Relatório do Trabalho Prático 1: Agentes Inteligentes

Página de Rosto

- Nome da instituição: Universidade
- Curso: Licenciatura em Engenharia Informática
- Unidade Curricular: Sistemas de Controlo
- Trabalho Prático 1: Agentes Inteligentes
- Data de Entrega: 21/10/2024
- Nome e Número dos Alunos: José Alves al73239 Diogo Aperta al 75748

1. Introdução

Com este trabalho pretende-se desenvolver competências sobre a simulação computacional de sistemas com agentes utilizando o NetLogo. Para isso realizamos um trabalho sobre um modelo Netlogo onde se simula o processo de poluição numa superfície retangular através de três poluidores e um agente de limpeza que vai tentar limpar o ambiente .

2. Fase de Implementação 1: Desenvolvimento do Modelo (Robot1)

Na primeira fase de implementação implementamos no NetLogo o "Robot1", que simula um sistema de poluição e limpeza através de Polluters e um cleaner.

O projeto é composto por um ambiente, agentes, a movimentação dos agentes, os resíduos, a energia e a limpeza.

Ambiente: É um conjunto de células(patches) que representam o espaço físico.

Agentes: Os agentes existentes são os Polluters e o Cleaner, onde os Polluters poluem o ambiente e o Cleaner tenta limpar o que esta poluído. Os Polluters decidem se poluem através de uma função probabilística, que são alteradas pelo utilizador. O Cleaner também tem uma energia inicial ajustável.

Resíduos: Os Polluters podem depositar 3 tipos de resíduos que podem ser distinguidos por 3 tipos de cores diferentes.

Energia: O Cleaner a cada movimento que faz consome energia, e retoma ao ponto de recarregamento quando a energia esta baixa. A energia do Cleaner é carregada com um determinado valor ajustável pelo utilizador.

Limpeza: O cleaner sempre que passa pelo resíduo limpa e armazena-o. O cleaner só pode transportar um número máximo de resíduos e quando atinge esse limite tem de ir ao contentor

3. Fase de Implementação 2: Inovações e Melhorias (Robot2)

Nesta fase, o modelo foi melhorado e foram criadas novas inovações de modo a aumentar a complexidade do projeto.

Inovações criadas:

- Foi implementado um painel solar onde o cleaner pode se recarregar indo ter com o próprio painel solar. Este Painel tem um tempo de vida aleatório e uma eficiência que varia aleatoriamente entre 1 e 3.
- Os Polluters movem preferindo os patches que estão limpos e evitam os depósitos.
- Como cada polluter tem o seu nível de poluição, vai afetar a energia do cleaner ao limpar, onde o lixo pesado retira 25% da energia atual do cleaner e o lixo toxico retira 50%.
- Foi implementado uma super bomba de lixivia que é capaz de limpar num raio determinado todo o resíduo acumulado. Esta bomba só pode ser utilizada uma vez.

4. Conclusão

O trabalho Pratico de simulação de agentes em Netlogo resultou num modelo eficaz e bem estruturado de um sistema de poluição e limpeza. O modelo evolui ao longo das fases de implementação, com varias inovações.

5. Referências Bibliográficas

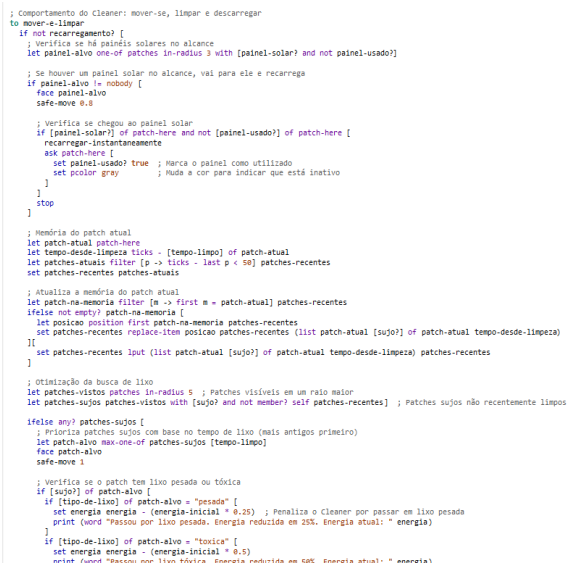
1. Wilensky, U. (1997). NetLogo. http://ccl.northwestern.edu/netlogo/
2. Moura Oliveira P. B., Solteiro Pires E. J., et al. (2022), “Revisiting the Simulated Annealing Algorithm from a Teaching Perspective”

Anexo A: Código

Este Código é capaz de ajustar o raio de exploração, examinar os patches ao seu redor, encontrar os patches validos e mover-se para um deles.

