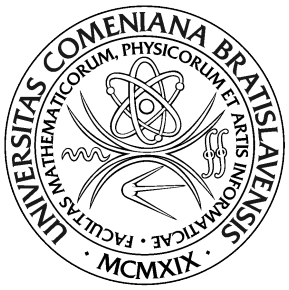
COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

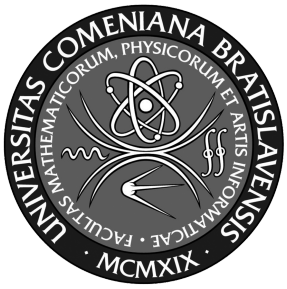


GRAPHLET STRUCTURE ANALYSIS OF THE REAL NETWORKS

Master's thesis

2022 Bc. Michal Puškel

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS



GRAPHLET STRUCTURE ANALYSIS OF THE REAL NETWORKS

Master's thesis

Study programme:	Applied Informatics
Field of study:	2511 Applied Informatics
Training center:	Department of Applied Informatics
Supervisor:	doc. RNDr. Mária Markošová, PhD.

Bratislava, 2022 Bc. Michal Puškel

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Michal Puškel
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Graphlet Structure Analysis of the Real Networks
Analýza grafletovej štruktúry reálnych sietí

Anotácia: Študent vytvorí nový softvér alebo vylepší existujúci softvér na analýzu grafletovej štruktúry komplexných sietí. S pomocou tohto softvéru zanalyzuje grafletovú štruktúru sietí, reálnych dát a vypočíta všetky miery s touto štruktúrou súvisiace. Zanalyzuje tiež grafletovú štruktúru umelo vytvorených sietí a porovná s dátami.

Cieľ: Analyzovať grafletovú štruktúru reálnych sietí pomocou vylepšeného alebo vlastného softvéru.

Vedúci: doc. RNDr. Mária Markošová, PhD.
Konzultant: Mgr. Peter Náther, PhD.
Konzultant: Mgr. Andrej Jursa, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Spôsob sprístupnenia elektronickej verzie práce:
prípustná pre vlastnú VŠ

Dátum zadania: 25.09.2017

Dátum schválenia: 09.10.2017 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

I hereby affirm that this Master's thesis represents my own written work and that I have used no sources and aids other than those indicated. All passages quoted from publications or paraphrased from these sources are properly cited and attributed.

. . . . . .

Bratislava, 2022 Bc. Michal Puškel

Acknowledgement

I would like to thank all people around me supporting me while writing this thesis. I would like to thank especially my supervisor doc. RNDr. Mária Markošová, PhD., as well as all the other members of FMFI Graph research group: doc. RNDr. Tatiana Jajcayová, PhD., Mgr. Andrej Jursa, PhD., RNDr. Martin Nehéz, PhD., Mgr. Peter Náther, PhD., doc. RNDr. Boris Rudolf, PhD. I would like to thank all my family and friends as well.

Abstract

One of the goals of this Master's thesis is to finally find out if an attribute of a functional brain network *„to suffer from the Alzheimer disease"* shows off in the structure of its graph. Because the comparison of the network structure using isomorphism is a hard problem, we have decided to use probabilistic comparison proposed by Nataša Pržulj in her paper Biological Network Comparison Using Graphlet Degree Distribution.

Another goal is to improve our existing network distributed system to compute graphlet degree distribution.

Ultimatelly we implemented our own version of combinatorial approach to graphlet counting (ORCA) inspired by paper of Tomaž Hočevar: A combinatorial approach to graphlet counting.

Keywords: graphs, networks, degree distribution, graphlet degree distribution, orca

Abstrakt

Jedným z cieľov tejto diplomovej práce je konečne zistiť, či sa atribút funkčnej siete mozgu *„mať Alzheimerovu chorobu"* prejavuje v štruktúre siete. Keďže porovnávať štruktúru sietí priamo pomocou izomorfizmu je ťažký problém, rozhodli sme sa využiť metodiku pravdepodobnostného porovnania navrhnutú Natašou Pržulji v práci Biological Network Comparison Using Graphlet Degree Distribution.

Našim ďalším cieľom je vylepšiť naše už existujúce softvérové distribuované paralelizované riešenie na výpočet grafletovej stupňovej distribúcie.

Napokon sme implementovali našu vlastnú verziu kobminatorického prístupu k počítaniu grafletov (ORCA) inšpirovanú publikáciou Tomaža Hočevara: A combinatorial approach to graphlet counting.

Kľúčové slová: grafy, siete, distribúcia stupňa vrcholov, distribúcia orbít grafletov, orca

Contents

Chapter 1

Introduction

This thesis is a direct sequel to our former thesis [Pu7].

