UnB

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

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

João Victor de Souza Calassio (jvcalassio)

Brasília, 2023-01-06 23:27:49Z

Lista de tarefas pendentes

Sumário

Lista de Figuras

Lista de Tabelas

Parte I

Simulação Computacional

Capítulo 1

T6 - Aprimoramento de uma Simulação: Laboratório e Experimento pd_grid, por João Victor de Souza Calassio (jvcalassio)

1.1 Introdução

Este capítulo apresenta a construção e uso do laboratório de simulações pd_grid para a realização de experimentos que tem por objetivo investigar a hipótese causal "o aumento no número de agentes cooperantes aumenta proporcionalmente o ganho para o sistema como um todo" que relaciona variáveis independentes e variáveis dependentes, supostamente presente nos estudos bibliométricos por mim realizados.

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 dilema do prisioneiro (PRISONER'S..., 2022) é uma modelagem da interação entre dois agentes, cada um buscando maximizar individualmente o valor recebido nessa interação.

Tabela 1.1: Tabela de resultados da situação imaginada.

		A	
		Reduzir preço	**Não reduzir preço**
B	**Reduzir preço**	+ R$70.000 + R$70.000	- R$10.000 + R$160.000
	Não reduzir preço	+ R$160.000 - R$10.000	+ R$100.000 + R$100.000

Tabela 1.2: Tabela de resultados para o dilema do prisioneiro iterativo com mais de dois agentes.

	Cooperar	**Desertar**
Cooperar	1; 1	0; D
Desertar	D; 0	0; 0

No mundo real, o dilema do prisioneiro pode ser exemplificado com a competição entre duas empresas: imagine que existam duas empresas A e B que vendem árvores de natal, e ambas estão tentando decidir (individualmente, sem comunicação entre si) se devem reduzir os preços em dezembro de 2022.

Se nenhuma delas reduzir os preços, continuarão a vender igualmente e terão um lucro imaginário de R$100.000. Se ambas reduzirem os preços, ainda venderão igualmente mas a uma margem de lucro mais baixa, e portanto terão um lucro imaginário de R$70.000. Se apenas uma reduzir os preços, a que tomar essa decisão venderá mais árvores e terá um lucro de R$160.000, e a que não aumentar os preços venderá muito menos e terá prejuízos. Esses resultados podem ser representados pela tabela de resultados 1.1.

No dilema do prisioneiro **iterativo** com mais de dois agentes e com múltiplas rodadas, os agentes que não cooperam são beneficiados por uma variável independente D (que seria o "prêmio por deserção") multiplicada pela quantidade de agentes que cooperam. A propagação da cooperação seria diretamente dependente desse valor D, e teríamos uma matriz de propagação como mostra a tabela 1.2. Após as múltiplas rodadas de interação entre os agentes, conseguimos observar qual a dinâmica da propagação da cooperação.

1.3 O Laboratório pd_grid

O laboratório pd_grid é uma implementação do dilema do prisioneiro iterativo, conhecido como "dilema do prisioneiro demográfico". Demográfico porque os agentes envolvidos nas interações estão dispostos em um *grid*, e tomam as decisões de cooperar/desertar baseado nas ações de seus vizinhos.

Tabela 1.3: Tabela de resultados presente na simulação do MESA originalmente.

	Cooperar	**Desertar**
Cooperar	1; 1	0; 1,6
Desertar	1,6; 0	0; 0

1.3.1 O Conceito da Simulação

O laboratório pd_grid presente no Python/MESA é uma implementação do dilema do prisioneiro desenvolvido por (EPSTEIN, 1998), e uma adaptação do modelo incluso em (WILENSKY, 2002).

Nesse modelo, um agente qualquer está cercado por até oito agentes diferentes, e deve tomar a decisão de cooperar (C) ou desertar (D) com todos eles. A cada rodada, o agente verifica as pontuações de seus vizinhos e decide qual será sua ação baseado na ação tomada pelo agente com maior pontuação. Essa ação é utilizada para interagir com todos os seus vizinhos na rodada corrente, e a pontuação recebida por essa rodada é decidida através da matriz de resultados 1.3.

