Analýza Sítí: Projekt

Richard Ficek

Úvod

V této práci jsem se zaměřil na analýzu dvou různých typů sítí. Cílem bylo provést analýzu vlastností obou sítí, interpretovat výsledky a identifikovat zajímavé vzorce v síti.

Výběr statické sítě

Pro tuto analýzu jsem vybral statickou síť **US airports**, která představuje síť letů mezi americkými letišti v roce 2010. Každá hrana představuje spojení mezi dvěma letišti, přičemž váha hrany udává počet letů mezi těmito dvěma letišti v daném směru v roce 2010.

Výběr Dynamická síťě

Pro tuto analýzu jsem vybral dynamickou síť **contacts-prox-high-school-2013**, která představuje kontaktní síť ve střední škole z roku 2013. Každý uzel představuje jednoho studenta a hrany mezi nimi zobrazují jejich fyzickou blízkost a kontakty během školního roku.

Preprocessing a Analýza - Statická síť

Statickou síť jsem analyzoval pomocí vlastního kódu v Pythonu a knihovny NetworkX. Vypočítal jsem základní metriky, které popisují strukturu a vlastnosti této sítě.

```
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt

# Load the space-separated file into a numpy array
file_path = 'out.opsahl-usairport'
data = np.loadtxt(file_path, skiprows=1)

# Create a graph from the numpy array
G = nx.Graph()
for edge in data:
    G.add_edge(int(edge[0]), int(edge[1]))

# Remove isolated nodes
G.remove_nodes_from(list(nx.isolates(G)))

# Compute network statistics
num_nodes = G.number_of_nodes()
num_edges = G.number_of_edges()
average_degree = sum(dict(G.degree()).values()) / num_nodes
density = nx.density(G)
is_connected = nx.is_connected(G)
diameter = nx.diameter(G) if is_connected else 'Graph is not connected'

# Print network statistics
print("Základní vlastnosti grafu:")
print(f"- Počet uzlů: {num_nodes}")
print(f"- Počet hran: {num_edges}")
print(f"- Průměrný stupeň uzlu: {average_degree:.2f}")
print(f"- Síť je souvislá: {'Ano' if is_connected else 'Ne'}")

if is_connected:
    print(f"- Průměrná délka cesty: {nx.average_shortest_path_length(G):.2f}")
    print(f"- Průměrný průměr grafu (diameter): {diameter}")
else:
    largest_cc = max(nx.connected_components(G), key=len)
    subgraph = G.subgraph(largest_cc)
    print(f"- Velikost největší souvislé komponenty (uzly): {len(largest_cc)}")
    print(f"- Počet uzlů v souvislé komponentě: {subgraph.number_of_nodes()}")
    print(f"- Počet hran v souvislé komponentě: {subgraph.number_of_edges()}")
    print(f"- Průměrná délka cesty v souvislé komponentě: {nx.average_shortest_path_length(subgraph):.2f}")
    print(f"- Průměrný průměr grafu (diameter) v souvislé komponentě: {nx.diameter(subgraph)}")

print("\nNejdůležitější uzly podle různých metrik:")
degree_centrality = nx.degree_centrality(G)
top_degree = sorted(degree_centrality.items(), key=lambda x: x[1], reverse=True)[:5]

print("1. Stupňová centrálnost:")
for node, value in top_degree:
    print(f"   - Uzel {node}: Centrálnost {value:.4f}")

print("\nDalší vlastnosti grafu:")
print(f"- Hustota grafu: {density:.4f}")

# Export the graph to a GraphML file for Gephi
output_file = 'graph.graphml'
nx.write_graphml(G, output_file)
print(f"\nGraph data has been exported to {output_file} for Gephi.")

# Find and visualize the largest clique, star, and edge-core
cliques = list(nx.find_cliques(G))
```

```
largest_clique = max(cliques, key=len)
print(f"\nLargest clique size: {len(largest_clique)}")

stars = [node for node, degree in G.degree() if degree == max(dict(G.degree()).values())]
print(f"Largest star size: {max(dict(G.degree()).values())}")

core_number = nx.core_number(G)
max_core = max(core_number.values())
print(f"Max core number: {max_core}")

# Visualize the largest clique
plt.figure(figsize=(12, 8))
pos = nx.spring_layout(G)
nx.draw_networkx_nodes(G, pos, nodelist=largest_clique, node_color='red', label='Largest Clique')
nx.draw_networkx_edges(G, pos, edgelist=[(u, v) for u, v in G.edges() if u in largest_clique and v in larges
plt.legend()
plt.title("Visualization of Largest Clique")
plt.show()

# Visualize the largest star
plt.figure(figsize=(12, 8))
nx.draw_networkx_nodes(G, pos, nodelist=stars, node_color='yellow', label='Largest Star')
nx.draw_networkx_edges(G, pos, edgelist=[(u, v) for u, v in G.edges() if u in stars or v in stars], edge_col
plt.legend()
plt.title("Visualization of Largest Star")
plt.show()

# Visualize the highest edge-core
nodes_in_max_core = [node for node, core in core_number.items() if core == max_core]
plt.figure(figsize=(12, 8))
nx.draw_networkx_nodes(G, pos, nodelist=nodes_in_max_core, node_color='green', label='Highest Edge-Core')
nx.draw_networkx_edges(G, pos, edgelist=[(u, v) for u, v in G.edges() if u in nodes_in_max_core and v in noc
plt.legend()
plt.title("Visualization of Highest Edge-Core")
plt.show()
```