Our main goal is to finish research whether the property of functional brain network to suffer from Alzheimer's disease is projected to graph structure.

We need to hotfix our existing software application to correctly enumerate graphlet degree distribution and carry out the experiment again.

Next we need to extend functionality of our application to actually compare graph structure by graphlet degree distribution agreement, the measure proposed by Pržulj [Pr7].

But the essential goal of this thesis is to create our own implementation of combinatorial approach to graphlet counting (ORCA). If possible but not mandatory, to try our best to create faster implemenation to the existing one, provided by Hočevar [HD14a].

Finally we shall verify graphlet degree distribution agreement measure

as suitable for graph structure comparison by comparing some trivially distinguishable networks (for example, Erdős–Rényi, Barabási–Albert, random geometric models).

We explain fundamental theoretical concept in the next chapter 2.

Chapter 2

State of the art

This chapter elaborates fundamental graph theory used by this thesis. This chapter is heavily inspired by several publications: Budinská [Bud16], Vyslúžil [Vys17], Bohumel [Boh19], Gross and Yellen [GYA18], Stanoyevitch [Sta11], West [Wes00], Grimaldi [Gri04], Pržulj [Pr7], [Pr0], Hočevar [HD14a], Puškel [Pu7] and other resources: Ferdinandy [Fer18] . There is no formula or theorem in this thesis that was in invented by ourselves. All theory is cited from mentioned resources. We modified terms slightly for our purposes.

2.1 Graph theory basics

Theorem 1 *Graph G is mathematical model, mirroring relation (firmly defined binary relation) over set of elements - its nodes (vertices) V. Graph is defined by an ordered pair V and E. G, V, E satisfy equations:*

$$G = (V, E), \tag{2.1}$$

$$V(G) = V, \tag{2.2}$$

$$E(G) = E. \tag{2.3}$$

Existence of relation between two elements of set V from equation (2.1) *is called edge. Every edge belonging to graph G belongs to $G's$ edge set E.*

Theorem 2 *If the relation between nodes is one sided, it is called directed edge. Directed edge $e_1 \in E$ is represented by an ordered pair of nodes $e_1 = (v_1, v_2)$, with domain:*

$$E \subseteq V \times V, \tag{2.4}$$

edge e_1 leads from node v_1 to node v_2.

Should the relation be bidirectional, it can be represented with already defined representation (2.4). *We just need to add another directed edge $e_2 = (v_2, v_1)$ to edge set E.*

As an alternative we can define completely new representation of bidirected edge $e_3 = \{v_1, v_2\}$ as unordered pair of vertices i.e. as subset of node set V with 2 elements:

$$E \subseteq [V]^2. \tag{2.5}$$

Theorem 3 *None of representations above* (2.4) *and* (2.5) *is suitable for modelling multiple edges - edges incident with the same pair of nodes (with E being simple set).*

Edge leading from and to the very same node is called loop.

Theorem 4 *Number of vertices of graph G is called the order of graph G:*

$$|V| = n. \tag{2.6}$$

Shall we not note it the other way, variable n or N will always stand for the order of graph in this paper.

Theorem 5 *Number of edges of graph G is called the size of graph G:*

$$|E| = m. \tag{2.7}$$

Shall we not note it the other way, variable m or M will always stand for the size of graph in this paper.

Theorem 6 *Directed graph is graph, that is made up of directed edges.*

Theorem 7 *Undirected graph is graph, that is made up of unordered pairs of vertices / bidirected edges.*

Theorem 8 *Simple graph does not include loops or multiple edges.*

Theorem 9 *Graph containing multiple edges or loops is called multigraph,*

Theorem 10 *Null graph is special edge case graph with properties:*

$$V = \emptyset \wedge E = \emptyset. \tag{2.8}$$

Theorem 11 *Trivial graph is special edge case graph with properties:*

$$|V| = 1 \wedge E = \emptyset. \tag{2.9}$$

In our thesis we will examine simple undirected graphs exclusively.

Theorem 12 *Node v is incident to an edge e if exists such a node u that, edge e satisfies equation: $e = \{v, u\}$.*

Theorem 13 *Two edges e_1 and e_2 are incident, if exists such a node v, and nodes u, w, that satisfies equation: $e_1 = \{u, v\} \wedge e_2 = \{v, w\}$.*

Theorem 14 *Two nodes, u and v are adjacent, if exists edge e that satisfies equation: $e = \{v, u\}$.*

Theorem 15 *Adjacent nodes are called neighbours.*

Theorem 16 *The degree (valency) of a node is the number of edges incident to the node. We usually note degree of node as a function of node v $d(v)$, or as a variable k:*

$$d(v) = k. \tag{2.10}$$

If we do not note otherwise, this notation will be used in this all thesis.

