Propagação de doenças: um estudo através de autômatos celulares

Grupo 8

Requerimentos

In [119…

```
import numpy as np
import random
#import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap, Normalize
```

Funções Auxiliares

In [120…

```
# Criar matriz de spins e vizinhos (one-based)
def gen_vizinhanca(L):

  s = np.zeros(L*L + 1)
  viz = np.zeros((L*L+1,8))
  for i in range(1,s.shape[0]):
    viz[i][0] = i + 1
    if(i%L == 0):
      viz[i][0] = i + 1 - L

    viz[i][1] = i + L
    if(i > (L*L - L)):
      viz[i][1] = i + L - (L*L)

    viz[i][2] = i - 1
    if((i - 1)%L == 0):
      viz[i][2] = i + L - 1

    viz[i][3] = i - L
    if(i < (L + 1)):
      viz[i][3] = i + (L*L) - L

    viz[i][4] = i - L + 1
    if(i%L == 0 and i < (L + 1)):
      temp = i + (L*L) - L
      viz[i][4] = temp + 1 - L
    elif(i%L == 0):
      viz[i][4] = i + 1 - 2*L
    elif(i < (L + 1)):
      viz[i][4] = i + (L*L) - L + 1

    viz[i][5] = i + L + 1
    if(i%L == 0 and i > (L*L - L)):
      temp = i + L - (L*L)
      viz[i][5] = temp + 1 - L
    elif(i%L == 0):
      viz[i][5] = i + 1
    elif(i > (L*L - L)):
      viz[i][5] = i + L - (L*L) + 1

    viz[i][6] = i + L - 1
    if(i > (L*L - L) and (i - 1)%L == 0):
      temp = i + L - (L*L)
      viz[i][6] = temp + L - 1
    elif(i > (L*L - L)):
      viz[i][6] = i + L - (L*L) - 1
```

```
    elif((i - 1)%L == 0):
      viz[i][6] = i + 2*L - 1

    viz[i][7] = i - L - 1
    if(i < (L + 1) and (i - 1)%L == 0):
      temp = i + (L*L) - L
      viz[i][7] = temp + L - 1
    elif(i < (L + 1)):
      viz[i][7] = i + (L*L) - L - 1
    elif((i - 1)%L == 0):
      viz[i][7] = i - 1

  s = s.astype(int)
  viz = viz.astype(int)
  return s, viz
```

In [121…

```
def create_simulation(iter, L, p_c, p_r):
  # 0 -> suscetível (amarelo)
  # 1 -> infectado (vermelho)
  # 2 -> curado (verde)
  norm = Normalize(vmin=0, vmax=2)
  cmap = ListedColormap(["darkgoldenrod", "maroon", "forestgreen"])
  m = plt.cm.ScalarMappable(norm=norm, cmap=cmap)

  s, viz = gen_vizinhanca(L)
  s[random.randint(1,L*L)] = 1
  visualizacao = np.array(s)

  fig, axs = plt.subplots(int(iter/3), 3)
  fig.suptitle("SIR Simulation - Contamination/Recovery Rate = " + str(p_c) + "/" +
  axs[0, 0].imshow(visualizacao[1:].reshape(L,L), cmap=cmap, norm=norm)
  axs[0, 0].axis('off')
  axs[0 ,0].set_title("Step 0", fontsize = 20)

  sus_list = list()
  inf_list = list()
  rec_list = list()

  for k in range(1, iter):
    new_s = np.zeros(L*L + 1)
    new_s = new_s.astype(int)

    for i in range(1,s.shape[0]):
      if s[i] == 1:
        if random.uniform(0,1) < p_r:
          new_s[i] = 2
        else:
          new_s[i] = 1
      elif s[i] == 0:
        for j in range(viz.shape[1]):
          if s[viz[i][j]] == 1:
            if random.uniform(0,1) < p_c:
              new_s[i] = 1
      else:
        new_s[i] = s[i]
    s = new_s

    visualizacao = np.array(s)
    axs[int(k/3), (k%3)].imshow(visualizacao[1:].reshape(L,L),cmap=cmap, norm=norm)
    axs[int(k/3), (k%3)].axis('off')
    axs[int(k/3), (k%3)].set_title("Step "+str(k), fontsize = 20)

    sus = 0
    inf = 0
    rec = 0
    for i in range(1,s.shape[0]):
      if s[i] == 0:
        sus = sus + 1
      elif s[i] == 1:
```

```
            inf = inf + 1
        elif s[i] == 2:
            rec = rec + 1

    sus_list.append(sus/(L*L))
    inf_list.append(inf/(L*L))
    rec_list.append(rec/(L*L))

  plt.axis('off')
  fig.set_figheight(6*int((k+2)/3))
  fig.set_figwidth(18)
  plt.show()
  plt.close()

  plt.figure(figsize=(12,8))
  plt.plot(range(iter-1),sus_list,color='darkgoldenrod',label='Suscetíveis')
  plt.plot(range(iter-1),inf_list,color='maroon',label='Infectados')
  plt.plot(range(iter-1),rec_list,color='forestgreen',label='Recuperados')
  plt.legend()
  plt.show()
```