Através dessa simulação, é possível observar como a cooperação pode ser propagada entre os agentes, por mais que a ação de desertar seja dominante em cada interação individual entre dois agentes.

Também é interessante notar que a forma com que as interações são realizadas tem um papel relevante nos resultados e na propagação da cooperação. Esses pontos são tratados na próxima seção.

1.3.2 O Simulador

No simulador original, há apenas uma variável de controle que define como a interação entre os agentes é feita.

Foram adicionadas mais duas variáveis independentes, que definem qual a ação inicial dos agentes e um modificador para o prêmio de deserção.

1.3.2.1 Variáveis Independentes ou de Controle

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

Modo de ativação Seletor para escolher qual será a forma em que os agentes irão interagir entre si: sequencialmente (na ordem que foram inseridos), aleatoriamente ou simultaneamente.

Cooperação inicial Porcentagem de agentes cooperantes inicial.

Prêmio de deserção Pontuação recebida pelo agente caso ele escolha desertar e o outro agente envolvido na interação escolha cooperar.

1.3.2.2 Variáveis Dependentes

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

Escolha do agente Indica visualmente se o agente escolheu cooperar (azul no grid) ou desertar (vermelho no grid).

Valor total do sistema Soma das pontuações de todos os agentes no sistema ao longo do tempo.

1.3.3 A Hipótese Causal

A hipótese causal proposta para esse experimento afirma que quanto mais agentes cooperantes houverem no sistema, maior será o benefício para o sistema e consequentemente, será gerado mais valor ao longo do tempo. Em outras palavras, a estratégia cooperante é muito mais benéfica no longo prazo, mesmo que o prêmio por deserção seja muito alto.

Essa hipótese é motivada pelo estudo original abordado na simulação por (WILENSKY, 2002) e (EPSTEIN, 1998), que busca mostrar como a cooperação consegue ser propagada entre os agentes, por mais que desertar seja a melhor estratégia individual.

1.3.4 O Código do Simulador

O código presente no laboratório utiliza o framework MESA para implementar uma simulação do dilema do prisioneiro iterativo com mais de dois jogadores em um *grid*.

O código é composto por duas classes principais:

PDAgent Define quais variáveis cada agente possui, e qual comportamento esse agente vai tomar a cada iteração.

As variáveis de cada agente são: posição (X, Y) no grid; pontuação total; movimento escolhido (cooperar ou desertar); próximo movimento (cooperar ou desertar). Conforme mostra o código 1.1, valores iniciais são recebidos por parâmetro, exceto para o score que sempre começa em zero.

Conforme mostra o código 1.2, o comportamento do agente a cada iteração consiste em verificar a pontuação de todos os seus vizinhos e então copiar o movimento do agente vizinho que tenha a maior pontuação.

Listagem de Código 1.1: Inicialização de um agente

```
1       def __init__(self, pos, model, starting_move=None):
2           """
3           Create a new Prisoner's Dilemma agent.
4
5           Args:
6               pos: (x, y) tuple of the agent's position.
7               model: model instance
8               starting_move: If provided, determines the agent's initial state:
9                           C(ooperating) or D(efecting). Otherwise, random.
```

```
        """
        super().__init__(pos, model)
        self.pos = pos
        self.score = 0
        if starting_move:
            self.move = starting_move
        else:
            self.move = self.random.choice(["C", "D"])
        self.next_move = None
```

Listagem de Código 1.2: Ação realizada pelos agentes a cada iteração.

```
    def step(self):
        """Get the neighbors' moves, and change own move accordingly."""
        neighbors = self.model.grid.get_neighbors(self.pos, True, include_center=True)
        best_neighbor = max(neighbors, key=lambda a: a.score)
        self.next_move = best_neighbor.move

        if self.model.schedule_type != "Simultaneous":
            self.advance()

    def advance(self):
        self.move = self.next_move
        self.score += self.increment_score()
```

PdGrid Descreve o sistema em que os agentes estão inseridos, cria os agentes iniciais e recebe as variáveis de controle do modelo.