Theorem 17 *Average degree of graph is sum of all node degrees divided by the number of nodes.*

$$d(G) = \bar{k} = \frac{1}{n} \sum_{i=0}^{n-1} d(v_i). \tag{2.11}$$

We can see example of average degree measure on figure 2.1.

Theorem 18 *The degree sequence of a graph is non-increasing sequence of its node degrees.*

We can see example of degree sequence measure on figure 2.2.

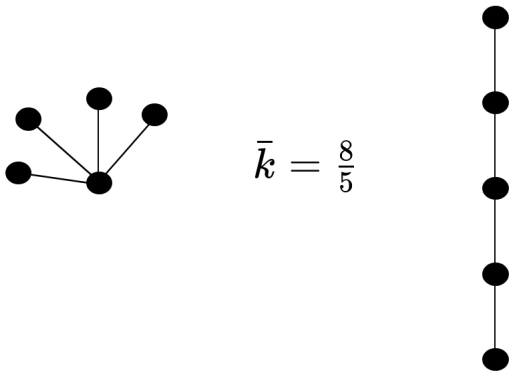


Figure 2.1: Average degree comparison of two graphs.

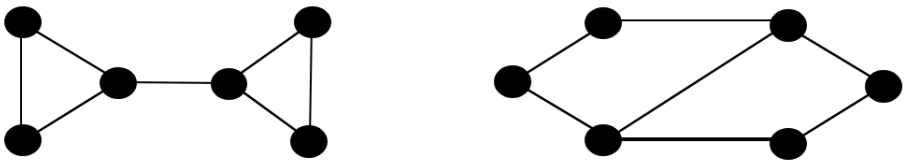


Figure 2.2: Degree sequence comparison of two graphs.

$$Degree\ sequence = (3, 3, 2, 2, 2, 2)$$

Theorem 19 *Degree distribution is a measure of graph examining frequency of particular node degrees existing in graph.*

It is defined as a function mapping node degree onto a count of degrees with that particular degree.

Figure 2.3 shows us degree distribution of graphs depicted in the figure 2.2.

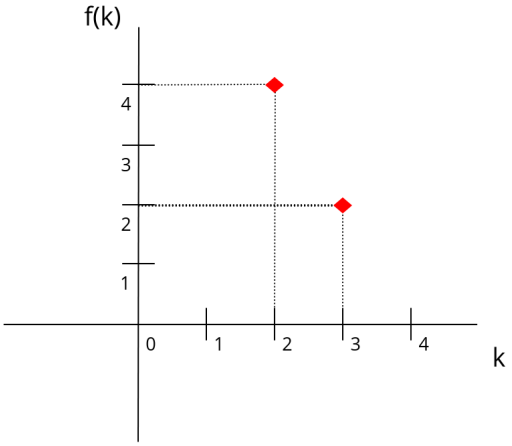


Figure 2.3: Example of degree distribution, where k represents node degree and $f(k)$ stands for the number of nodes with degree k in graph.

Theorem 20 *Isolated node is a node with no neighbours.*

Theorem 21 *Leaf is a node with degree one.*

Theorem 22 *Sum of all nodes' degrees in any ghraph is always an even number. Every edge contributes with 1 degree exactly to 2 nodes, hence that:*

$$\sum_{v \in V} d(v) = 2 \cdot |E|. \tag{2.12}$$

Theorem 23 *Walk (chain) is alternating sequence of nodes and edges. Its length equals to number of edges. Every edge of walk is incident to nodes placed right after and before that particular edge. Walk starts and ends with a node.*

If starting node is same as ending node, we call this walk closed, we call it open otherwise.

Theorem 24 *Trail is walk with distinct edges.*

Theorem 25 *Path is trail with distinct nodes.*

If path starts and ends with the same node, we call it cycle (circuit / tour). We consider cycle as legit only if its length is > 2.

Theorem 26 *Eulerian trail is trail containing every edge of a graph.*

It goes without saying that it can be observed only in graphs containing at most 2 nodes with odd degree.

Theorem 27 *Eulerian cycle is closed Eulerian trail.*

It goes without saying that it can be observed only in graphs containing only nodes with even degree.

Theorem 28 *Eulerian graph is graph containing Eulerian cycle.*

Theorem 29 *Hamiltonian path is path containing every node of graph.*

Theorem 30 *Hamiltonian cycle is Hamiltonian path, which is a cycle at the same time.*

Theorem 31 *Graph is called connected if there is some path between every pair of nodes, otherwise it is called disconnected.*

Theorem 32 *Distance between nodes is length of the shortest path between them.*

Theorem 33 *Graph G' is subgraph of graph G when satisified:*

$$\forall G = (V, E), \forall G' = (V', E'), (V' \subseteq V) \wedge (E' \subseteq E), \tag{2.13}$$

When G' contains all edges from E incident to subset of vertices V' (as well as incident non-edges: non-existing edges between 2 nodes), we call it induced subgraph.