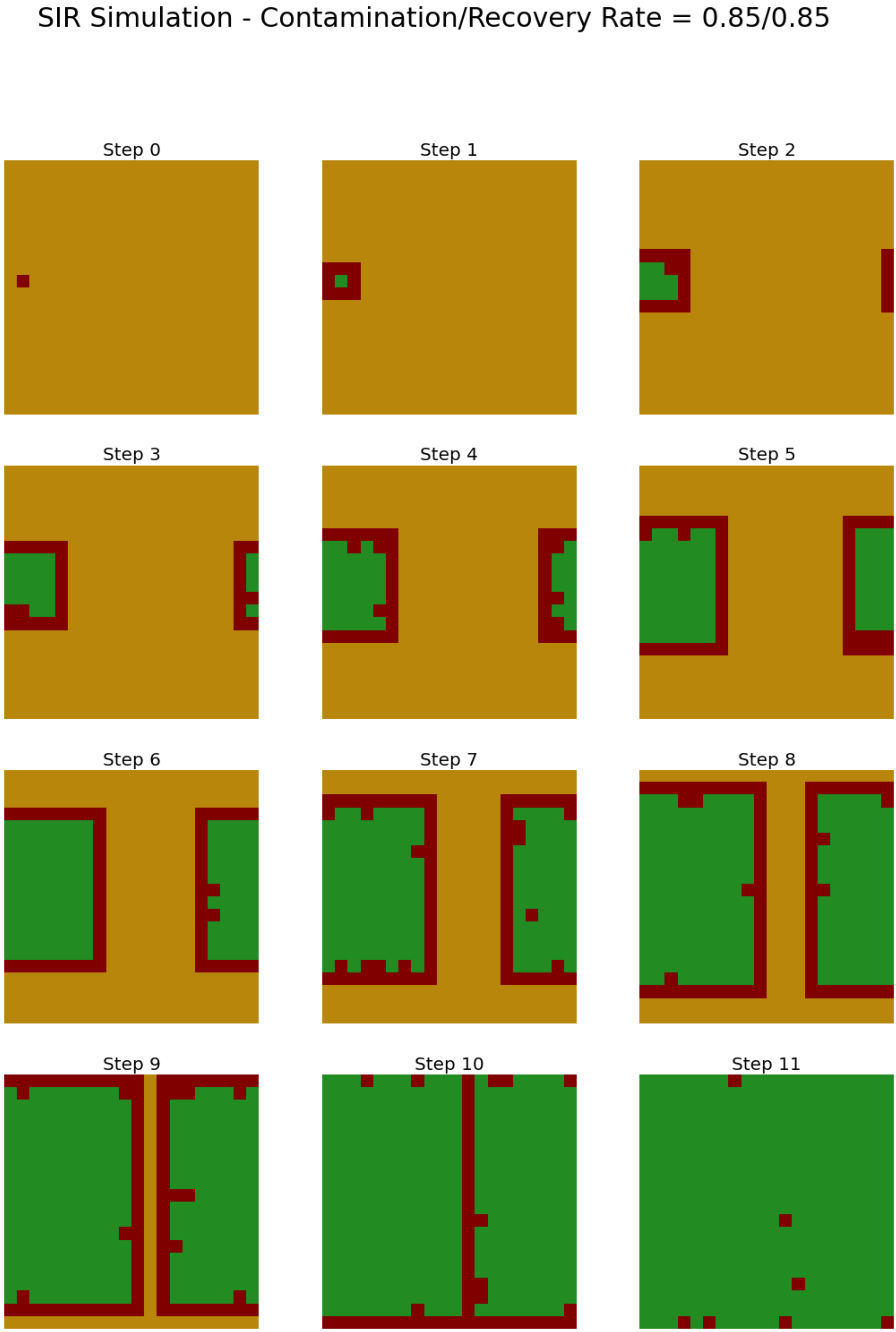
Casos

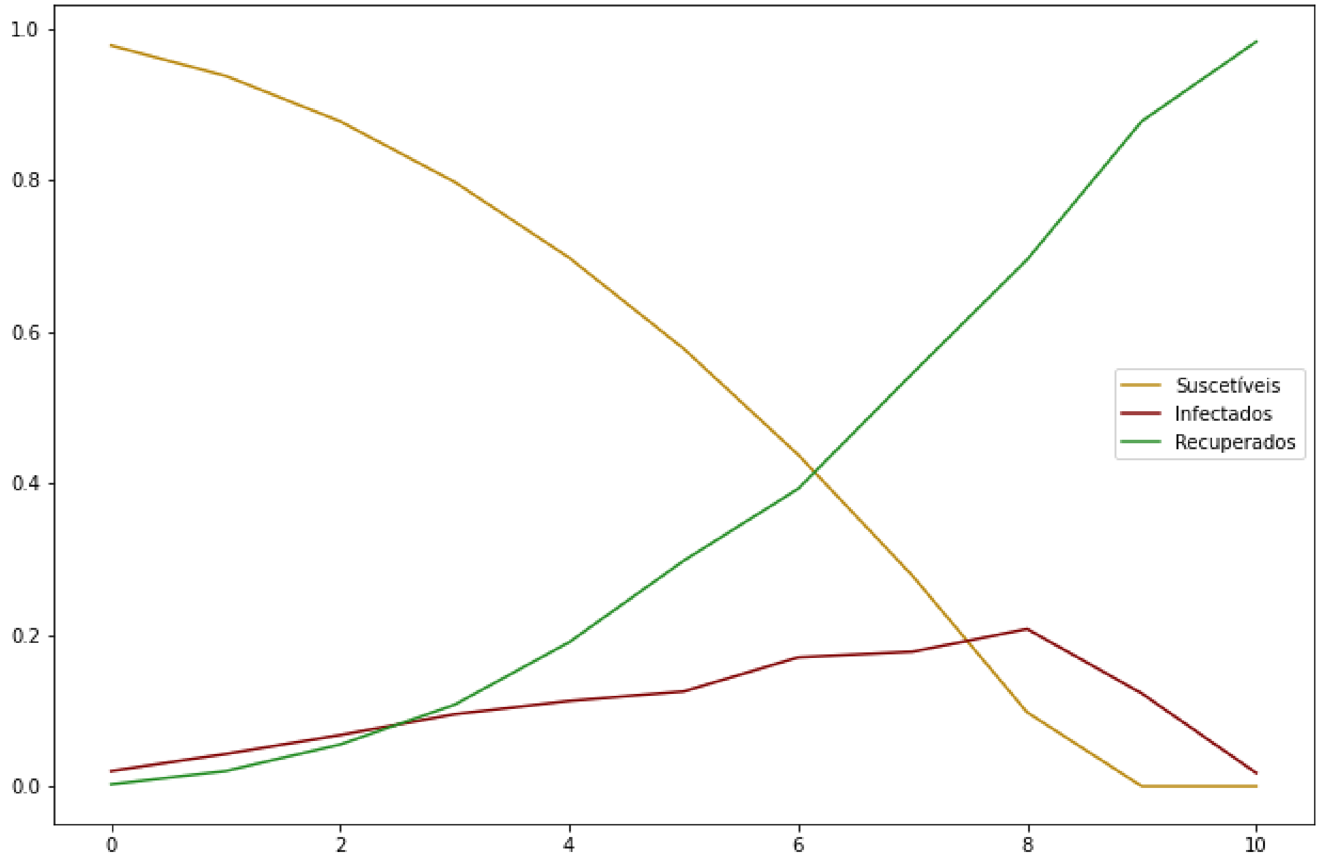
Estudaremos casos variando o tamanho L, a probabilidade p_r de recuperação, a probabilidade p_c de contaminação e o número de iterações iter.

Caso 1: Alta taxa de recuperação e alta taxa de contaminação

In [122…

```
L = 20
iter = 12
p_c = 0.85
p_r = 0.85
create_simulation(iter, L, p_c, p_r)
```


SIR Simulation - Contamination/Recovery Rate = 0.85/0.85
Step 0
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7
Step 8
Step 9
Step 10
Step 11

