicação de conceitos e técnicas de três capítulos principais da disciplina de Introdução à Inteligência Artificial, nomeadamente: programação em Python; tipos e arquitectures de agentes; e técnicas de pesquisa para resolução de problemas.

No âmbito deste trabalho, deverá desenvolver um agente que explora um ambiente virtual, com o objectivo de se deslocar até uma determinada localização nesse ambiente (o destino). No fim, o agente deverá regressar à localização inicial (a origem). O ambiente, que tem uma configuração rectangular, contém vários obstáculos, podendo assumir a tiplogia de um labirinto. A área de destino é assinalada por um emissor de um sinal luminoso (o farol). O agente possui diversos sensores (sensor de colisões, sensores de obstáculos, sensor de farol, bússola, GPS, etc.). O agente movimenta-se no ambiente controlando a potência aplicada aos motores que accionam as suas duas rodas.

III Ambiente e ferramentas

O ambiente virtual será suportado pelo simulador do Concurso CiberRato. O simulador, outras ferramentas e a documentação relevante podem ser descarragados do seguinte repositório:

```
http://sourceforge.net/projects/cpss/files/cpss/
```

Deverá descarregar a versão 2.1, de 2013/04/23. No documento "CiberRato 2013: Rules and Technical Specifications", pode encontrar as especificações técnicas detalhadas sobre a simulação e visualização, incluindo os sensores e actuadores que o seu agente virtual poderá utilizar. Note que, segundo as regras actuais do Concurso CiberRato, as entradas na competição são equipas de vários agentes virtuais, que cooperam na exploração do ambiente, tendo todos os agentes que atingir o farol e regressar à origem. No âmbito deste trabalho, cada grupo desenvolve um único agente que desempenha, de forma individual, as fases de exploração/procura e regresso.

Juntamente com o simulador, está disponibilizado um agente simples na pasta robsample. O simulador, o visualizador e o agente podem ser lançados da seguinte forma:

```
~/ciber/cibertools-v2.1$ ./startSimViewer
.... (noutra consola:)
~/ciber/cibertools-v2.1$ cd robsample
~/ciber/cibertools-v2.1/robsample$ LD_LIBRARY_PATH=../libRobSock/ ./robsample
```

Deverá desenvolver o seu agente em Python. Para o efeito, disponibiliza-se em anexo ao presente enunciado, o módulo crobolink. Este módulo contém a classe CRobLink, a qual agrega todas as funcionalidades de percepção e acção que o agente pode usar. O construtor da classe recebe como entrada o nome do agente, o identificador da posição do agente na grelha de partida e a máquina onde corre. Além do construtor, a classe possui os seguinte métodos:

- readSensors() - Método que lê os sensores, colocando a informação no campo measures. Este campo é, por sua vez, uma instância da classe CMeasures, a qual tem os seguintes campos (valores iniciais entre parentesis):
  - gpsReady (False) - Disponibilidade da leitura do GPS.
  - x (0.0) - Leitura do GPS, coordenada X.
  - y (0.0) - Leitura do GPS, coordenada Y.
  - gpsDirReady (False) - Disponibilidade da leitura da orientação dada pelo GPS.
  - dir (0.0) - Leitura da orientação dada pelo GPS.

- compassReady (False) - Disponibilidade da leitura da bússola.
- compass (0.0) - Leitura da bússola.
- irSensorReady ([False, ..., False]) - Vector de booleanos que indicam disponibilidade de leituras dos sensores de obstáculos.
- irSensor ([0.0, ..., 0.0]) - Leituras dos sensores de obstáculos.
- beaconReady (False) - Disponibilidade da leitura de sensor de farol.
- beacon (False, 0.0) - Leitura do sensor de farol, um tuplo em que o primeiro elemento a True indice que o farol está visível, e o segundo elemento indica o ângulo.
- collisionReady (False) - Disponibilidade da leitura do sensor de colisão.
- collision (False) - True se existe colisão.
- collisionsReady (False) - Disponibilidade do número de colisões disponível.
- collisions (0) - Número de colisões.
- groundReady (False) - Disponibilidade da leitura do sensor de chão.
- ground (−1) - Identificador da área de chão (0,1,2,...) em que está, ou −1, caso não esteja numa área.
- start (False) - True se botão start está ligado.
- stop (False) - True se botão stop está ligado.
- endLed (False) - True se luz de terminação está ligada.
- returningLed (False) - True se luz de retorno está ligada.
- visitingLed (False) - True se luz de visita do farol está ligada.
- time (0) - Tempo de prova.
- scoreReady (False) - Disponibilidade da pontuação.
- score (100000) - Pontuação.
- arrivalTimeReady (False) - True se tempo de chegada está disponível.
- arrivalTime (10000) - Tempo de chegada.
- returningTimeReady (False) - True se tempo de retorno está disponível.
- returningTime (10000) - Tempo de retorno.