O sistema possui 50 x 50 agentes, que são criados na inicialização do sistema e têm o movimento inicial escolhido aleatoriamente à partir das probabilidades definidas pelo usuário, como mostra o código 1.3

Em cada iteração do sistema, é captada a soma das pontuações de todos os agentes no sistema para conseguirmos metrificar a evolução, como mostrado nos códigos 1.4 e 1.5.

Listagem de Código 1.3: Criação dos agentes.

```
    for x in range(width):
        for y in range(height):
            agent = PDAgent(
                (x, y),
                self,
                self.random.choices(["C", "D"], weights=[initial_cooperation, 1-initial_cooperation])[0]
            )
            self.grid.place_agent(agent, (x, y))
            self.schedule.add(agent)
```

Listagem de Código 1.4: Captura das somas das pontuações dos agentes.

```
    self.datacollector = DataCollector(
        {
            "Cooperating_Agents": lambda m: len(
                [a for a in m.schedule.agents if a.move == "C"]
            ),
            "Defecting_Agents": lambda m: len(
                [a for a in m.schedule.agents if a.move == "D"]
            ),
            "Total_value": lambda m: sum(
                [agent.score for agent in m.schedule.agents]
            )
        }
    )
```

Listagem de Código 1.5: Ação realizada pelo sistema a cada iteração.

```
    def step(self):
        self.schedule.step()
        # collect data
        self.datacollector.collect(self)
```

1.4 Os Experimentos Realizados

Foram realizados diversos experimentos modificando as possíveis variáveis de controle para tentar verificar se existe alguma relação entre o número de agentes cooperantes que restam no sistema e a pontuação total do sistema após algum número de rodadas.

1.4.0.1 Valores padrão e modo de ativação

Por padrão, o modo de ativação do modelo é "aleatório", a taxa de agentes cooperantes inicialmente é de 50%, e o prêmio por desertar é 1,6.

Foram realizados três testes com essa configuração, alterando apenas o modo de ativação. Em todas elas o número de agentes desertores sempre começava "ganhando" nos primeiros três ciclos, chegando a somar cerca de 2300 desertores em média. No entanto, após alguns ciclos (de 20 a 30 ciclos), os agentes cooperantes que restavam começavam a dominar o sistema e se tornavam maioria, somando 2333 cooperantes em média (e em alguns casos até se tornando o comportamento de todos os 2500 agentes) e atingindo estabilidade. Um exemplo pode ser observado na figura 1.1.

Alterações no modo de ativação alteram a velocidade com que essa mudança de comportamento é realizada (comparando as figuras 1.1 e 1.2). O modo de ativação simultâneo, no entanto, sempre levava à dominação completa de uma das escolhas.

É possível observar que nos casos em que restaram mais agentes cooperantes, a pontuação foi mais de 100 vezes maior. Isso acontece pois após a dominação dos agentes desertores a pontuação se mantém constante durante toda a execução, conforme mostra a figura 1.3.

1.4.0.2 Alterações no prêmio de deserção

Naturalmente, um próximo experimento é verificar como a alteração do prêmio de deserção influencia na pontuação do sistema. Se o prêmio for muito alto, há alguma chance de gerar valor maior para o sistema?

Foram realizadas três execuções, uma com a variável "prêmio de deserção" em 2.0 (o dobro do valor recebido pela cooperação mútua), uma em 4.0 e uma em 8.0. Notou-se que quando o "prêmio de deserção" é muito alto, há pouca chance de sobrevivência do comportamento cooperativo, e a deserção domina completamente o sistema.

Se compararmos a figura 1.4 à figura 1.2, podemos notar que um prêmio de deserção maior na verdade diminuiu o valor total do sistema, pois favorece o comportamento de deserção. Se o comportamento de cooperação for completamente extinguido o valor se mantém constante pois o sistema é incapaz de gerar valor após alguns ciclos, e é o que aconteceu com qualquer execução com "prêmio de deserção" maior que 2.0, como exemplificado na figura 1.5.