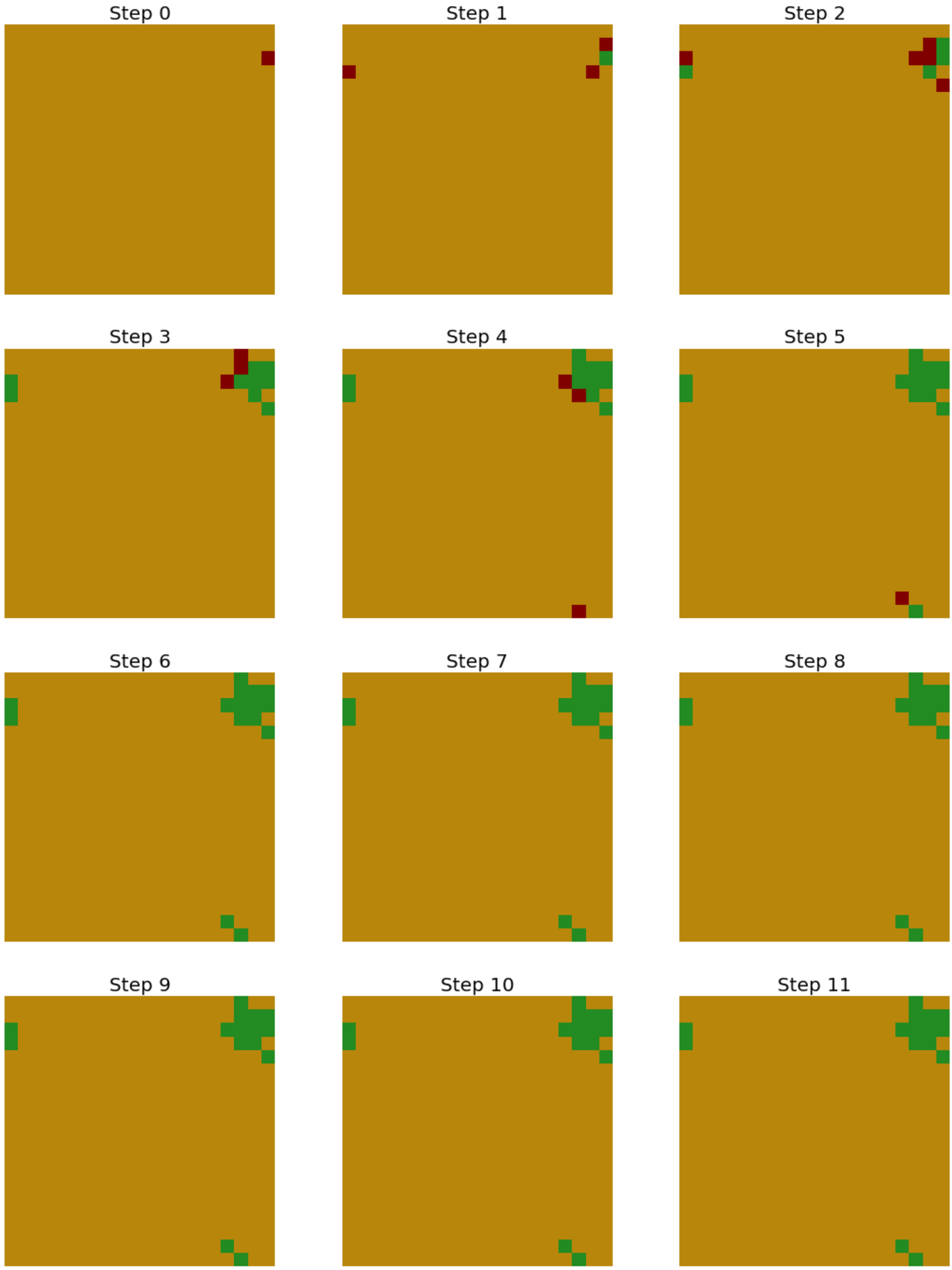


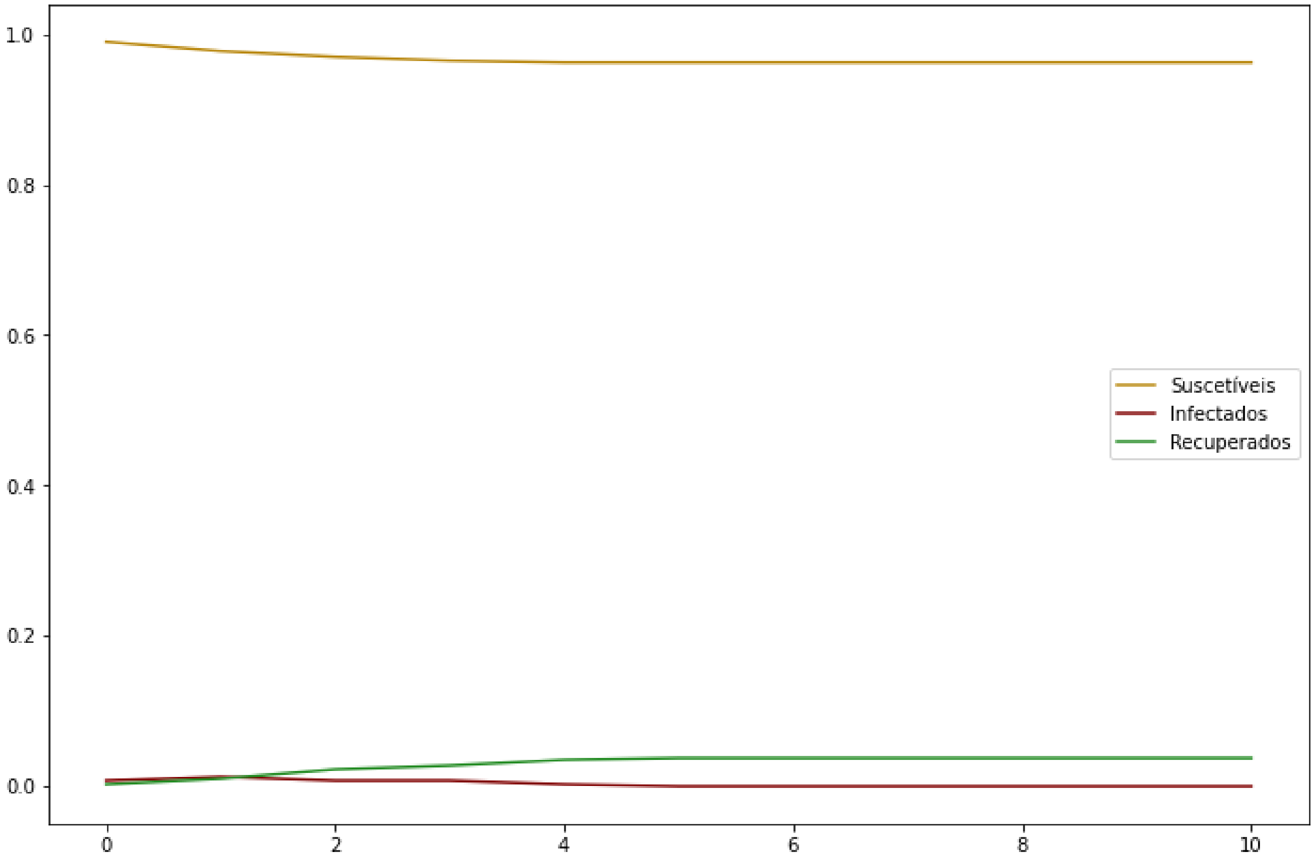
Caso 2: Alta taxa de recuperação e baixa taxa de contaminação

In [123…

```
L = 20
iter = 12
p_c = 0.15
p_r = 0.85
create_simulation(iter, L, p_c, p_r)
```

