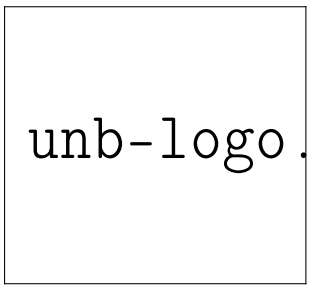

CIC0203 - Computação Experimental - TA - 2022.2 - Tarefa T6 - Aprimoramento de uma Simulação

URL Read-only Overleaf: https://www.overleaf.com/read/vttbsjhdybcd

André Cássio Barros de Souza (andreloff)

Brasília, 2023-02-01 02:33:50Z

Lista de tarefas pendentes

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este documento contém o produto da tarefa especificada no título deste documento, conforme as orientações em https://www.overleaf.com/read/cytswcjsxxqh.

Parte I

Preparação

Parte II

Simulação Computacional

Capítulo 1

T6 - Aprimoramento de uma Simulação: Laboratório e Experimento Predator-Prey, por André Cássio (andreloff)

1.1 Introdução

Este capítulo apresenta a construção e uso do laboratório de simulações Predator-Prey para a realização de experimentos que tem por objetivo investigar a hipótese causal de que ovelhas com comportamento alerta tendem a sobreviver, que relaciona variáveis independentes e variáveis dependentes, supostamente presente nos estudos bibliométricos por mim realizados e disponíveis em **??**.

É composto por mais cinco seções:

1. Descrição do fenômeno real;
2. Apresentação do laboratório de simulações;
3. Apresentação de análises exploratórias dos dados de experimentos realizados com o uso do laboratório;
4. Discussão sobre *insights* obtidos após os experimentos; e
5. Conclusões.

1.2 O Fenômeno do Mundo Real

O fenômeno da relação presa-predador, ou predação, é de natureza biológica e ecológica, que consiste em um organismo matar e consumir (predador) um outro organismo (presa). É fácil

perceber essa relação no mundo real, já que em quase todo ecossistema existem pelo menos alguma relação de predação.

1.3 O Laboratório Predator-Prey

O laboratório Predator-Prey se trata da simulação multi-agente de um micro ecossitema que possui um agente do tipo predador e um agente do tipo presa. A finalidade é entender melhor a dinâmica das populações dos agentes.

1.3.1 O Conceito da Simulação

O código usado no laboratório pode ser acessado a partir do framework Mesa, que possibilita a rápida modelagem e criação de simulações multiagentes. O código fonte utilizado é uma variação de um dos vários exemplos disponibilizados pelos desenvolvedores, que pode ser obtido pelo GitHub ou pelo próprio site. Pela própria natureza de um ecossistema, é de extrema dificuldade criar uma simulação que leve em consideração as variáveis necessárias para descrever relações entre espécies diferentes, sendo uma delas a predação. O aprimoramento criado nesse laboratório tenta aproximar um pouco da realidade o que seria essa relação.

1.3.2 O Simulador

O framework Mesa possibilta a visualização da simulação por meio de uma interface gráfica, na qual o usuário pode modificar variáveis (com um controle especificado pelo criador do modelo) e executar a simulação com estados iniciais diferentes.

1.3.2.1 Variáveis Independentes ou de Controle

São as seguintes as variáveis Independentes ou de Controle, manipuláveis na interface gráfica do simulador:

Grass Enabled Diz se a simulação criará células de grama para que as ovelhas se alimentem.

Grass Regrowth Time Tempo que levará uma célula que teve a grama consumida crescer.

Initial Sheep Population População inicial de ovelhas.

Sheep Reproduction Rate Taxa de reprodução das ovelhas, que ocorre de forma assexual. Probabilidade de uma ovelha gerar um novo indivíduo.

Initial Wolf Population População inicial de lobos.

Wolf Reproduction Rate Taxa de reprodução dos lobos, que ocorre de forma assexual. Probabilidade de um lobo gerar um novo indivíduo.

Wolf Gain From Food Rate Quantidade de energia que o lobo ganha ao comer uma ovelha.

Sheep Gain From Food Quantidade de energia que a ovelha ganha ao comer uma célula de grama.

Sheep Alert Behaviour Chance Probabilidade de cada ovelha ter um comportamento alerta no próximo passo.

1.3.2.2 Variáveis Dependentes

São as seguintes as variáveis Dependentes, cujos valores são coletados e apresentados na interface gráfica do simulador:

Wolves Quantidade, a cada iteração da simulação, da população de lobos.