1.4.0.3 Alterações no número de agentes cooperantes iniciais

Por fim, foi testado como a escolha inicial dos agentes interfere na evolução do sistema. É possível que caso hajam mais agentes cooperantes no início do sistema, o comportamento

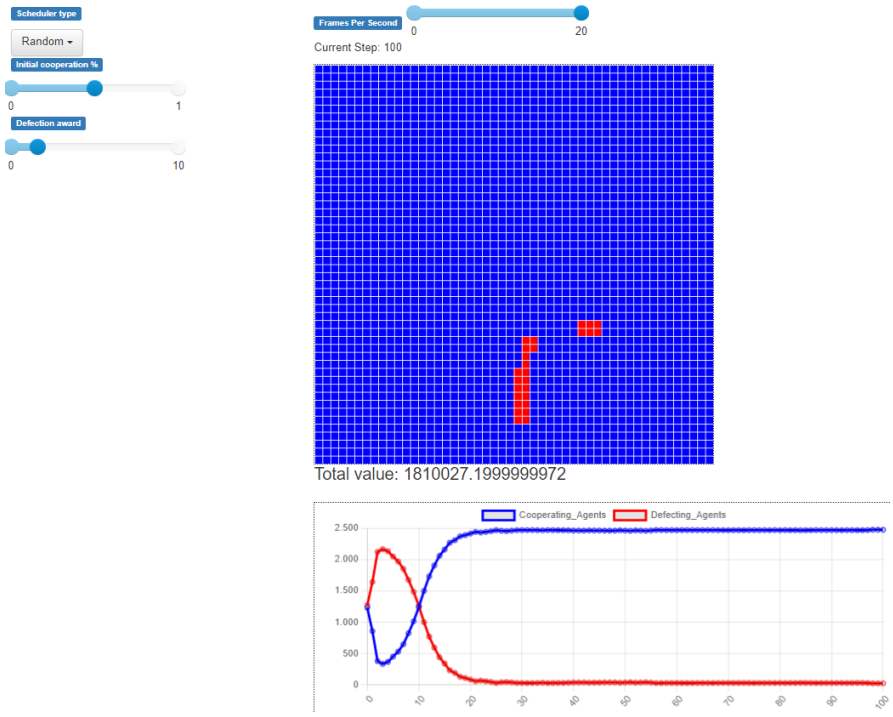

Figura 1.1: Exemplo de experimento com as configurações padrão

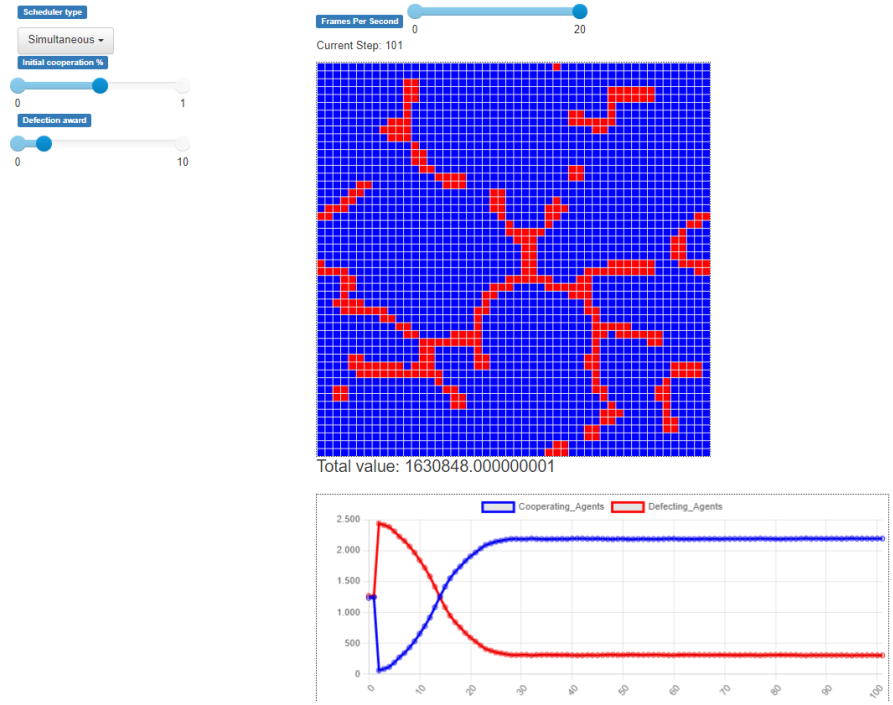

Figura 1.2: Exemplo de experimento com modo de ativação simultâneo

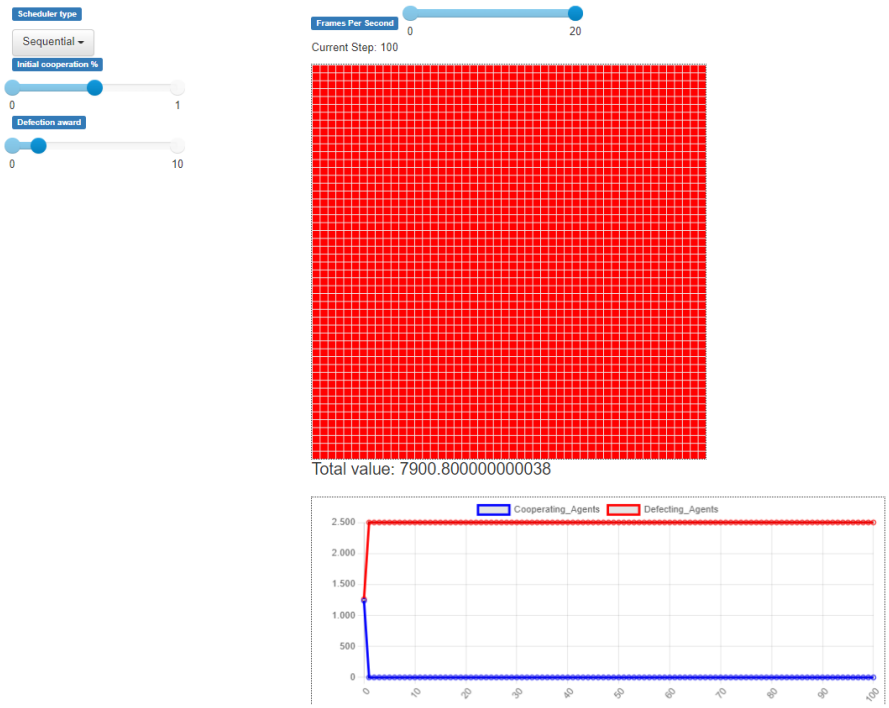

Figura 1.3: Exemplo de experimento com modo de ativação sequencial

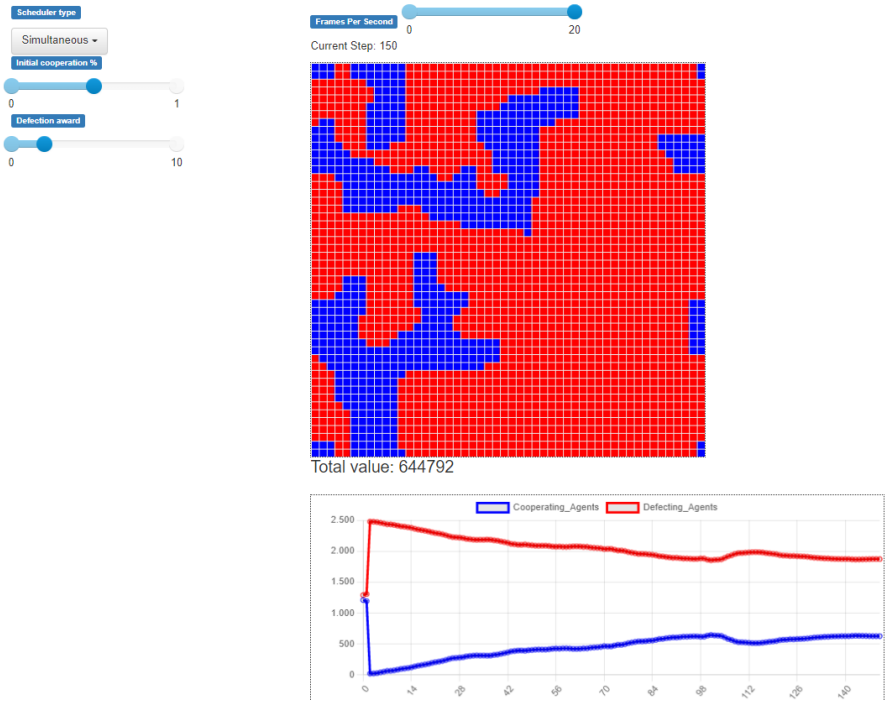

Figura 1.4: Experimento com "prêmio de deserção" em 2.0

