Zachodniopomorski Uniwersytet Technologiczny w Szczecinie

Wydział Informatyki

LAB 11 - ZAGROŻENIA I ODPORNOŚĆ SIECI

Karol Działowski

nr albumu: 39259

przedmiot: Modelowanie zachowań w sieciach złożonych

Szczecin, 27 grudnia 2020

Spis treści

1 Cel laboratorium 1

2 Wyniki 2

3 Podsumowanie 3

Bibliografia 3

A Implementacja 4

1 Cel laboratorium

Celem laboratorium było przeprowadzenie symulacji na wcześniej zaimplementowanym modelu SIR (*Susceptible, Infectious, or Recovered*)) [1] badając wpływ losowego usunięcia węzłów na zasięg procesu propagacji.

Zaimplementowany model definiują parametry:

początkowa liczba węzłów zarażonych n – określa liczbę węzłów zarażonych w pierwszej iteracji

prawdopodobieństwo wyzdrowienia m – określa z jakim prawdopodobieństwem węzeł przechodzi ze stanu zarażony **I – infected** do stanu ozdrowienia **R – recovered**.

prawdopodobieństwo zarażanie b – określa z jakim prawdopodobieństwem węzeł zaraża sąsiada.

procent usuniętych węzłów l – określa ile węzłów będzie niedostępnych do roznoszenia choroby i zarażania się.

2 Wyniki

W zadaniu należało przygotować sieć WS [2] z około 2000 węzłami oraz sieć BA [3] o takiej samej liczbie węzłów z porównywalnym średnim stopniem wierzchołków.

Stworzono sieci o następujących parametrach:

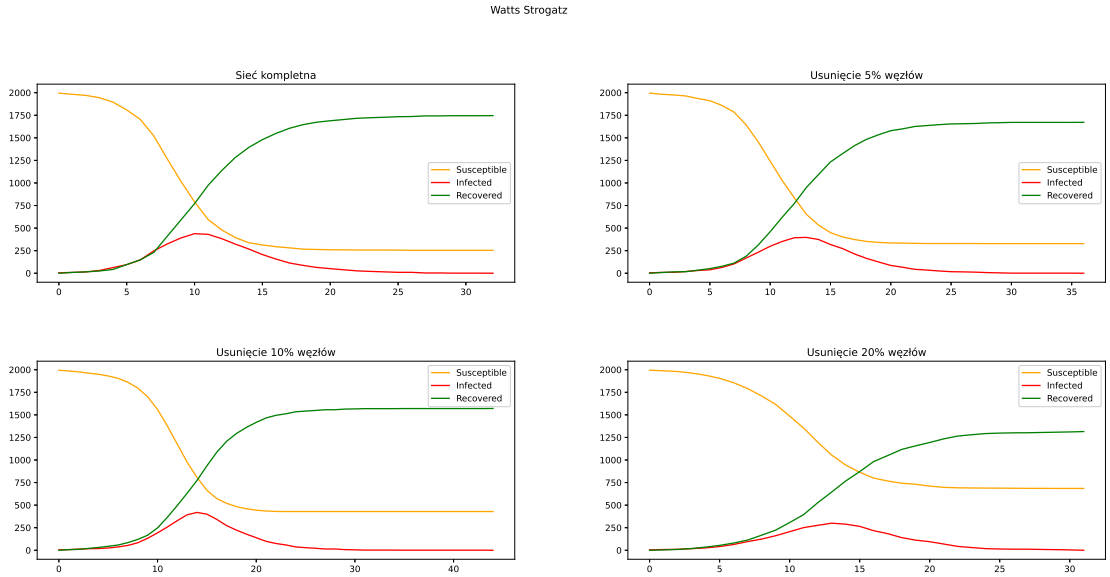
Kod źródłowy 1: Parametry badanych sieci
Źródło: Opracowanie własne

```
1       g = Graph.Watts_Strogatz(dim=1, size=2000, nei=3, p=0.8)
2       g = Graph.Barabasi(n=2000, m=3)
```