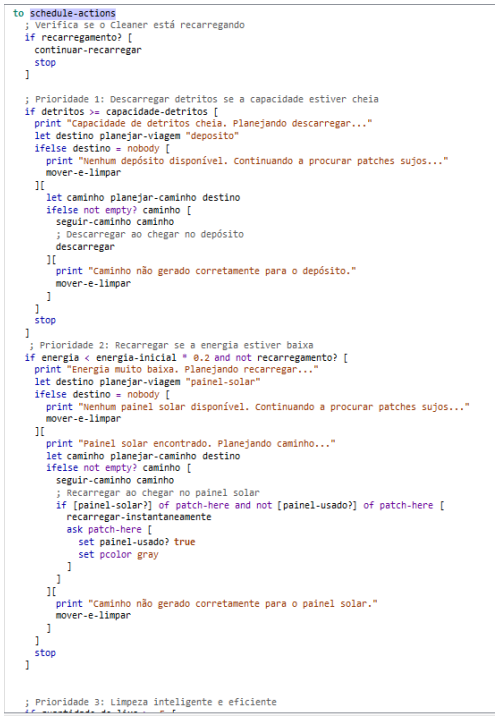
```
to andar-explorar
  ; Ajusta o raio de exploração baseado na energia restante
  let raio-exploracao 5
  if energia < energia-inicial * 0.5 [
    print "Energia abaixo de 50%. Reduzindo raio de exploração para 3."
    set raio-exploracao 3
  ]

  ; O Cleaner examina os patches ao seu redor dentro do raio ajustado
  let patches-visiveis patches in-radius raio-exploracao
  print (word "Patches visíveis no raio de " raio-exploracao ": " count patches-visiveis)

  ; Seleciona patches que não estão na lista de patches recentes e que podem estar sujos
  let patches-validos patches-visiveis with [
    not member? (list pxcor pycor) patches-recentes
    and not deposito?
    and not painel-solar?
  ]
  print (word "Patches válidos encontrados: " count patches-validos)


  ; Se encontrar patches válidos, move-se para um deles, senão move-se aleatoriamente
  ifelse any? patches-validos [
    ; Se encontrar patches válidos, seleciona um e move-se para ele
    let patch-alvo one-of patches-validos
    print (word "Patch alvo encontrado em: (" [pxcor] of patch-alvo "," [pycor] of patch-alvo ")")

    face patch-alvo
    safe-move 1
    print (word "Movendo para patch: (" [pxcor] of patch-here "," [pycor] of patch-here ")")

    ; Atualiza a memória do patch atual
    let patch-atual patch-here
    let tempo-desde-limpeza ticks - [tempo-limpo] of patch-atual
    set patches-recentes lput (list [pxcor] of patch-atual [pycor] of patch-atual) patches-recentes
    print (word "Memória atualizada: Patch (" [pxcor] of patch-atual "," [pycor] of patch-atual ") visitado.")

  ] [
    ; Se não encontrar patches válidos, move-se aleatoriamente
    print "Nenhum patch estratégico encontrado, movendo-se aleatoriamente..."
    right random 360
    safe-move 1
    print (word "Movendo-se aleatoriamente para: (" [pxcor] of patch-here "," [pycor] of patch-here ")")
  ]
        set energia energia - 1

  ; Verifica a energia e decide se é necessário recarregar ou continuar explorando
  ifelse energia <= 0 [
    print "Energia esgotada durante a exploração. Iniciando recarga..."
    recarregar
  ] [
    print (word "Energia restante: " energia)
  ]
end
```

Este Código é capaz fazer com que o cleaner se mova para o painel solar para recarregar, que interage com o Painel Solar, que tenha memoria do patch atual e que faça limpeza dos patches sujos.



```
; Comportamento do Cleaner: mover-se, limpar e descarregar
to mover-e-limpar
  if not recarregamento? [
    ; Verifica se há painéis solares no alcance
    let painel-alvo one-of patches in-radius 3 with [painel-solar? and not painel-usado?]

    ; Se houver um painel solar no alcance, vai para ele e recarrega
    if painel-alvo != nobody [
      face painel-alvo
      safe-move 0.8

      ; Verifica se chegou ao painel solar
      if [painel-solar?] of patch-here and not [painel-usado?] of patch-here [
        recarregar-instantaneamente
        ask patch-here [
          set painel-usado? true  ; Marca o painel como utilizado
          set pcolor gray         ; Muda a cor para indicar que está inativo
        ]
      ]
      stop
    ]

    ; Memória do patch atual
    let patch-atual patch-here
    let tempo-desde-limpeza ticks - [tempo-limpo] of patch-atual
    let patches-atuais filter [p -> ticks - last p < 50] patches-recentes
    set patches-recentes patches-atuais

    ; Atualiza a memória do patch atual
    let patch-na-memoria filter [m -> first m = patch-atual] patches-recentes
    ifelse not empty? patch-na-memoria [
      let posicao position first patch-na-memoria patches-recentes
      set patches-recentes replace-item posicao patches-recentes (list patch-atual [sujo?] of patch-atual tempo-desde-limpeza)
    ][
      set patches-recentes lput (list patch-atual [sujo?] of patch-atual tempo-desde-limpeza) patches-recentes
    ]

    ; Otimização da busca de lixo
    let patches-vistos patches in-radius 5  ; Patches visíveis em um raio maior
    let patches-sujos patches-vistos with [sujo? and not member? self patches-recentes]  ; Patches sujos não recentemente limpos

    ifelse any? patches-sujos [
      ; Prioriza patches sujos com base no tempo de lixo (mais antigos primeiro)
      let patch-alvo max-one-of patches-sujos [tempo-limpo]
      face patch-alvo
      safe-move 1

      ; Verifica se o patch tem lixo pesada ou tóxica
      if [sujo?] of patch-alvo [
        if [tipo-de-lixo] of patch-alvo = "pesada" [
          set energia energia - (energia-inicial * 0.25)  ; Penaliza o Cleaner por passar em lixo pesada
          print (word "Passou por lixo pesada. Energia reduzida em 25%. Energia atual: " energia)
        ]
        if [tipo-de-lixo] of patch-alvo = "toxica" [
          set energia energia - (energia-inicial * 0.5)
          print (word "Passou por lixo tóxica. Energia reduzida em 50%. Energia atual: " energia)
```