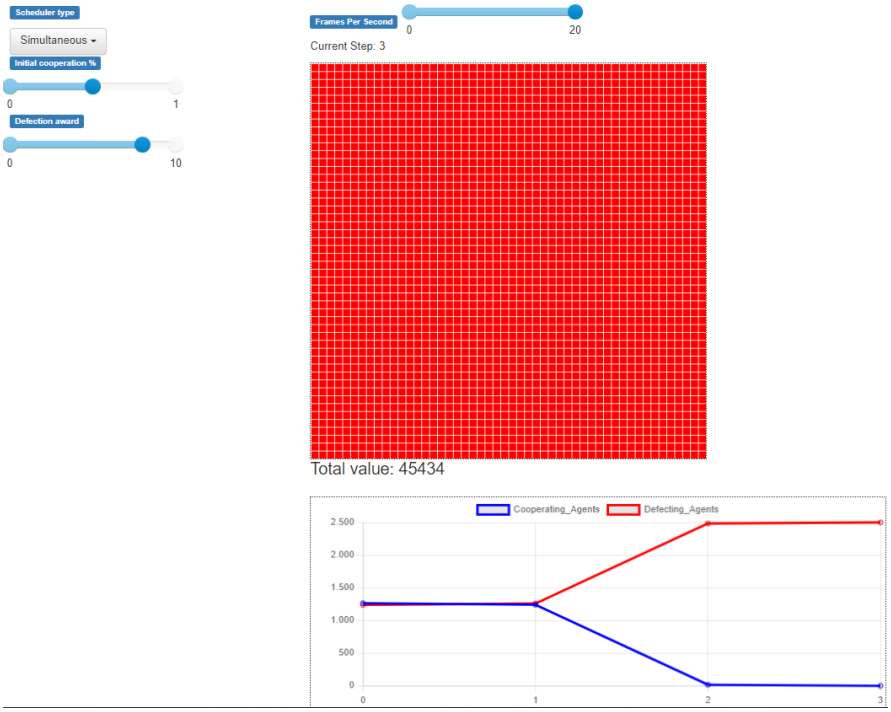

Figura 1.5: Experimento com "prêmio de deserção" em 8.0

cooperativo não se extingua e seja possível manter a geração de valor?

Simulando o sistema com 80% de agentes cooperantes iniciais, e "prêmio de deserção" em 2.0 observou-se que houve um maior equilíbrio entre a quantidade final de agentes cooperantes e desertores. Graças a isso, gerou-se mais valor total para o sistema, se compararmos as figuras 1.6 e 1.4

Um último experimento foi verificar como o sistema se comporta se houver poucos agentes cooperantes no começo, mas com um prêmio de deserção que possibilite a futura expansão dos cooperantes (ou seja, menor que 2.0). Observou-se que é pouco vantajoso, pois o pico inicial de agentes desertores se mantém por muito mais tempo do que com a configuração padrão (50% inicialmente), o que prejudica a geração de valor ao longo do tempo. Resultados podem ser vistos na figura 1.7.

1.5 Discussão e *insights* preliminares sobre as hipóteses

Com os experimentos realizados, é possível verificar que aparentemente os agentes que se mantém cooperantes por tempo suficiente no sistema nunca se tornam desertores. Isso quer dizer que para agentes cooperantes próximos uns dos outros, a cooperação leva a uma maior pontuação individual ao longo do tempo.

