UNIVERSIDADE
DE TRÁS-OS-MONTES
E ALTO DOURO

Inteligência Artificial – 3º ano de Engenharia Informática

2025/2026

Relatório do Trabalho Prático 1 – Simulação de Ecossistema Aquático com Agentes

Feito por:

Luís Carvalho, al81967

João Nogueira, al81605

Conteúdo

1. Introdução e Objetivo

Este relatório descreve o desenvolvimento do modelo de simulação Eco2, criado no âmbito da unidade curricular de Inteligência Artificial. O principal objetivo deste trabalho é a modelação e simulação computacional de um ecossistema aquático, explorando a interação entre diferentes tipos de agentes e o impacto da poluição nesse ambiente.

Utilizando a ferramenta NetLogo, foi desenvolvido um sistema multiagente que representa um ecossistema numa superfície quadrangular, habitado por agentes com papéis de fauna, flora e poluidores. O modelo final, Eco2, representa a segunda fase de implementação, introduzindo um maior grau de complexidade e dinâmicas que permitem observar a procura por um equilíbrio do ecossistema.

2. Descrição do Modelo

O modelo Eco2 simula um ambiente aquático onde as interações entre os agentes e o ambiente determinam a saúde e a evolução do ecossistema.

2.1. O Ambiente

O mundo da simulação é uma grelha 2D de células (patches). O ambiente divide-se em duas áreas principais: uma zona terrestre e uma zona aquática. A dinâmica do ecossistema ocorre primariamente na zona aquática, definida por um agentset global chamado patches-de-agua. Cada patch aquático possui três propriedades fundamentais:

- **quantidade-alga:** Representa a biomassa de alga presente na célula.
- **afetado?:** Um valor booleano que indica se a célula está ou não contaminada.
- **nivel-toxicidade:** Um valor numérico que quantifica o grau de poluição da célula.

2.2. Os Agentes

O ecossistema é composto por três tipos de agentes principais, conforme sugerido pelo enunciado:

A. Agente "Planta" (Algas) A flora do ecossistema é representada pela variável quantidade-alga de cada patch. Esta abordagem modela eficazmente a natureza estacionária das plantas. O seu comportamento é regido por duas funções principais:

- **Crescimento:** As algas crescem a uma taxa definida por um slider (taxa-crescimento-algas), com maior intensidade no centro do ecossistema. O crescimento é inibido pelo nivel-toxicidade do patch.
- **Consumo:** As algas servem de alimento para o agente "Animal", sendo consumidas quando um peixe se encontra no patch.

B. Agente "Animal" (Peixes) A fauna é representada por um breed de agentes móveis chamado peixes. Cada peixe possui um ciclo de vida complexo, gerido pelas seguintes propriedades: energia, idade, passos-sem-virar e max-passos-retos. Os seus comportamentos são:

- **Movimento:** Deslocam-se de forma semi-aleatória dentro da zona aquática, um passo por tick. Os agentes possuem a capacidade de percecionar o estado da célula adjacente à sua frente e têm uma probabilidade de desviar o seu percurso se detetarem que a mesma está contaminada.
- **Alimentação:** Ao passar por um patch com quantidade-alga suficiente, o peixe come, repondo a sua energia.
- **Ciclo de Vida e Morte:** A energia diminui a cada tick. Um peixe morre se a sua energia chegar a zero ou se atingir uma idade máxima.
- **Reprodução:** Se um peixe acumular energia suficiente (acima de um limiar), pode reproduzir-se, gerando um novo agente e gastando uma porção da sua energia no processo.

C. Agente "Poluidor" (Meteoritos) O agente poluidor foi implementado na forma de meteoritos que caem sobre o ecossistema. Este agente introduz a poluição no ambiente.

