Introdução à Inteligência Artificial

Licenciatura em Engenharia Informática, Engenharia Informática – Pós Laboral e Engenharia Informática – Curso Europeu

2º Ano – 1º semestre

Aulas Laboratoriais

Ficha 3: Agentes reativos com memória

Nesta ficha será incorporada, a parte do problema apresentado na ficha 2, a capacidade de os agentes implementados terem memória. A memória dos agentes vai permitir-lhes ter mais capacidades inteligentes e comportamentos adicionais. Especificamente, nesta ficha, pretende-se implementar o seguinte:

- **Alimentação dos agentes**
  Os agentes deverão ter um nível de energia, que lhes possibilitará fazer as suas respetivas ações. Este nível poderá ser incrementado sempre que se alimentarem ou decrementado quando executam uma outra qualquer ação. Os agentes só poderão morrer quando o seu nível de energia for menor ou igual a zero;
- **Formigas com capacidade de transporte de comida**
  Uma formiga só poderá transportar consigo uma quantidade limitada de comida. Se a capacidade máxima de armazenamento for atingida, a formiga não poderá recolher mais comida. Deverá então encontrar o seu ninho (célula azul) e despejar lá a comida que transporta. Se encontrar o ninho antes de atingir a sua capacidade máxima, deverá, também, libertar a comida que leva consigo, reiniciando a contagem do número de pedaços de comida que transporta;
- **Relação formigas – caracóis**
  Uma formiga deverá poder comer um caracol, desde que esteja a transportar pelo menos dez pedaços comida (contagem feita em separado do nível de energia). Quando a formiga comer um caracol, a contagem dos pedaços de comida que transporta nesse momento deverá ser reinicializada, fazendo com que tenha que voltar a recolher novamente pelo menos mais dez pedaços de comida antes de poder voltar a comer outro caracol;
- **Agentes com capacidade de reprodução**
  As formigas e os caracóis deverão reproduzir-se quando tiverem um nível de energia superior a um determinado limiar, mas apenas com uma determinada probabilidade.

Assim, para adicionar estas novas características ao modelo iniciado na ficha 2, seguir os seguintes passos:

Passo 1: Recuperação de parte do modelo implementado na ficha 2

Abrir o ficheiro implementado aquando da resolução da ficha 2 ou, se não o tiver, abrir o ficheiro ***IIA_Ficha3_Inicio.nlogo***, que se encontra no *Moodle* (este ficheiro contém a resolução parcial do problema dos agentes reativos, apresentado na ficha 2).

Passo 2: Manutenção dos agentes no ambiente

No modelo a implementar nesta ficha, as formigas e caracóis quando chegarem ao ninho não deverão desaparecer, podendo continuar a andar pelo ambiente. Assim, retirar todos os comandos ***die*** dos procedimentos ***move-ants*** e ***move-snails***, quando os agentes detetarem o respetivo ninho.

Passo 3: Colocação de comida no ambiente

a) Alterar o procedimento setup-patches de forma a colocar comida no ambiente. Para isso deverá ser pintado de verde 15% das células do ambiente 2D, mantendo na mesma as armadilhas no ambiente;
b) Colocar um elemento monitor para se ver a quantidade de comida.

Passo 4: Alimentação dos agentes, com recurso à MEMÓRIA

c) Na zona de definição das variáveis do modelo (logo no seu início), associar aos agentes uma propriedade (**memória**) de nome energia:

turtles-own [energia]

d) Inicializar esta propriedade, no procedimento ***setup-turtles***, com o valor 100, para todos os agentes;
e) Alterar o procedimento ***move-ants***, de forma a integrar o seguinte:
    1. Quando a formiga percecionar comida na célula que está logo à sua frente (ou seja, se essa *patch* for verde):
        i. Avançar a formiga;
        ii. Aumentar a energia da formiga em 50 unidades;
        iii. Alterar a cor da *patch* para preto (comida desaparece).
    2. Qualquer movimento da formiga, à exceção do anterior, retira-lhe uma unidade de energia.
f) Alterar o procedimento ***move-snails***, de forma a integrar o seguinte:
    1. Quando o caracol percecionar comida na célula onde habita (ou seja, a *patch* em que está for verde):
        i. Aumentar a energia do caracol em 50 unidades;
        ii. Alterar a cor da patch para preto (comida desaparece).
    2. Qualquer movimento do caracol, à exceção do anterior, retira-lhe uma unidade de energia.

Passo 5: Morte dos agentes, com recurso a MEMÓRIA

a) No procedimento ***go***, acrescentar a chamada ao procedimento ***check-death***, conforme se pode ver abaixo:

```
to go
   …
   check-death
   if count turtles = 0
   [ stop ]
end
```

b) Criar o procedimento ***check-death***, que verifique se a energia de cada um dos agentes é inferior ou igual a zero. Os agentes nesta situação deverão morrer;
c) No separador *Interface* simular do modelo, corrigindo eventuais erros.