Základní metriky pro statickou síť

Pro statickou síť jsem spočítal následující vlastnosti:

- Počet uzlů: 1574
- Počet hran: 17215
- Průměrný stupeň uzlu: 21.87
- Síť je souvislá: Ne
- Velikost největší souvislé komponenty (LCC): 1572
- Počet uzlů v souvislé komponentě: 1572
- Počet hran v souvislé komponentě: 17214
- Průměrná délka cesty v souvislé komponentě: 3.12
- Průměrný průměr grafu (diameter) v souvislé komponentě: 8

Nejdůležitější uzly podle různých metrik

Na základě výpočtů jsem identifikoval nejdůležitější uzly podle stupňové centrálnosti:

- Uzel 46: Centrálnost 0.1996
- Uzel 69: Centrálnost 0.1901
- Uzel 88: Centrálnost 0.1882
- Uzel 165: Centrálnost 0.1856
- Uzel 74: Centrálnost 0.1850

Další vlastnosti grafu

Dále jsem spočítal několik dalších vlastností grafu:

- Hustota grafu: 0.0139
- Giniho koeficient: 0.7534
- Průměrná délka cesty: 3.13685
- Clustering coefficient: 0.384

1 Největší letiště v USA

- **Mezinárodní letiště Hartsfield-Jackson Atlanta (ATL)** – Atlanta, Georgia.
- **Mezinárodní letiště Dallas/Fort Worth (DFW)** – Dallas/Fort Worth, Texas.
- **Mezinárodní letiště Denver (DEN)** – Denver, Colorado.
- **Mezinárodní letiště O'Hare (ORD)** – Chicago, Illinois.
- **Mezinárodní letiště Los Angeles (LAX)** – Los Angeles, Kalifornie.
- **Mezinárodní letiště Charlotte Douglas (CLT)** – Charlotte, Severní Karolína.
- **Mezinárodní letiště Orlando (MCO)** – Orlando, Florida.
- **Mezinárodní letiště Harryho Reida (LAS)** – Las Vegas, Nevada.
- **Mezinárodní letiště Phoenix Sky Harbor (PHX)** – Phoenix, Arizona.
- **Mezinárodní letiště Miami (MIA)** – Miami, Florida.