- **Geração:** Os meteoritos são criados no topo do mundo com uma probabilidade definida pelo slider frequencia-meteoritos.
- **Impacto:** Cada meteorito cai verticalmente e, ao atingir uma profundidade aleatória na água, "desintegra", contaminando uma área circular (in-radius).
- **Mecânica de Poluição:** Conforme o protocolo, o impacto de um meteorito apenas deposita resíduos em células que se encontram limpas (afetado? = false). A cor da célula muda de acordo com o nível de toxicidade depositado, que é variável.

3. Componente de Inovação

O modelo Eco2 vai além dos requisitos mínimos, introduzindo várias inovações que aumentam o seu realismo e complexidade, um aspeto valorizado na avaliação.

- **Dinâmica de Poluição Avançada:** Em vez de um agente poluidor com movimento simples, foi criado um sistema de eventos de poluição (queda de meteoritos). Este sistema é controlado por múltiplos parâmetros (frequência, severidade, raio de impacto), permitindo simular desde pequenas contaminações localizadas a eventos catastróficos.
- **Ciclo de Energia e População:** Os agentes animais não se movem apenas aleatoriamente; eles participam num ciclo ecológico completo de consumo de recursos (energia), reprodução e morte, o que permite observar dinâmicas populacionais realistas.
- **Autolimpeza do Ambiente:** A poluição não é permanente. A função degradar-toxicidade simula a capacidade de o ecossistema se limpar ao longo do tempo, permitindo estudar cenários de recuperação ambiental e a busca por um equilíbrio dinâmico.

4. Interface e Controlo da Simulação

A interface foi desenhada para oferecer um controlo completo sobre os parâmetros da simulação e para uma visualização clara dos resultados.

- **Botões:** Inclui os botões Setup (para inicializar o mundo), Go (para execução contínua), Go_Once (para avançar um único tick) e Go_N (para avançar N ticks), conforme solicitado no protocolo.
- **Sliders:** Diversos sliders permitem ao utilizador ajustar em tempo real todos os parâmetros chave do modelo, como o tamanho inicial da população (Pop_size), taxas de reprodução e crescimento, e todas as variáveis da ameaça dos meteoritos.
- **Visualização:** O estado do ecossistema é apresentado visualmente através das cores dos patches (que indicam densidade de algas e toxicidade). Adicionalmente, um conjunto de **gráficos** permite a análise quantitativa da evolução temporal da População de Peixes, Quantidade de Algas, e Nível de Contaminação, cumprindo os requisitos de visualização.

5. Conclusão

O modelo Eco2 implementado cumpre com sucesso todos os objetivos propostos para o trabalho prático. Através da ferramenta NetLogo, foi possível criar uma simulação rica de um ecossistema aquático, onde as interações complexas entre os agentes e o ambiente dão origem a dinâmicas emergentes. A flexibilidade da interface permite a realização de diversas experiências, testando a resiliência do ecossistema a diferentes tipos e intensidades de poluição. O trabalho serve como uma demonstração prática e eficaz das capacidades dos sistemas multiagente na modelação de problemas ecológicos.

6. Código (Eco2)

6.1 Variáveis globais e definição de agentes

```
globals [
  step-counter          ; Contador para o botão Go N
  total-mortes          ; Contadores para estatísticas gerais
  total-nascimentos
  mortes-tick           ; Contadores para os gráficos, resetados a cada tick
  nascimentos-tick
  patches-de-agua       ; Agentset para guardar os patches de água, melhora a performance
]

breed [meteoritos meteorito]
breed [peixes peixe]

; Variáveis que cada patch (célula do mundo) vai ter
patches-own [
  afetado?              ; A célula está contaminada ou não? (true/false)
  nivel-toxicidade      ; Qual o nível da contaminação
  quantidade-alga       ; Quantidade de comida (algas) na célula
]

; Variáveis que cada meteorito vai ter
meteoritos-own [
  ja-infetou?           ; Para garantir que cada meteorito só polui uma vez
  profundidade-impacto  ; O ponto Y onde o meteorito vai "explodir"
]

; Variáveis que cada peixe vai ter
peixes-own [
  passos-sem-virar      ; Controla o movimento para não ser demasiado errático
  max-passos-retos
  energia               ; Essencial para a sobrevivência e reprodução
  idade                 ; Para morrerem de velhice
]
```