SIR Simulation - Contamination/Recovery Rate = 0.15/0.85



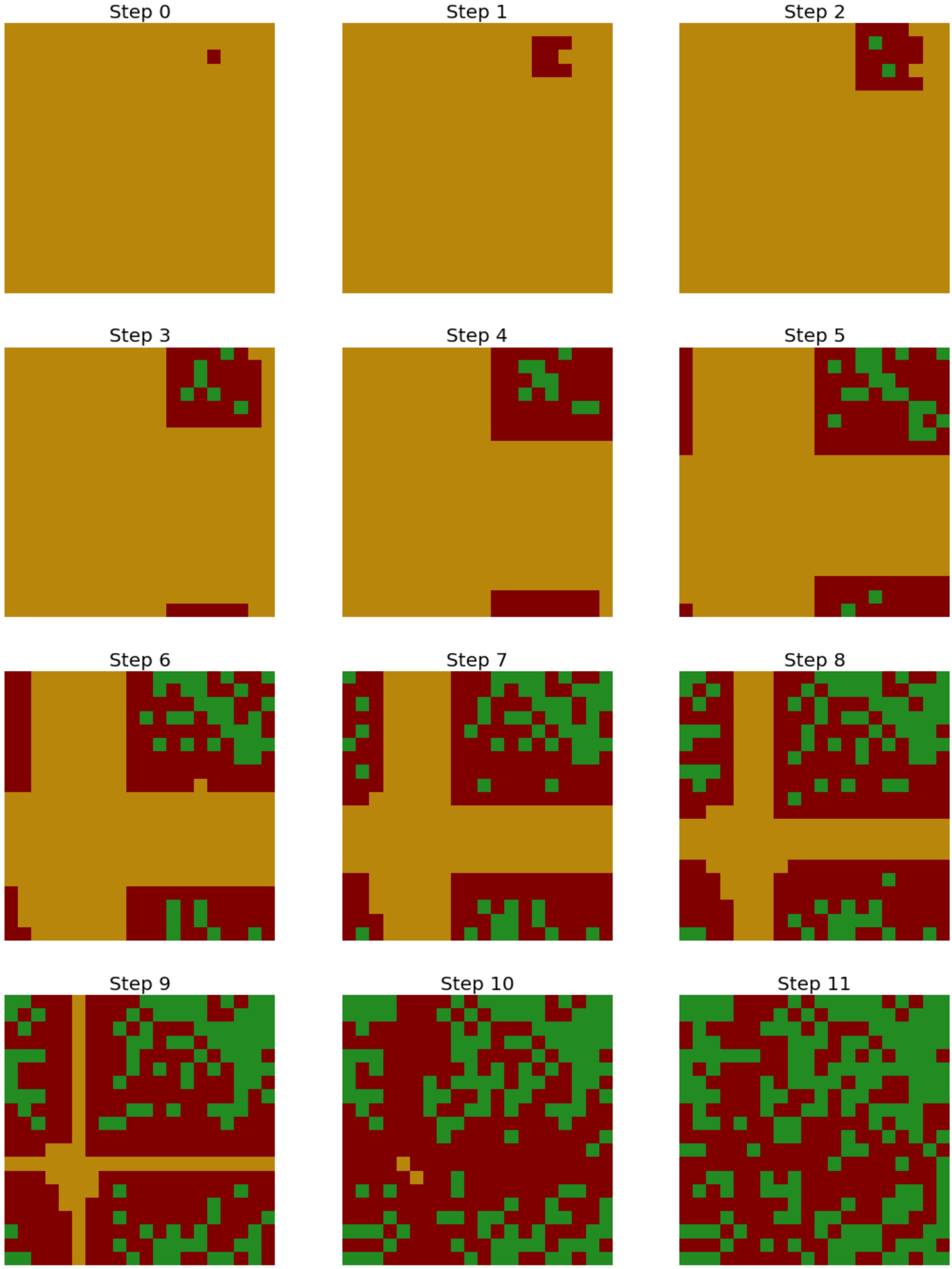