Vizualizace sítě

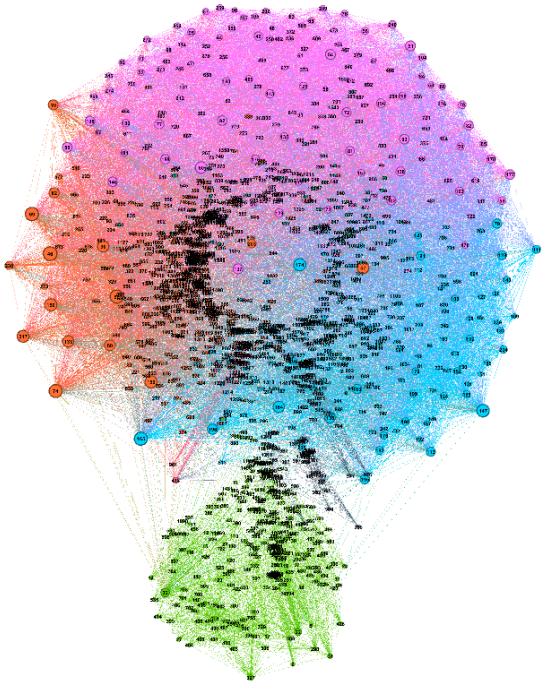


Figure 1: Distribuce stupňů v síti

Interpretace výsledků

Na základě výpočtů a vizualizací jsem identifikoval několik zajímavých vlastností statické sítě:

Statická síť

V případě statické sítě jsem pozoroval, že většina uzlů má nízký stupeň, ale existuje několik uzlů s vysokým stupněm, což ukazuje na existenci centrálních uzlů v síti. Tento jev je charakteristický pro mnoho reálných sítí, které vykazují vlastnosti tzv. malých světů (small-world networks).

Nejdůležitější uzly

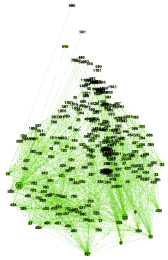
Uzly s vysokou stupňovou centrálností, jako jsou uzly 46, 69, 88, 165 a 74, hrají klíčovou roli v síťové struktuře a spojení mezi letišti. Tyto uzly jsou velmi propojené s ostatními uzly, což znamená, že jsou důležité pro zajištění plynulého provozu mezi letišti.

Hustota grafu

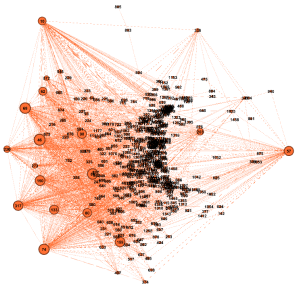
Hustota grafu je velmi nízká, což naznačuje, že mezi letišti existuje mnoho nepropojených cest, což může být důsledek geografických nebo provozních omezení. Hustota 0.0139 také ukazuje, že síť není příliš propojená, což může znamenat, že existují oblasti s nízkou frekvencí letů nebo omezeným spojením.

Clustering a assortativity

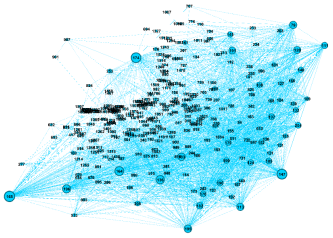
Clustering coefficient ukazuje střední úroveň propojení mezi sousedními uzly, což znamená, že existují skupiny uzlů (letišť), které jsou více propojené mezi sebou. Degree assortativity s hodnotou -0.113 naznačuje negativní korelaci mezi stupni propojení uzlů, což znamená, že uzly s vysokým stupněm mají tendenci se propojit s uzly s nízkým stupněm.



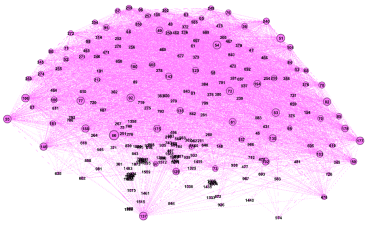
(a) Nejvýznamější ego, uzel číslo 32, možné Mezinárodní letiště Hartsfield-Jackson Atlanta