No âmbito deste trabalho, o simulador deve ser configurado de forma a que todos os sensores estejam disponíveis em cada iteração do ciclo de controlo. Para tal, deverá ir ao menu File, submenu Configuration, opção Edit configuration, tabela Requests, e desmarcar todos os sensores.

- driveMotors(lPow,rPow) - Aplicar potência aos motores das rodas esquerda e direita.
- setReturningLed(val) - Liga a luz de retorno se val=True, ou desliga-a caso contrário.
- setVisitingLed(val) - Liga a luz de visita se val=True, ou desliga-a caso contrário.
- finish() - Termina a prova, indicando que regressou à localização inicial.

Em anexo a este enunciado, encontra ainda um pequeno módulo de teste (test_croblink), que poderá usar da seguinte forma:

```
~/ciber/cibertools-v2.1$ ./startSimViewer
.... (noutra consola:)
~/ciber/cibertools-v2.1$ python test_croblink.py
```

IV Adendas

1 2013-11-26

- Foi corrigido o processamento do sensor de chão na função readSensors() da classe CRobLink.
- Disponibiliza-se em anexo o módulo myrob, parcialmente desenvolvido nas aulas teóricas, o qual contém um agente que adopta um comportamento reactivo e exploratório, tanto para a fase de procura do farol como para a fase de regresso à posição de partida.
- No teste dos agentes desenvolvidos poderão ser usados labirintos como os seguintes:
  - Ciber 2001 - Manga 2
  - Ciber 2004 - Manga 1
  - Ciber 2004 - Manga 2
  - Ciber 2004 - Manga 3
  - Ciber 2005 - Manga 1
  - CiberRTSS2007 - Stage 1
  - Ciber 2008 - Manga 1
  - Ciber 2009 - Manga 2

2 2013-12-03

- Foi extendido o prazo do trabalho até 13 de Dezembro de 2013.

3 2013-12-11

- Está disponibilizada em anexo a este guião uma versão do simulador que calcula a pontuação para o cenário do trabalho. Devem substituir a pasta simulator por esta versão e repetir os passos de instalação.

4 2013-12-12

- As regras de cálculo da pontuação são as seguintes:
  - A pontuação inicial é de 200 pontos.
  - Cada colisão com paredes ou outros agentes é penalizada com 2 pontos
  - Quando o robô assinala correctamente a sua chegada ao farol, 100 pontos são subtraídos à pontuação.
  - Cada 25 ciclos de tempo de regresso em excesso de $T_R + 25$, em que T_R é o tempo óptimo, é penalizado com 1 ponto. (Nota: O tempo de regresso óptimo é o tempo correspondente ao melhor caminho de regresso em velocidade máxima. Exemplo: Se o tempo óptimo é 200 ciclos, e o agente demorou 325 ciclos, a penalização será de 4 pontos.)

  - Acresce ainda uma penalização proporcional à distância final do agente relativamente à posiçao de partida. O número de pontos correspondentes a esta penalização é dado pela percentagem da distância remanescente até à posição de partida relativamente à distância total do farol à posição de partida. (Exemplo: Se o agente tinha percorrido apenas 80% do caminho total de regresso quando a prova terminou, terá uma penalização de 20 pontos.)

  Nota: Estas regras eram as usadas na versão antiga (não cooperativa) do Concurso Ciber-Rato.

- Para o simulador calcular a pontuação correctamente, o agente, ao chegar ao farol, deve ligar o sinal de visita (visitingLed), e depois deve desligá-lo e ligar o sinal de regresso (returningLed). O módulo myrob, disponibilizado em anexo, foi actualizado para cumprir com esta especificação.

- Cada grupo deve submeter uma pasta zipada contendo a apresentação Powerpoint referida acima, e um ou mais ficheiros contendo todo o código Python relevante.

- De preferência, os programas devem estar preparados para correr em Python 2.7. Grupos que não o possam garantir, devem avisar o docente.

- Para facilitar os testes, o módulo principal deverá chamar-se myrob, permitindo o lançamento do agente através do comando:

```
~/testes$ python myrob.py
```