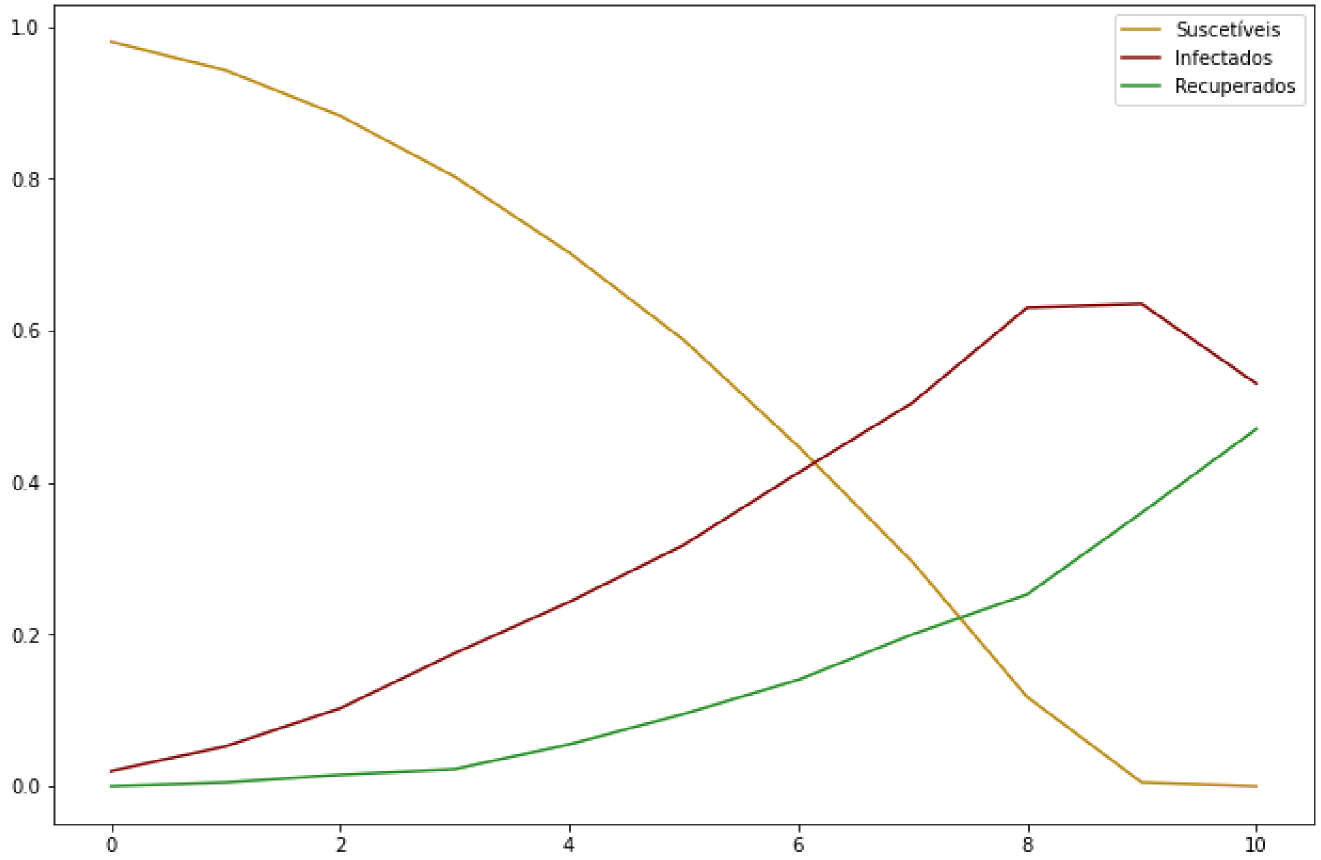
Caso 3: Baixa taxa de recuperação e alta taxa de contaminação