Este Código verifica se o cleaner esta recarregado, onde este descarrega os detritos se a capacidade estiver cheia e recarrega a bateria do cleaner



```
to schedule-actions
  ; Verifica se o Cleaner está recarregando
  if recarregamento? [
    continuar-recarregar
    stop
  ]

  ; Prioridade 1: Descarregar detritos se a capacidade estiver cheia
  if detritos >= capacidade-detritos [
    print "Capacidade de detritos cheia. Planejando descarregar..."
    let destino planejar-viagem "deposito"
    ifelse destino = nobody [
      print "Nenhum depósito disponível. Continuando a procurar patches sujos..."
      mover-e-limpar
    ][
      let caminho planejar-caminho destino
      ifelse not empty? caminho [
        seguir-caminho caminho
        ; Descarregar ao chegar no depósito
        descarregar
      ][
        print "Caminho não gerado corretamente para o depósito."
        mover-e-limpar
      ]
    ]
    stop
  ]
   ; Prioridade 2: Recarregar se a energia estiver baixa
  if energia < energia-inicial * 0.2 and not recarregamento? [
    print "Energia muito baixa. Planejando recarregar..."
    let destino planejar-viagem "painel-solar"
    ifelse destino = nobody [
      print "Nenhum painel solar disponível. Continuando a procurar patches sujos..."
      mover-e-limpar
    ][
      print "Painel solar encontrado. Planejando caminho..."
      let caminho planejar-caminho destino
      ifelse not empty? caminho [
        seguir-caminho caminho
        ; Recarregar ao chegar no painel solar
        if [painel-solar?] of patch-here and not [painel-usado?] of patch-here [
          recarregar-instantaneamente
          ask patch-here [
            set painel-usado? true
            set pcolor gray
          ]
        ]
      ][
        print "Caminho não gerado corretamente para o painel solar."
        mover-e-limpar
      ]
    ]
    stop
  ]


  ; Prioridade 3: Limpeza inteligente e eficiente
```

Este Código faz a Movimentação para patches limpos e que não são depósitos, se não houverem patches limpos move-se aleatoriamente, os polluters também depositam com uma probabilidade definida pelos sliders.

```
  ]
end
to mover-e-poluente
  ; Encontrar patches ao redor que não estão sujos e que não são depósitos
  let patches-limpos patches in-radius 3 with [not sujo? and not deposito? ]

  ifelse any? patches-limpos [
    ; Se houver patches limpos (e sem depósitos), mover-se para um deles
    let patch-alvo one-of patches-limpos
    face patch-alvo
    fd 0.8
  ] [
    ; Se não houver patches limpos, mover-se aleatoriamente, mas evitar depósitos
    let patches-validos patches in-radius 3 with [not deposito?]
    ifelse any? patches-validos [
      let patch-aleatorio one-of patches-validos
      face patch-aleatorio
      fd 0.8
    ] [
      right random 360
      fd 0.8
    ]
  ]
  ; Deposição de lixo com probabilidade definida pelos sliders
  if not [sujo?] of patch-here and not [deposito?] of patch-here [  ; Verifica se o patch não é depósito
    let chance-deposit random-float 1
    if chance-deposit < probabilidade-depositos [
      ask patch-here [
        set sujo? true
        set tipo-de-lixo [tipo-de-poluicao] of myself
        if tipo-de-lixo = "leve" [
          set pcolor yellow  ; Poluição leve
        ]
        if tipo-de-lixo = "pesada" [
          set pcolor brown   ; Poluição pesada
        ]
        if tipo-de-lixo = "toxica" [
          set pcolor green   ; Poluição tóxica
        ]
      ]
    ]
  ]
end
```