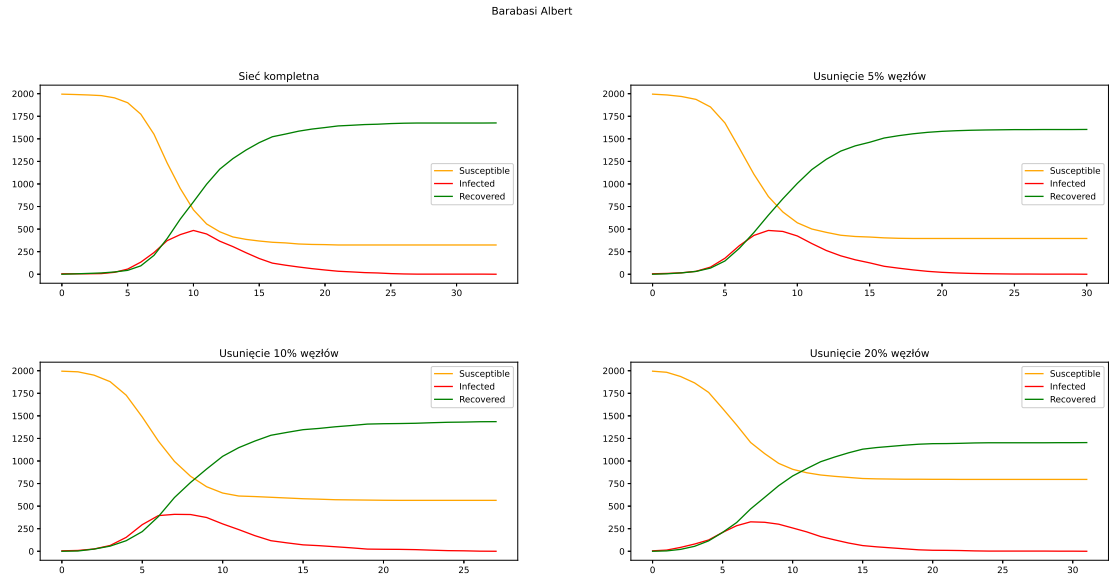
Uzyskując średni stopień wierzchołków $d = 6.0$ dla modelu WS i $d = 5.994$ dla modelu BA.

Następnie należało zaimplementować w dowolnym języku programowania model SIR wzbogacony o usunięcie węzłów z wykorzystaniem biblioteki *iGraph* [4].

Przeprowadzono eksperymenty dla czterech parametrów - sieci kompletnej, sieci z usuniętymi 5% węzłów, 10% węzłów i 20% węzłów dla obu modeli – WS i BA. Wyniki przedstawiono na rysunkach 1 i 2.



Rysunek 1: Model WS



Rysunek 2: Model BA

3 Podsumowanie

Eksperyment jest podobny do wcześniej przeprowadzonego eksperymentu nad modelem adaptacyjnym. Modele te różnią się tym, że opisywany w tym sprawozdaniu dezaktywuje węzły pernamentnie.

Dla obu badanych sieci, WS i BA, zwiększanie usuniętej liczby węzłów ma istotny wpływ na przebieg propagacji procesu. Usuwając 20% węzłów z sieci WS występuje spadek maksymalnej liczby zarażonych z około 500 do 250, analogicznie w przypadku sieci BA.

Usunięcie węzłów ma też wpływ na przesunięcie szczytu zakażeń, dla sieci kompletnej WS szczyt zakażeń wystąpił w okolicy 10 iteracji, dla usuniętych 5% w okolicach 13 iteracji, dla usuniętych 10% węzłów w okolicach 13 iteracji a dla 20% węzłów w okolicach 14 iteracji.

Czegoś takiego nie zaobserwowano w modelu BA, gdzie usunięcie węzłów przyspieszało szczyt zakażeń i przesuwało, go w lewą stronę wykresu. Wynika to z różnicy w jaki sposób modele te są zbudowane. Model WS ma modelować małe społeczności, gdzie średnia ścieżka pomiędzy węzłami wynosi pewną określoną wartość będącą parametrem modelu, a model BA charakteryzuje się tym, że prawdopodobieństwo przyłączenia się do węzła wzrasta z jego stopniem.

Zaimplementowany model może służyć szacowaniu wpływu kwarantanny i izolacji społecznej na rozprzestrzenianie się choroby.