Já com relação ao o sistema como um todo, é possível concluir que sistemas que têm um

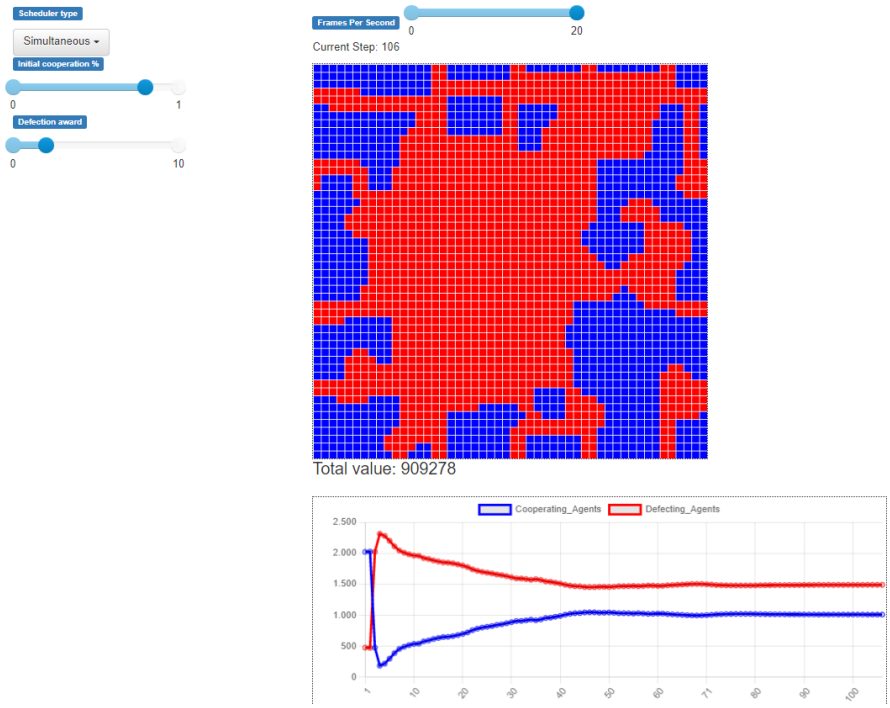

Figura 1.6: Experimento com "prêmio de deserção" em 2.0 e porcentagem de agentes cooperantes inicialmente em 80%

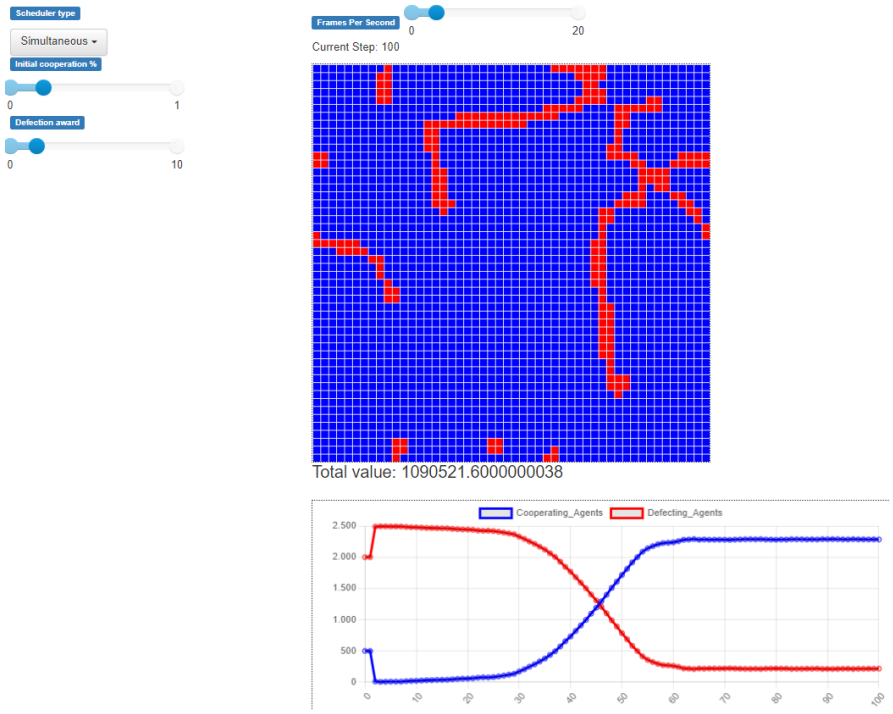

Figura 1.7: Experimento com "prêmio de deserção" em 1.6 e porcentagem de agentes cooperantes inicialmente em 20%