(b) Komponenta grafu, uzlem 46, 69, 74,88

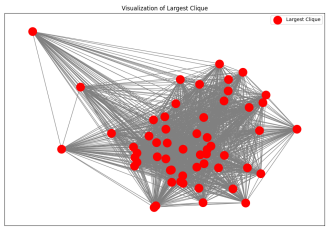


(c) Komponenta grafu, uzlem 165, 147

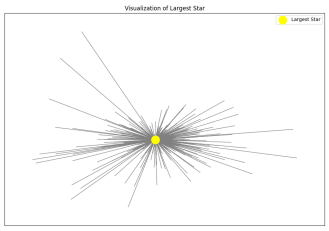


(d) Komponenta grafu, vyznamný uzel 60

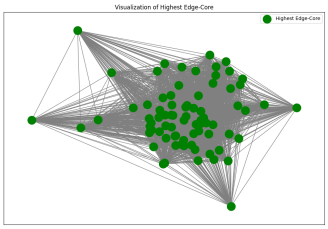
Figure 2: Vizualizace největších.



(a) Největší klika



(b) Největší hvězda



(c) Největší počet hran mezi uzly

Figure 3: Vizualizace největších.

Závěr

V této práci jsem analyzoval statickou síť letů mezi letišti v USA v roce 2010. Síť vykazuje typické vlastnosti grafu malého světa a prokázala existenci centrálních uzlů, které jsou klíčové pro zajištění spojení mezi letišti. Hustota grafu a nízká assortativita naznačují možné oblasti, kde dochází k omezenému propojení. Tato analýza poskytuje cenné informace pro porozumění tomu, jak jsou letiště propojena a jaká jsou potenciální slabá místa v síti.

Preprocessing a Analýza - Dynamická síť

Dinamickou síť jsem analyzoval pomocí vlastního kódu v Pythonu a knihovny NetworkX. Vypočítal jsem základní metriky, které popisují strukturu a vlastnosti této sítě. Také rozdělil na dané časové úseky.

```
import os
import pandas as pd
import networkx as nx

class Graph:
    def __init__(self):
        self.nodes = []

    def add_point(self, point):
        self.nodes.append(point)

    def load_from_file(self, file_path):
        if not os.path.exists(file_path):
            print("File not found.")
            return

        data = pd.read_csv(file_path, header=None, names=['source', 'target', 'timestamp'])
        for _, row in data.iterrows():
            self.add_point((row['source'], row['target'], row['timestamp']))

    def get_unique_timestamps(self):
        unique_timestamps = set(point[2] for point in self.nodes)
        sorted_timestamps = sorted(unique_timestamps)
        return sorted_timestamps

    def export_cumulative_snapshots(self, output_directory, number_of_snapshots):
        if not os.path.exists(output_directory):
            os.makedirs(output_directory)

        sorted_nodes = sorted(self.nodes, key=lambda x: x[2])
        total_nodes = len(sorted_nodes)
        nodes_per_snapshot = total_nodes // number_of_snapshots

        cumulative_nodes = []

        for i in range(1, number_of_snapshots + 1):
            current_snapshot_limit = min(i * nodes_per_snapshot, total_nodes)
            cumulative_nodes.extend(sorted_nodes[:current_snapshot_limit])

            file_path = os.path.join(output_directory, f'cumulative_snapshot_{i}.csv')

            with open(file_path, 'w') as writer:
                writer.write("Source,Target,Weight\n")
                for node in cumulative_nodes:
                    writer.write(f"{node[0]},{node[1]},1\n")

            snapshot_df = pd.DataFrame(cumulative_nodes, columns=['source', 'target', 'timestamp'])
            analysis = analyze_snapshot(snapshot_df)
            print(f'Cumulative snapshot {i} exported to {file_path}')
            print(f'Snapshot {i} analysis: {analysis}')

def analyze_snapshot(snapshot):
    G = nx.from_pandas_edgelist(snapshot, 'source', 'target')
    if len(G) == 0:
        return {
            'avg_degree': 0,
            'avg_weighted_degree': 0,
            'num_communities': 0,
            'avg_community_size': 0,
            'max_community_size': 0
        }
```

```
    avg_degree = sum(dict(G.degree()).values()) / len(G)
    avg_weighted_degree = sum(dict(G.degree(weight='weight')).values()) / len(G)
    num_communities = nx.number_connected_components(G)
    communities = list(nx.connected_components(G))
    community_sizes = [len(c) for c in communities]
    avg_community_size = sum(community_sizes) / len(community_sizes)
    max_community_size = max(community_sizes)
    return {
        'avg_degree': avg_degree,
        'avg_weighted_degree': avg_weighted_degree,
        'num_communities': num_communities,
        'avg_community_size': avg_community_size,
        'max_community_size': max_community_size
    }

def main():
    file_path = 'contacts-prox-high-school-2013.edges'
    output_directory = 'snapshots'
    number_of_snapshots = 5

    graph = Graph()
    graph.load_from_file(file_path)
    graph.export_cumulative_snapshots(output_directory, number_of_snapshots)

if __name__ == "__main__":
    main()
```