6.2 Procedimento de atualização (Botão Setup)

```
to setup
  clear-all
  set-default-shape peixes "fish"
  set-default-shape meteoritos "meteorito"

  ; Define a nossa "zona de água" com limites em todos os lados
  set patches-de-agua patches with [pycor <= 8 and pycor > -15 and pxcor >= -14 and pxcor <= 14]

  ; Configura o aspeto inicial do mundo (céu e água)
  ask patches [
    ifelse pycor <= 10
      [ set pcolor [73 104 144] ]
      [ set pcolor [130 170 222] ]
    set afetado? false
    set nivel-toxicidade 0
    set quantidade-alga 0
  ]

  ; Distribui as algas iniciais, com mais concentração no centro
  ask patches-de-agua [
    let dist sqrt (pxcor ^ 2 + (pycor + 2) ^ 2)
    let max-dist 15
    let fator max list 0 (1 - (dist / max-dist))
    set quantidade-alga (random 50) * fator
  ]

  ; Cria a população inicial de peixes
  create-peixes Pop_size [
    move-to one-of patches-de-agua with [pxcor >= -14 and pxcor <= 14]
    set heading random 360
    set size 1.5
    set color pink
    set passos-sem-virar 0
    set max-passos-retos 5 + random 10
    set energia 80 + random 40
    set idade 0
  ]

  atualizar-cores-algas

  ; Reseta todos os contadores
  set step-counter 0
  set total-mortes 0
  set total-nascimentos 0
  set mortes-tick 0
  set nascimentos-tick 0
  reset-ticks
end
```

6.3 Comportamento dos meteoritos

```
to processar-meteoritos
  ; Decide se um novo meteorito é criado, com base na probabilidade do slider
  if random-float 1 < (frequencia-meteoritos / 100) [
    create-meteoritos 1 [
      set size 2
      set heading 180
      setxy random-xcor max-pycor ; Começa numa posição X aleatória no topo
      set color orange
      set ja-infetou? false
      ; Define uma profundidade aleatória na água onde vai haver o impacto
      set profundidade-impacto -14 + random-float (10 - (-14))
    ]
  ]
  ask meteoritos [ cair ]
end

to cair
  ; Move o meteorito para baixo
  if pycor > min-pycor [
    set ycor ycor - 1
    set heading 180

    ; Quando atinge a profundidade de impacto, "explode"
    if (not ja-infetou?) and (pycor <= profundidade-impacto) [
      let patch-de-impacto patch-here
      let toxicidade-maxima (1 + random-float severidade-impacto)

      ; Contamina todos os patches limpos num certo raio (splash damage)
      ask patches in-radius raio-impacto [
        if not afetado? [
          let distancia distance patch-de-impacto
          ; A toxicidade é mais forte no centro e diminui com a distância
          let toxicidade-resultante toxicidade-maxima * (1 - (distancia / (raio-impacto + 1)))
          set afetado? true
          set nivel-toxicidade toxicidade-resultante
        ]
      ]
      set ja-infetou? true ; Marca como já tendo infetado
    ]
  ]
  ; Morre quando chega ao fundo
  if pycor <= min-pycor [ die ]
end
```