maior número de agentes cooperantes geram mais valor ao longo do tempo. Em contrapartida, sistemas que só têm agentes desertores param de gerar valor à partir de certo ponto, pois não existem mais agentes cooperantes para premiar relações *(D, C)*, e todas as interações entre agentes se tornam nulas. Essa conclusão concorda com a hipótese causal e mostra que além de aumentar o ganho para o sistema, os agentes cooperantes são estritamente necessários para a geração de valor como um todo.

Vale citar que esse modelo não representa muito bem como funcionaria a evolução e propagação da cooperação em uma sociedade do mundo real, já que o comportamento é algo "booleano" (ou coopera, ou não coopera). Idealmente, a cooperação deve ter um grau variável para termos algo próximo do mundo real, como é feito em (KILLINGBACK; DOEBELI, 2002).

A hipótese causal sugerida aparenta ter sido comprovada à partir dos experimentos realizados, e considerando as limitações e a proposta do modelo.

1.6 Conclusão

Através da realização desse exercício, foi possível compreender como são feitas simulações multi-agente, especialmente utilizando o framework Python/MESA, que provê ferramentas simples mas poderosas para a realização desse tipo de experimento.

Como o experimento se tratava de uma adaptação do modelo existente, houve pouco relacionamento com os temas mais relevantes encontrados na pesquisa bibliométrica realizada. Seria interessante estudar diferentes estratégias tomadas pelos agentes nas interações do dilema do prisioneiro, mas o modelo existente dificultava esse tipo de adaptação e seria necessária uma grande reformulação no código existente e por isso essa possibilidade foi descartada.

Ainda assim, foi possível elaborar uma hipótese causal inicial para o fenômeno do dilema do prisioneiro iterativo e investigar essa hipótese à partir dos experimentos realizados no laboratório de simulações desenvolvido.

Bibliografia

EPSTEIN, Joshua M. Zones of cooperation in demographic prisoner's dilemma. *Complexity*, v. 4, n. 2, p. 36–48, nov. 1998. Publisher: John Wiley & Sons, Ltd. ISSN 1076-2787. DOI: 10.1002/(SICI)1099-0526(199811/12)4:2<36::AID-CPLX9>3.0.CO;2-Z. Disponível em: <https://doi.org/10.1002/(SICI)1099-0526(199811/12)4:2%3C36::AID-CPLX9%3E3.0.CO;2-Z>. Acesso em: 27 dez. 2022. Citado nas pp. 5, 6.

KILLINGBACK, Timothy; DOEBELI, Michael. The Continuous Prisoner's Dilemma and the Evolution of Cooperation through Reciprocal Altruism with Variable Investment. *The American Naturalist*, v. 160, n. 4, p. 421–438, out. 2002. Publisher: The University of Chicago Press. ISSN 0003-0147. DOI: 10.1086/342070. Disponível em: <https://doi.org/10.1086/342070>. Acesso em: 2 jan. 2023. Citado na p. 14.

PRISONER'S Dilemma. Disponível em: <https://en.wikipedia.org/wiki/Prisoner%27s_dilemma>. Acesso em: 3 dez. 2022. Citado na p. 3.

WILENSKY, U. *NetLogo PD Basic Evolutionary model*. English. Center for Connected Learning e Computer-Based Modeling, Northwestern University, Evanston, IL., 2002. Disponível em: <http://ccl.northwestern.edu/netlogo/models/PDBasicEvolutionary>. Acesso em: 27 dez. 2022. Citado nas pp. 5, 6.