Analýza sítě:

Snapshot	Nodes	Edges	Average Degree	Avg Weighted Degree	Communities	Average Community Size	Max Community Size
1	312	2242	14.6834	14.6834	1	319.0	319
2	326	3765	23.0982	23.0982	1	326.0	326
3	327	4571	27.4740	27.4740	1	327.0	327
4	327	5301	31.9266	31.9266	1	327.0	327
5	327	5815	35.5841	35.5841	1	327.0	327

Table 1: Vývoj průměrného stupně sítě `contacts-prox-high-school-2013` v čase.

průměrný stupeň: ukazuje na posílení propojení v síti (hodnota se zvyšuje). To znamená, že počet hran roste v každém časovém snímku. průměrný vážený stupeň: ukazuje na zvyšování intenzity mezi uzly to znamená, že váha hran taky roste.

hustota: růst hustoty ukazuje na zhuštění grafu, tedy na zvyšování počtu hran vůči počtu možných spojení.

průměrný shlukovači koeficient: zvýšení průměrného shlukovacího koeficientu znamená, že uzly v grafu mají tendenci vytvářet více uzavřených trojúhelníků, ale tato hodnota nezměnila se až tak moc.

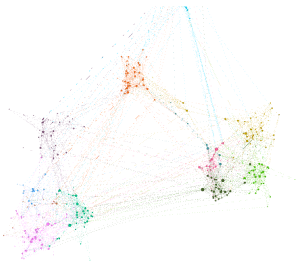
počet komunit: vždy využívalo se resolution = 1, ale počet komunit snížil se, síť stává více integrovanou s menším počtem, ale většími skupinami (komunitami)..

vrchol s největším stupněm: většinou to byl vrchol 106, ale stupeň vrcholu aktivně měnil se. V S3 stal vrchol 106 až třetím po velikosti stupňů. Ale pak znova stal prvním.

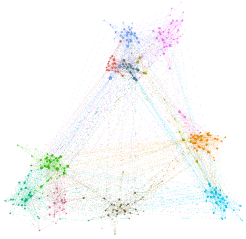
distribuce stupňů: není nějaký stupeň který vyskytoval se by o vele vice krát než ostatní, a je zajímavé ze vrchole z malým stupněm nevyskytuje se vice než z větším.

Zajímavá část grafu s popisem: Tento graf neobsahuje viditelné zajímavé struktury, ale jsem vybrala jeden z vrcholu:

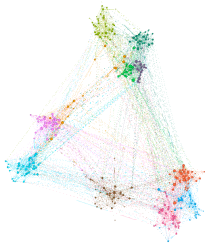
Vizualizace



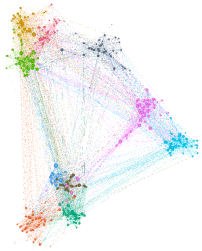
(a) Snímek 1



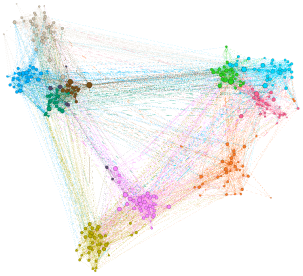
(b) Snímek 2



(c) Snímek 3



(d) Snímek 4

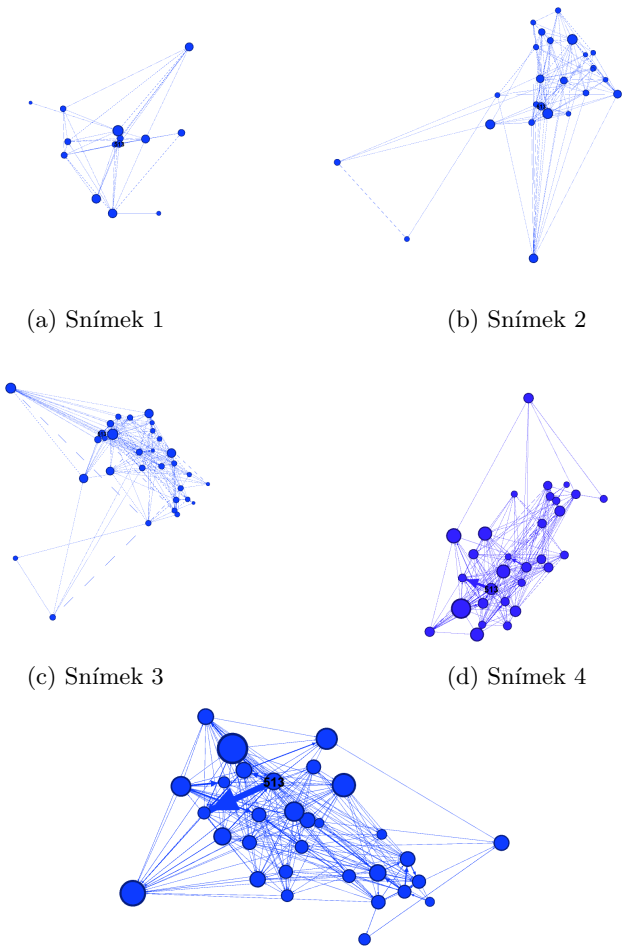


(e) Snímek 5

Figure 4: Vizualizace největších.

Zajímavá část grafu:

Tento graf neobsahuje viditelné zajímavé struktury, ale jsem vybrala jeden z vrcholu:



(a) Snímek 1

(b) Snímek 2

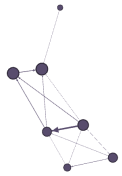
(c) Snímek 3

(d) Snímek 4

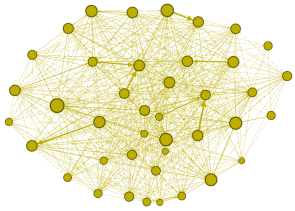
(e) Snímek 5

Figure 5: Vizualizace největších.

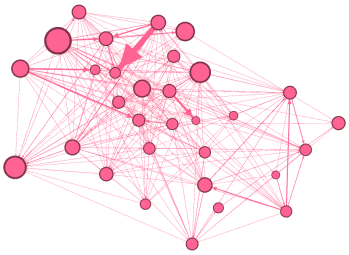
Zajímavá struktury:



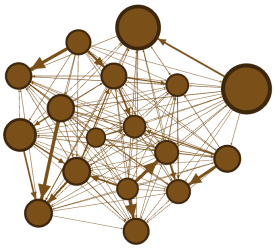
(a) Snímek 1



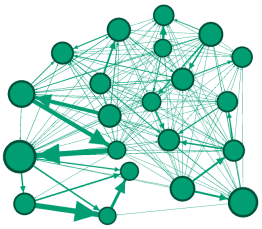
(b) Snímek 2



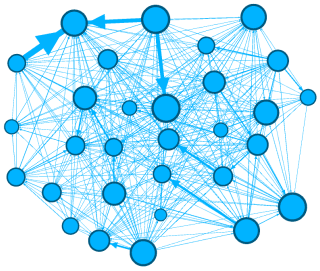
(c) Snímek 3



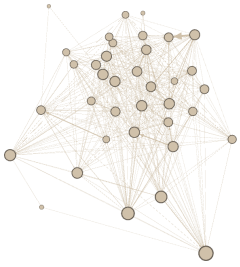
(d) Snímek 4



(e) Snímek 5



(f) Snímek 6



(g) Snímek 7