In [124…

```
L = 20
iter = 12
p_c = 0.85
p_r = 0.15
create_simulation(iter, L, p_c, p_r)
```

SIR Simulation - Contamination/Recovery Rate = 0.85/0.15



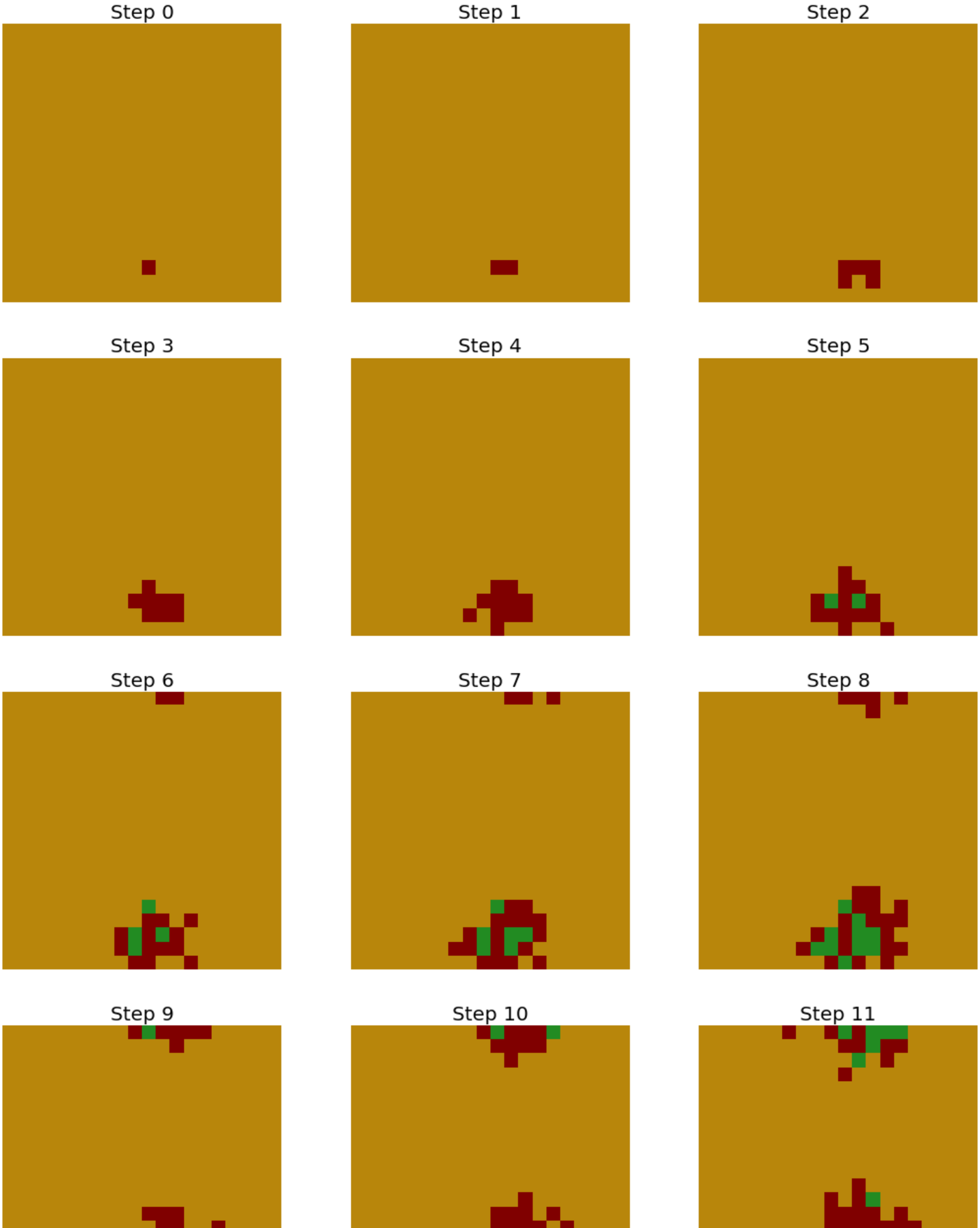


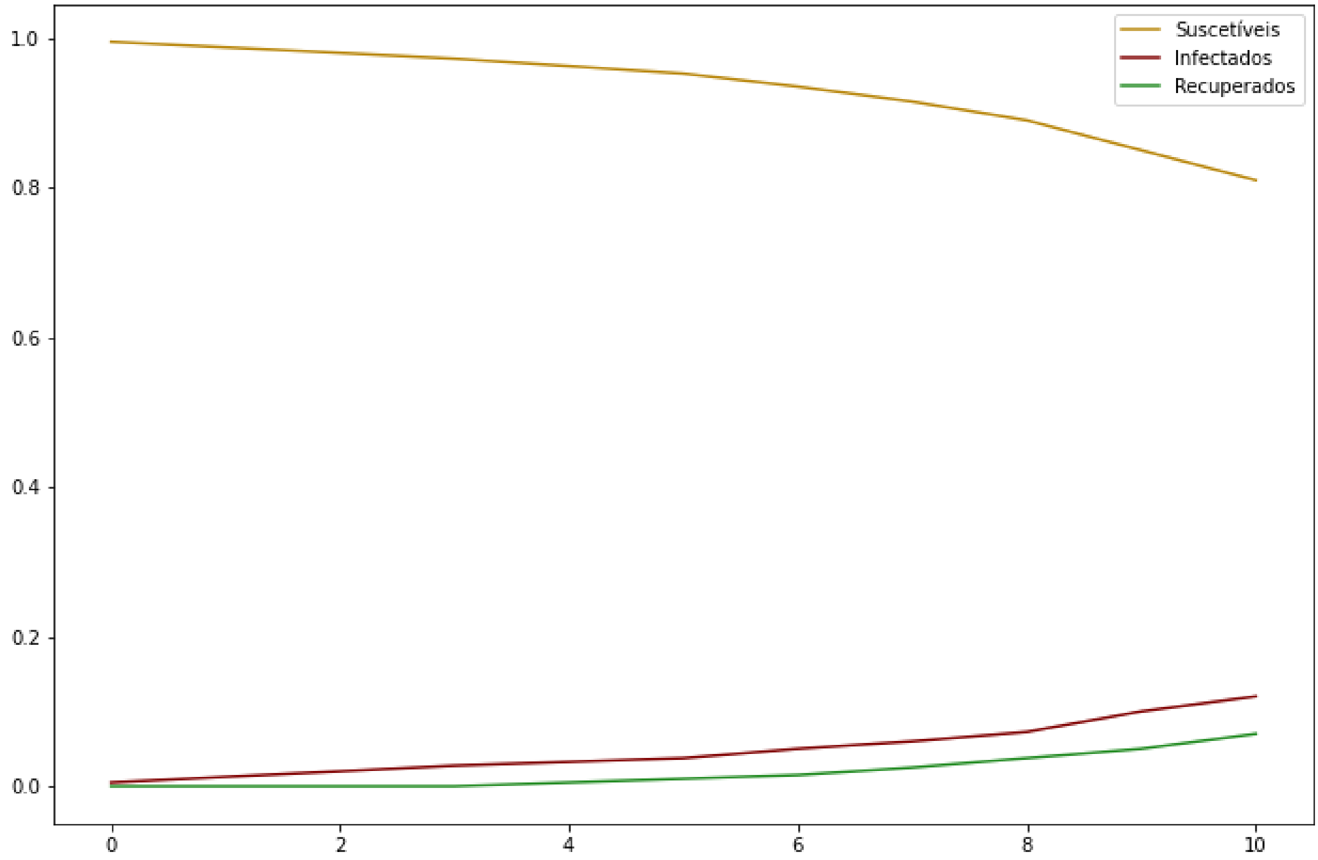
Caso 4: Baixa taxa de recuperação e baixa taxa de contaminação

In [125…

```
L = 20
iter = 12
p_c = 0.15
p_r = 0.15
create_simulation(iter, L, p_c, p_r)
```

SIR Simulation - Contamination/Recovery Rate = 0.15/0.15





Suscetíveis
Infectados
Recuperados
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10