Bibliografia

[1] Kermack W. O., McKendrick A. G.: Contributions to the mathematical theory of epidemics—i. *Bulletin of mathematical biology*, vol. 53, no. 1-2, pp. 33–55, 1991.

[2] Watts D. J., Strogatz S. H.: Collective dynamics of ‘small-world’ networks. *Nature*, vol. 393, no. 6684, pp. 440–442, czer. 1998, DOI: 10.1038/30918.

[3] Albert R., Barabási A.-L.: Statistical mechanics of complex networks. *Rev. Mod. Phys.*, vol. 74, pp. 47–97, sty. 2002, DOI: 10.1103/RevModPhys.74.47.

[4] Csardi G., Nepusz T.: The igraph software package for complex network research. *InterJournal*, vol. Complex Systems, s. 1695, 2006, URL: https://igraph.org.

A Implementacja

Kod źródłowy 2: Implementacja modelu
Źródło: Opracowanie własne

```
"""
Wykorzastanie sieci WS i BA złożonych z około 500-1000 węzłów z porównywalnym średnim degree.

Zbadać wypływu usunięcia losowego 5%, 10%, 20% węzłów na zasięg procesu propagacji.

Usunięcie może być reprezentowane przez wprowadzenie dodatkowego attrybutu dla węzła.

Propagacja symulowana na podstawie dowolnego z wcześniejszych modeli z ustalonymi parametrami propagacji, taką samą liczbą seedów.
Przykładowo transmission rate 0.1 i seeding 5% dla wszystkich procesów.

Efektem są dwa wykresy z przebiegami procesów na każdym efekty dla sieci kompletnej i po usunięciu 5%, 10%, 20% węzłów.
"""

from igraph import *
import random
import numpy as np
import matplotlib.pyplot as plt


def simulation(g, n, m, b, limitation):
    """
    SIR Model simulation
    :param g: graph
    :param n: size of initial infection
    :param m: probability of recovery
    :param b: probability of infection
    :param limitation: how many nodes are unavaiable (as a fraction)
    :return: list of graphs in each step, array of S, I, R counts
    """
    g.vs["state"] = "S"
    g.vs["color"] = "orange"
    g.vs["size"] = 50
    g.vs["availability"] = True

    seeding(g, n=n)
    set_availability(g, limitation)
```

```
37
38      update_colors(g)
39
40      g_history = [deepcopy(g)]
41      counts = [count_states(g)]
42      flag = True
43      while flag:
44          infection(g, b=b)
45          recovery(g, m=m)
46          update_colors(g)
47          s_count, i_count, r_count = count_states(g)
48          counts.append((s_count, i_count, r_count))
49          flag = i_count != 0
50          g_history.append(deepcopy(g))
51
52      return g_history, np.array(counts)
53
54
55  def seeding(g, n):
56      indexes = g.vs.indices
57      indexes_to_infect = random.sample(indexes, n)
58      g.vs[indexes_to_infect]["state"] = "I"
59
60
61  def set_availability(g, limitation):
62      indexes = g.vs.indices
63      n = int(len(indexes) * limitation)
64      indexes_to_disable = random.sample(indexes, n)
65      g.vs[indexes_to_disable]["availability"] = False
66
67
68  def update_colors(g):
69      infected_indexes = list(np.where(np.array(g.vs["state"]) == "I")[0])
70      g.vs[infected_indexes]["color"] = "red"
71      recovered_indexes = list(np.where(np.array(g.vs["state"]) == "R")[0])
72      g.vs[recovered_indexes]["color"] = "green"
73      recovered_indexes = list(np.where(np.array(g.vs["availability"]) == False)[0])
74      g.vs[recovered_indexes]["color"] = "blue"
75
76
77  def infection(g, b):
78      infected_indexes = list(np.where(np.array(g.vs["state"]) == "I")[0])
79      for i in infected_indexes:
80          if g.vs["availability"]:
81              neighbors = g.neighbors(g.vs[i])
82              for neighbor in neighbors:
83                  if g.vs[neighbor]["state"] == "S" and g.vs[neighbor]["availability"]:
84                      if np.random.rand() < b:
85                          g.vs[neighbor]["state"] = "I"
86
87
```

```
def recovery(g, m):
    infected_indexes = list(np.where(np.array(g.vs["state"]) == "I")[0])
    for i in infected_indexes:
        if np.random.rand() < m:
            g.vs[i]["state"] = "R"


def count_states(g):
    infected_indexes = list(np.where(np.array(g.vs["state"]) == "I")[0])
    recovered_indexes = list(np.where(np.array(g.vs["state"]) == "R")[0])
    susceptible_indexes = list(np.where(np.array(g.vs["state"]) == "S")[0])
    return len(susceptible_indexes), len(infected_indexes), len(recovered_indexes)


def ws():
    g = Graph.Watts_Strogatz(dim=1, size=2000, nei=3, p=0.8)
    _, counts_orig = simulation(g, n=5, m=0.3, b=0.3, limitation=0)
    _, counts_5 = simulation(g, n=5, m=0.3, b=0.3, limitation=0.05)
    _, counts_10 = simulation(g, n=5, m=0.3, b=0.3, limitation=0.10)
    _, counts_20 = simulation(g, n=5, m=0.3, b=0.3, limitation=0.20)
    plot_comparison(counts_orig, counts_5, counts_10, counts_20, title="Watts
Strogatz")


def plot_comparison(counts_orig, counts_5, counts_10, counts_20, title=""):
    fig, axes = plt.subplots(2, 2)
    fig.suptitle(title)

    axes[0, 0].plot(counts_orig[:, 0], label="Susceptible", c="orange")
    axes[0, 0].plot(counts_orig[:, 1], label="Infected", c="red")
    axes[0, 0].plot(counts_orig[:, 2], label="Recovered", c="green")
    axes[0, 0].set_title("Sieć kompletna")
    axes[0, 0].legend()

    axes[0, 1].plot(counts_5[:, 0], label="Susceptible", c="orange")
    axes[0, 1].plot(counts_5[:, 1], label="Infected", c="red")
    axes[0, 1].plot(counts_5[:, 2], label="Recovered", c="green")
    axes[0, 1].set_title("Usunięcie 5% węzłów")
    axes[0, 1].legend()

    axes[1, 0].plot(counts_10[:, 0], label="Susceptible", c="orange")
    axes[1, 0].plot(counts_10[:, 1], label="Infected", c="red")
    axes[1, 0].plot(counts_10[:, 2], label="Recovered", c="green")
    axes[1, 0].set_title("Usunięcie 10% węzłów")
    axes[1, 0].legend()

    axes[1, 1].plot(counts_20[:, 0], label="Susceptible", c="orange")
    axes[1, 1].plot(counts_20[:, 1], label="Infected", c="red")
    axes[1, 1].plot(counts_20[:, 2], label="Recovered", c="green")
    axes[1, 1].set_title("Usunięcie 20% węzłów")
    axes[1, 1].legend()
```

```
138
139     plt.tight_layout()
140     plt.show()
141
142
143 def ba():
144     g = Graph.Barabasi(n=2000, m=3)
145     _, counts_orig = simulation(g, n=5, m=0.3, b=0.3, limitation=0)
146     _, counts_5 = simulation(g, n=5, m=0.3, b=0.3, limitation=0.05)
147     _, counts_10 = simulation(g, n=5, m=0.3, b=0.3, limitation=0.10)
148     _, counts_20 = simulation(g, n=5, m=0.3, b=0.3, limitation=0.20)
149
150     plot_comparison(
151         counts_orig, counts_5, counts_10, counts_20, title="Barabasi Albert"
152     )
153
154
155 if __name__ == "__main__":
156     ws()
157     ba()
158
159     # Sprawdzenie czy, mają podobny stopień wierzchołka
160     g_ba = Graph.Barabasi(n=2000, m=3)
161     g_ws = Graph.Watts_Strogatz(dim=1, size=2000, nei=3, p=0.8)
162     print("BA:", np.mean(g_ba.degree()), len(g_ba.vs.indices))
163     print("WS:", np.mean(g_ws.degree()), len(g_ws.vs.indices))
```