6.4 Comportamento dos peixes

```
to mover-peixe
  let next-patch patch-ahead 1

  ; Para o movimento não ser sempre em frente, vira um pouco de vez em quando
  if passos-sem-virar >= max-passos-retos [
    rt random 90 - 45
    set passos-sem-virar 0
    set max-passos-retos 5 + random 10
  ]

  ; Perceciona o patch à frente: se estiver afetado, há uma chance de se desviar
  if [afetado?] of next-patch [
    if random-float 1 < 0.3 [ rt random 180 ]
  ]

  ; Se o próximo patch for água, avança. Senão, vira para não bater na "parede".
  ifelse next-patch != nobody and member? next-patch patches-de-agua [
    forward 1
    set passos-sem-virar passos-sem-virar + 1
  ]
  [
    rt random 180
    set passos-sem-virar 0
    set max-passos-retos 5 + random 10
  ]
  set idade idade + 1
end

to comer-alga
  ; Se houver comida suficiente no patch atual, come e ganha energia
  if [quantidade-alga] of patch-here > 10 [
    let alga-comida 20 + random 10
    set energia energia + alga-comida
    ask patch-here [ set quantidade-alga quantidade-alga - alga-comida ]
  ]
end

to morrer-peixe
  ; Um peixe morre se ficar sem energia (fome) ou se ficar muito velho
  if energia <= 0 or idade > 300 [
    set mortes-tick mortes-tick + 1
    set total-mortes total-mortes + 1
    die
  ]
end
```

```
to reproduzir-peixe
  ; Limita a população máxima para evitar sobrepopulação
  if count peixes > pop-max-peixes [ stop ]

  ; Se tiver energia suficiente, tem uma chance de se reproduzir
  if random-float 10 < taxa-reproducao [
    if energia > 80 [
      set energia energia - 50  ; Gasta energia para se reproduzir
      hatch 1 [ ; Cria um "filho"
        rt random-float 360
        fd 1
        ; Define as propriedades do novo peixe
        set idade 0
        set passos-sem-virar 0
        set max-passos-retos 5 + random 10
        set energia 70 + random 30
      ]
      set nascimentos-tick nascimentos-tick + 1
      set total-nascimentos total-nascimentos + 1
    ]
  ]
end
```

6.5 Comportamentos do ambiente

```
to crescer-algas
  ; Para cada patch de água, faz crescer um pouco as algas
  ask patches-de-agua [
    let dist sqrt (pxcor ^ 2 + (pycor + 2) ^ 2)
    let max-dist 15
    let fator max list 0 (1 - (dist / max-dist)) ; Fator de crescimento (mais no centro)
    ; A toxicidade do patch reduz a taxa de crescimento
    let crescimento taxa-crescimento-algas * fator * (1 - (nivel-toxicidade / 10))
    set quantidade-alga quantidade-alga + crescimento
    ; Limita a quantidade máxima de algas por patch
    if quantidade-alga > max-alga-por-patch [ set quantidade-alga max-alga-por-patch ]
    if quantidade-alga < 0 [ set quantidade-alga 0 ]
  ]
end

to degradar-toxicidade
  ; Simula a autolimpeza do ambiente: a toxicidade diminui lentamente
  ask patches with [afetado?] [
    set nivel-toxicidade nivel-toxicidade - 0.1
    if nivel-toxicidade <= 0 [
      set afetado? false
      set nivel-toxicidade 0
    ]
  ]
end
```

```
to verificar-contaminacao
  ; Outra causa de morte: se um peixe está num patch contaminado, tem 50% de chance de morrer
  ask peixes [
    if [afetado?] of patch-here [
      if random-float 1 > 0.5 [
        set mortes-tick mortes-tick + 1
        set total-mortes total-mortes + 1
        die
      ]
    ]
  ]
end



to atualizar-cores-algas
  ; Atualiza a cor de cada patch na área de água.
  ask patches with [pycor <= 10] [

    ; Define 70 como o valor de referência para "100% verde".
    let valor-fixo-para-cor 70

    ; Calcula a intensidade do verde (de 0 a 1) com base na referência fixa.
    let intensidade-alga min list 1 (quantidade-alga / valor-fixo-para-cor)

    ; Calcula a cor base, que vai de azul (pouca alga) a verde (muita alga).
    let r-base 73 * (1 - intensidade-alga)
    let g-base 104 + (100 - 104) * intensidade-alga
    let b-base 144 * (1 - intensidade-alga)

    ; Calcula e aplica um "filtro" de toxicidade que torna a cor mais amarelada.
    let ajuste-toxicidade nivel-toxicidade / 5 * 50
    let r r-base + ajuste-toxicidade
    let g g-base - ajuste-toxicidade / 2
    let b b-base

    ; Garante que os valores de cor RGB ficam no intervalo válido (0-255).
    set r min list 255 max list 0 r
    set g min list 255 max list 0 g
    set b min list 255 max list 0 b

    ; Aplica a cor final ao patch.
    set pcolor (list r g b)
  ]
end
```