Sheep Quantidade, a cada iteração da simulação, da população de ovelhas.

1.3.3 A Hipótese Causal

No código fonte original, como será mostrado, o comportamento adotado pelos agentes lobo e ovelha é escolher uma célula aleatória possível e andar para essa posição. Analisando como é apresentada a simulação, a hipótese causal é tentar criar uma relação entre o crescimento na probabilidade de um comportamento mais alerta das ovelhas e tentar mostrar como isso afeta em uma maior taxa de sobrevivência da espécie.

1.3.4 O Código do Simulador

1.3.4.1 Agentes

Listagem de Código 1.1: Código da Criação e Comportamento do Agente Ovelha.

```
5   class Sheep(RandomWalker):
6       energy = None
7
8       def __init__(self, unique_id, pos, model, moore, energy=None):
9           super().__init__(unique_id, pos, model, moore=moore)
10          self.energy = energy
11
12      def alertStep(self):
13          possible_moves = self.model.grid.get_neighborhood(self.pos, self.moore, True)
14          alert_moves = self.model.grid.get_neighborhood(self.pos, self.moore, True)
15          for neighborCell in possible_moves:
16              neighborCellContent = self.model.grid.get_cell_list_contents([neighborCell])
17              wolves = [obj for obj in neighborCellContent if isinstance(obj, Wolf)]
18              if len(wolves) > 0:
19                  wolf_possible_moves = self.model.grid.get_neighborhood(wolves[0].pos, wolves[0].moore, True)
20                  for wolfPossibleMovesCells in wolf_possible_moves:
21                      if wolfPossibleMovesCells in alert_moves:
22                          alert_moves.remove(wolfPossibleMovesCells)
23          next_move = []
24          if len(alert_moves) > 0:
25              next_move = self.random.choice(alert_moves)
26          else:
27              next_move = self.random.choice(possible_moves)
28          self.model.grid.move_agent(self, next_move)
29
```

```
30      def step(self):
31          if self.random.random() < self.model.sheep_alert_behaviour_chance:
32              self.alertStep()
33          else:
34              self.random_move()
35
36          living = True
37
38          if self.model.grass:
39              # Reduce energy
40              self.energy -= 1
41
42              # If there is grass available, eat it
43              this_cell = self.model.grid.get_cell_list_contents([self.pos])
44              grass_patch = [obj for obj in this_cell if isinstance(obj, GrassPatch)][0]
45              if grass_patch.fully_grown:
46                  self.energy += self.model.sheep_gain_from_food
47                  grass_patch.fully_grown = False
48
49              # Death
50              if self.energy < 0:
51                  self.model.grid._remove_agent(self.pos, self)
52                  self.model.schedule.remove(self)
53                  living = False
54
55          if living and self.random.random() < self.model.sheep_reproduce:
56              # Create a new sheep:
57              if self.model.grass:
58                  self.energy /= 2
59              lamb = Sheep(
60                  self.model.next_id(), self.pos, self.model, self.moore, self.energy
61              )
62              self.model.grid.place_agent(lamb, self.pos)
63              self.model.schedule.add(lamb)
```

Listagem de Código 1.2: Código da Criação e Comportamento do Agente Lobo.

```
66  class Wolf(RandomWalker):
67      energy = None
68
69      def __init__(self, unique_id, pos, model, moore, energy=None):
70          super().__init__(unique_id, pos, model, moore=moore)
71          self.energy = energy
72
73      def step(self):
74          self.random_move()
75          self.energy -= 1
76
77          # If there are sheep present, eat one
78          x, y = self.pos
79          this_cell = self.model.grid.get_cell_list_contents([self.pos])
80          sheep = [obj for obj in this_cell if isinstance(obj, Sheep)]
81          if len(sheep) > 0:
82              sheep_to_eat = self.random.choice(sheep)
83              self.energy += self.model.wolf_gain_from_food
84
85              # Kill the sheep
86              self.model.grid._remove_agent(self.pos, sheep_to_eat)
87              self.model.schedule.remove(sheep_to_eat)
88
89          # Death or reproduction
90          if self.energy < 0:
91              self.model.grid._remove_agent(self.pos, self)
92              self.model.schedule.remove(self)
93          else:
94              if self.random.random() < self.model.wolf_reproduce:
95                  # Create a new wolf cub
96                  self.energy /= 2
97                  cub = Wolf(
98                      self.model.next_id(), self.pos, self.model, self.moore, self.energy
99                  )
100                 self.model.grid.place_agent(cub, cub.pos)
101                 self.model.schedule.add(cub)
```