Passo 6: Transporte e armazenamento de comida, por parte das formigas, com recurso a MEMÓRIA

Uma formiga, quando chega a um local onde está comida, além de se alimentar, deverá transportar parte dessa comida para o ninho. Assim,

a) Na zona de definição das variáveis do modelo, associar às formigas apenas uma nova propriedade (**memória**), de nome ***nErva***:

ants-own [nErva]

b) Inicializar a zero a variável criada atrás, no procedimento ***setup-turtles*** (não esquecer que essa variável só pertence à formiga);
c) Sempre que uma formiga encontrar comida, deverá, também, atualizar o contador ***nErva***, assinalando que leva consigo mais um pedaço de comida para o ninho;
d) No separador interface, criar um *slider* para controlar a capacidade máxima de comida que as formigas poderão transportar (chamar ***capMax*** à variável global associada a esse *slider*);
e) Se o contador ***nErva*** ≥ ***capMax*** a formiga não poderá alimentar-se nem transportar mais pedaços de comida. Alterar esta limitação no procedimento ***move-ants***;
f) Quando a formiga chegar ao ninho, deverá descarregar a comida transportada, para que esta lá fique armazenada. Assim, atualizar a variável ***blue-nest***, incrementando-a com os pedaços de comida depositados e colocar a zero o contador ***nErva***.

Passo 7: Relação formigas – caracóis, com MEMÓRIA

Uma formiga poderá comer um caracol desde que esteja a transportar pelo menos dez pedaços de comida. Após comer um caracol, a contagem de pedaços de comida acumulados deverá ser reinicializada, fazendo com que tenha que voltar a recolher novamente pelo menos dez pedaços de comida antes de poder voltar a comer outro caracol. Assim,

a) Alterar o procedimento ***move-ants*** da seguinte maneira:
    1. Verificar se há algum caracol na célula da frente à que está a formiga e se essa formiga está a transportar pelo menos 10 pedaços comida (***nErva ≥ 10***);
    2. Se a condição atrás definida for verdadeira, a formiga recebe a energia do caracol, que morre, e o contador ***nErva*** é colocado a zero;
    3. Se não houver qualquer caracol ou o contador ***nErva*** for menor que 10, a formiga mantém o comportamento usual.

Passo 8: Reprodução de agentes

a) No separador *Interface*, acrescentar um *switch* para ativar ou desativar a reprodução (associar ao *switch* a variável global ***reproduce?***);
b) No separador *Interface*, acrescentar três novos *sliders* para parametrizar as probabilidades de reprodução e a energia mínima que um agente deverá ter para se poder reproduzir. Assim:
    1. Slider1 – com valores a variar entre 0 e 100, associado à variável global ***reproductionFormigas***, com um valor inicial de 50%;
    2. Slider2 – com valores a variar entre 0 e 100, associado à variável global ***reproductionCaracois***, com um valor inicial de 50%;
    3. Slider3 – com valores a variar entre 100 e 500, associado à variável global ***birthEnergy***, com um valor inicial de 200. Este parâmetro deverá ser usado por todos os agentes (formigas e caracóis).

c) No procedimento ***go***, acrescente a chamada ao procedimento ***reproduction***:

```
to go
   …
   check-death
   if reproduce?
   [
      reproduction
   ]
   if count turtles = 0
   [
      stop
   ]
end
```

d) Crie o procedimento ***reproduction***, implementando as seguintes funcionalidades:

1. As formigas e os caracois só podem reproduzir-se se a sua energia for superior ao da variável ***birthEnergy***, parametrizada no *slider* criado anteriormente;
2. As formigas reproduzem-se com uma probabilidade ***reproductionFormigas***, parametrizada no *slider* criado anteriormente;
3. Os caracóis reproduzem-se com uma probabilidade ***reproductionCaracois***, parametrizada no *slider* criado anteriormente;
4. Quando uma formiga se reproduz, a sua energia é dividida por dois e é criada uma cópia, que deve ficar posicionada a 5 *patches* de distância, conforme o comando seguinte:

   hatch 1 [jump 5]
5. Quando um caracol se reproduz, a sua energia é dividida por dois e é criada uma cópia, que deve ficar posicionada à esquerda do seu progenitor, conforme o comando seguinte:

   hatch 1 [move-to patch-left-and-ahead 90 1]

e) Execute no separador *Interface* os procedimentos ***setup*** e ***go***, corrigindo eventuais erros.

Passo 9: Reaparecimento de comida no ambiente

a) No procedimento ***go***, chamar ao procedimento ***regrow-food***:

```
to go
   …
   if reproduce?
   [
      reproduction
   ]
   regrow-food
   …
end
```

b) Criar o procedimento ***regrow-food***, de maneira a que pergunte a todas as *patches* se existem menos de 50 células verdes (comida) e, se existirem, altere as *patches* pretas para verdes com uma probabilidade

de 2% (fazendo com que novos pedaços de comida cresçam). Para contar as células verdes usar os comandos:

if count patches with [pcolor = green] < 50

Passo 10: Agentes com indicação da sua energia

a) No separador *Interface*, adicionar um elemento *switch* e atribuir-lhe o nome de ***show-energy?***;

b) Nos procedimentos ***setup-turtles*** e ***go*** chamar um novo procedimento de nome ***display-labels***;

c) Criar o procedimento ***display-labels***, que mostre a energia de todos os agentes, caso a variável associada ao *switch*, ***show-energy?***, esteja ativa:

```
to display-labels
  ask turtles
  [
    set label ""
    if show-energy?
    [
      set label energia
    ]
  ]
end
```

d) Regressar ao separador *Interface* para testar as funcionalidades acrescentadas.

Passo 11: Competição

a) No procedimento ***go*** chamar um novo procedimento de nome ***competition***:

```
to go
  …
  display-labels
  competition
  tick
end
```

b) Criar o procedimento ***competition***, que simule a competição entre agentes do mesmo tipo, de maneira a que se os agentes se encontrarem em *patches* vizinhas, o que tiver maior energia recebe a energia do outro, que morre.