6.6 Loop principal da simulação (botão Go)

```
to go
  ; Se não houver mais peixes, a simulação para
  if not any? peixes [ stop ]

  ; Reseta os contadores dos gráficos a cada passo
  set mortes-tick 0
  set nascimentos-tick 0

  ; Ordem dos eventos a cada tick
  processar-meteoritos  ; Verifica se um novo meteorito deve cair
  crescer-algas         ; As algas crescem

  ask peixes [          ; Cada peixe executa as suas ações
    mover-peixe
    set energia energia - 0.5  ; Gasto de energia passivo
    comer-alga
    morrer-peixe
    reproduzir-peixe
  ]

  degradar-toxicidade     ; O ambiente tenta limpar-se
  verificar-contaminacao  ; Verifica se algum peixe morre por contaminação
  atualizar-cores-algas   ; Atualiza as cores do mundo com base nos novos valores

  tick ; Avança o relógio da simulação
end
```

6.7 Botões auxiliares (Go_N e Go_once)

```
to go_n
  ; Código duplicado do 'go' para correr N passos
  if step-counter < N_moves [
    set mortes-tick 0
    set nascimentos-tick 0
    processar-meteoritos
    crescer-algas
    ask peixes [
      mover-peixe
      set energia energia - 1.0
      comer-alga
      morrer-peixe
      reproduzir-peixe
    ]
    degradar-toxicidade
    verificar-contaminacao
    atualizar-cores-algas
    set step-counter step-counter + 1
    tick
  ]
end
```

```
to go_once
  ; Código duplicado do 'go' para correr um único passo
  set mortes-tick 0
  set nascimentos-tick 0
  processar-meteoritos
  crescer-algas
  ask peixes [
    mover-peixe
    set energia energia - 0.8
    comer-alga
    morrer-peixe
    reproduzir-peixe
  ]
  degradar-toxicidade
  verificar-contaminacao
  atualizar-cores-algas
  tick
end
```

6.8 Funções de relatório (Reporters)

```
to-report prob[x]
  ; Função auxiliar para probabilidades
  report (random-float 1 < x)
end

to-report populacao-peixes
  ; Reporta o número atual de peixes
  report count peixes
end

to-report media-quantidade-algas
  ; Reporta a média de algas na água
  let patches-agua patches with [pycor <= 10]
  ifelse any? patches-agua [
    report mean [quantidade-alga] of patches-agua
  ] [ report 0 ]
end

to-report patches-contaminados
  ; Reporta o número de patches contaminados
  report count patches with [afetado?]
end

to-report taxa-contaminacao
  ; Reporta a percentagem do ambiente que está contaminada
  let total-patches-agua count patches with [pycor <= 10]
  ifelse total-patches-agua > 0 [
    report (patches-contaminados / total-patches-agua) * 100
  ] [ report 0 ]
end

to-report media-energia-peixes
  ; Reporta a energia média da população de peixes
  ifelse count peixes > 0 [ report mean [energia] of peixes ] [ report 0 ]
end
```

```
to-report idade-media-peixes
  ; Reporta a idade média da população de peixes
  ifelse count peixes > 0 [ report mean [idade] of peixes ] [ report 0 ]
end

to-report total-algas
  ; Reporta a biomassa total de algas
  report sum [quantidade-alga] of patches-de-agua
end
```