Listagem de Código 1.3: Código da Criação e Comportamento do Agente Grama.

```
104 class GrassPatch(Agent):
105
106     def __init__(self, unique_id, pos, model, fully_grown, countdown):
107         super().__init__(unique_id, model)
108         self.fully_grown = fully_grown
109         self.countdown = countdown
```

```
110         self.pos = pos
111
112     def step(self):
113         if not self.fully_grown:
114             if self.countdown <= 0:
115                 # Set as fully grown
116                 self.fully_grown = True
117                 self.countdown = self.model.grass_regrowth_time
118             else:
119                 self.countdown -= 1
```

1.3.4.2 Modelo

Listagem de Código 1.4: Código da Criação do Modelo.

```
20  class WolfSheep(Model):
21      """
22      Wolf-Sheep Predation Model
23      """
24
25      height = 20
26      width = 20
27
28      initial_sheep = 100
29      initial_wolves = 50
30
31      sheep_reproduce = 0.04
32      wolf_reproduce = 0.05
33
34      wolf_gain_from_food = 20
35
36      grass = False
37      grass_regrowth_time = 30
38      sheep_gain_from_food = 4
39      sheep_alert_behaviour_chance = 0.5
40
41      verbose = False  # Print-monitoring
42
43      description = (
44          "A model for simulating wolf and sheep (predator-prey) ecosystem modelling."
45      )
46
47      def __init__(
48          self,
49          height=20,
50          width=20,
51          initial_sheep=100,
52          initial_wolves=50,
53          sheep_reproduce=0.04,
54          wolf_reproduce=0.05,
55          wolf_gain_from_food=20,
56          grass=False,
57          grass_regrowth_time=30,
58          sheep_gain_from_food=4,
59          sheep_alert_behaviour_chance=0.5,
60      ):
61          """
62          Create a new Wolf-Sheep model with the given parameters.
63
64          Args:
65              initial_sheep: Number of sheep to start with
66              initial_wolves: Number of wolves to start with
67              sheep_reproduce: Probability of each sheep reproducing each step
68              wolf_reproduce: Probability of each wolf reproducing each step
69              wolf_gain_from_food: Energy a wolf gains from eating a sheep
70              grass: Whether to have the sheep eat grass for energy
71              grass_regrowth_time: How long it takes for a grass patch to regrow
72                                   once it is eaten
73              sheep_gain_from_food: Energy sheep gain from grass, if enabled.
74              sheep_alert_behaviour_chance: Probability of each sheep make a alerted behaviour in next step
75          """
76          super().__init__()
77          # Set parameters
78          self.height = height
79          self.width = width
80          self.initial_sheep = initial_sheep
81          self.initial_wolves = initial_wolves
82          self.sheep_reproduce = sheep_reproduce
83          self.wolf_reproduce = wolf_reproduce
84          self.wolf_gain_from_food = wolf_gain_from_food
85          self.grass = grass
86          self.grass_regrowth_time = grass_regrowth_time
```

```
87          self.sheep_gain_from_food = sheep_gain_from_food
88          self.sheep_alert_behaviour_chance = sheep_alert_behaviour_chance
89
90          self.schedule = RandomActivationByBreed(self)
91          self.grid = MultiGrid(self.height, self.width, torus=True)
92          self.datacollector = DataCollector(
93              {
94                  "Wolves": lambda m: m.schedule.get_breed_count(Wolf),
95                  "Sheep": lambda m: m.schedule.get_breed_count(Sheep),
96              }
97          )
98
99          # Create sheep:
100         for i in range(self.initial_sheep):
101             x = self.random.randrange(self.width)
102             y = self.random.randrange(self.height)
103             energy = self.random.randrange(2 * self.sheep_gain_from_food)
104             sheep = Sheep(self.next_id(), (x, y), self, True, energy)
105             self.grid.place_agent(sheep, (x, y))
106             self.schedule.add(sheep)
107
108         # Create wolves
109         for i in range(self.initial_wolves):
110             x = self.random.randrange(self.width)
111             y = self.random.randrange(self.height)
112             energy = self.random.randrange(2 * self.wolf_gain_from_food)
113             wolf = Wolf(self.next_id(), (x, y), self, True, energy)
114             self.grid.place_agent(wolf, (x, y))
115             self.schedule.add(wolf)
116
117         # Create grass patches
118         if self.grass:
119             for agent, x, y in self.grid.coord_iter():
120
121                 fully_grown = self.random.choice([True, False])
122
123                 if fully_grown:
124                     countdown = self.grass_regrowth_time
125                 else:
126                     countdown = self.random.randrange(self.grass_regrowth_time)
127
128                 patch = GrassPatch(self.next_id(), (x, y), self, fully_grown, countdown)
129                 self.grid.place_agent(patch, (x, y))
130                 self.schedule.add(patch)
131
132         self.running = True
133         self.datacollector.collect(self)
134
135     def step(self):
136         self.schedule.step()
137         # collect data
138         self.datacollector.collect(self)
139         if self.verbose:
140             print(
141                 [
142                     self.schedule.time,
143                     self.schedule.get_breed_count(Wolf),
144                     self.schedule.get_breed_count(Sheep),
145                 ]
146             )
147
148     def run_model(self, step_count=200):
149
150         if self.verbose:
151             print("Initial number wolves: ", self.schedule.get_breed_count(Wolf))
152             print("Initial number sheep: ", self.schedule.get_breed_count(Sheep))
153
154         for i in range(step_count):
155             self.step()
156
157         if self.verbose:
158             print("")
159             print("Final number wolves: ", self.schedule.get_breed_count(Wolf))
160             print("Final number sheep: ", self.schedule.get_breed_count(Sheep))
```

1.4 Os Experimentos Realizados

Para testar a hipótese causal, a simulação foi executada com algumas variáveis independentes fixadas e fazendo a manipulação da variável de probabilidade de cada ovelha assumir um comportamento alerta no seu próximo passo, e verificar se, em algum momento, a população

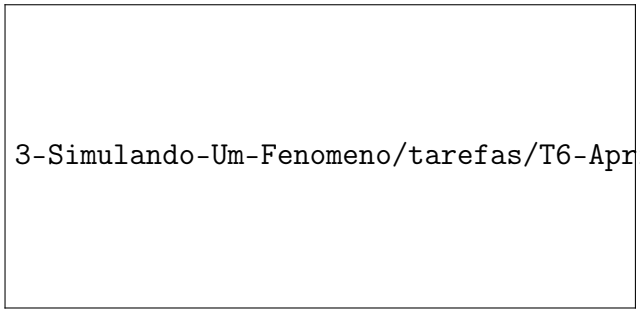

Figura 1.1: Gráfico que mostra a população de ovelhas com chance de alerta 100% caindo para 0.

de lobos foi para zero enquanto havia pelo menos uma ovelha. As variáveis independetes a seguir foram fixadas com os valores:

Grass Enabled False.

Grass Regrowth Time 20.

Initial Sheep Population 100.

Sheep Reproduction Rate 0,04.

Initial Wolf Population 50.

Wolf Reproduction Rate 0,05.

Wolf Gain From Food Rate 20.

Sheep Gain From Food 4.

Sendo que a variável Sheep Alert Behaviour Chance será modificada para análise da simulação.
Primeiro a simulação foi executada 10 vezes com valor 0. Sendo que 4 a ovelha saiu como espécie sobrevivente e em 6 delas a população de lobos consumiu toda a população de ovelhas. O mesmo aconteceu com a variável no valor de 0,5. Executando 10 vezes com a chance de ter comportamento alerta de 100%, a população de ovelhas sobreviveu em 6 das execuções.

1.5 Discussão e *insights* preliminares sobre as hipóteses

Com as variáveis independentes escolhidas fixadas e na mudança na probabilidade de adotar um comportamento alerta, a simulação apresentou uma pequena mudança na situação final de

se a espécie sobreviveram ou foram consumidas, apesar de não possuir muita confiança, a hipótese causal parece ter sido comprovada. Analisando a situação geral de um ecossistema, faz sentido espécies que possuem um comportamento mais voltado para a sobrevivência tenham mais tempo de vida.

1.6 Conclusão

Ter acesso a uma ferramenta visual para execução de simulações ajudou de forma muito significativa para a escolha de qual aprimoramento fazer, analisando o contexto da simulação e comparando com o fenômeno na vida real. A fácil edição nos valores das variáveis independentes também auxiliou no entendimento da dinâmica na relação entre os agentes. Porém com poucas amostras do experimento, é difícil dizer com certa confiança se a hipótese causal foi refutada ou não.