Theorem 34 *Components of graph are connected subgraphs, such that there is no existing path between any two nodes of two distinguished components.*

Theorem 35 *Bridge is such an edge, that by its removal that would cause number of components increase exactly by one.*

Theorem 36 *Articulation point is node incident to bridge. After removal of articulation point we would cause number of components increase at least by one. For instance, P_2 (definition 42) has one bridge, but does not have any articulation point.*

Theorem 37 *Acyclic graph does not include any cycle.*

Theorem 38 *Graph including only nodes with same degree k is called k-regular.*

Theorem 39 *Complete graph is connected $n-1$ regular graph.*

Usually it is called clique K_n, where n stands for number of nodes.

Theorem 40 *When defined graphs $G = (V, E)$ and $G' = (V', E')$ satisfy equation:*

$$(V = V') \wedge (E \cap E' = \emptyset), \tag{2.14}$$

and with union of edge set we get E'':

$$E'' = E \cup E', \tag{2.15}$$

and we create complete graph $G'' = (V, E'')$, then G and G' are called complement (inverse). Graph G is complement of graph G' and graph G' is complement of graph G.

Theorem 41 *Cycle graph is connected 2 regular graph.*

We use C_n for notation of cycle graph with n nodes.

Theorem 42 *Path (linear) graph is connected graph, in which at most two nodes have degree one and other nodes have degree two.*

We use P_n for notation of path graph with n nodes. We can see example of path graphs in the figure 2.4.

Theorem 43 *Bipartite graph is connected graph, which nodes can be divided into two disjoint subsets called partitions. Between any two nodes of same partition must not exist any edge.*

Theorem 44 *When we have bipartite graph containing all possible edges between partitions we call it complete bipartite graph or biclique.*

We use notation $K_{p,q}$, where p and q are vertices counts for particular partitions. Of course next equation is satisfied:

$$p + q = n. \tag{2.16}$$

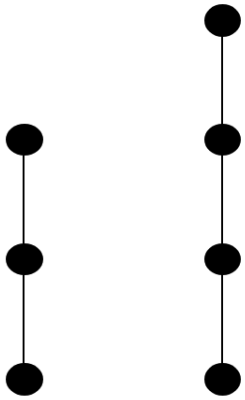


Figure 2.4: Path graphs P_3 and P_4.

Theorem 45 *Star is complete bipartite graph $K_{p,1}$.*

We use notation S_n, where n stands for number of nodes.

Theorem 46 *Connected acyclic graph is called tree.*

Theorem 47 *We recognize two graphs G and G' as isomorphic* (2.17)*, if there is bijection $f : V(G) \to V(G')$ between vertices of graphs G and G', satisfying condition, that any two nodes u and v from G are adjacent if and only if $f(u)$ and $f(v)$ are adjacent in G'. We can see examples of isomorphic graphs in the figures 2.5 and 2.6.*

$$G \cong G', \tag{2.17}$$

Theorem 48 *Automorphism is isomorphism between graph G and G (itself).*

Theorem 49 *Two isomorphic graphs G and G' have isomorphic complements.*

Theorem 50 *If graph is isomorphic with its complement we call it self-complementary graph.*

Theorem 51 *Set of all automorphisms of graph G is called Aut(G)*

Theorem 52 *To show that two graphs are not isomorphic, it is sufficient to distinguish them in any structure measure or in any statistical measure.*

Figure 2.5: Two isomorphic graphs.

Figure 2.6: Petersen graph is probably the most famous graph isomoprhism example.

2.1.1 Complex networks theory basics

"When we model a real, existing system as a graph, we tend to call it a network.", Ferdinandy [Fer18]. Complex networks usually elaborates simple undirected graphs consisting of large number of vertices and edges.

The complex network is a network with nontrivial structure. We consider trivial structure to be typical attributes and features we find in grid graphs, linear graphs or random graphs, see figure 2.7. We can observe nontrivial structure in networks generated on a long term basis and representing real systems. Complex networks may be of various origin Leskovec [LK14], for instance social networks, communication networks, traffic networks, citation networks, cargo distribution networks, video stream channel subscribers' networks etc.

Complex network is a network satisfying following restrictions:

- Network structure is neither random, nor regular (periodic).
- It is generated on a long term basis.
- Network has typical statistical properties such as: degree distribution, average path length, diameter of graph or average clustering coefficient.
- Network contains large number of vertices and edges.

Complex networks elaborate systems especially with large number of objects interacting one with each other, by well defined not random way. We can observe unique nontrivial topological attributes of complex networks described in consequent subsections.

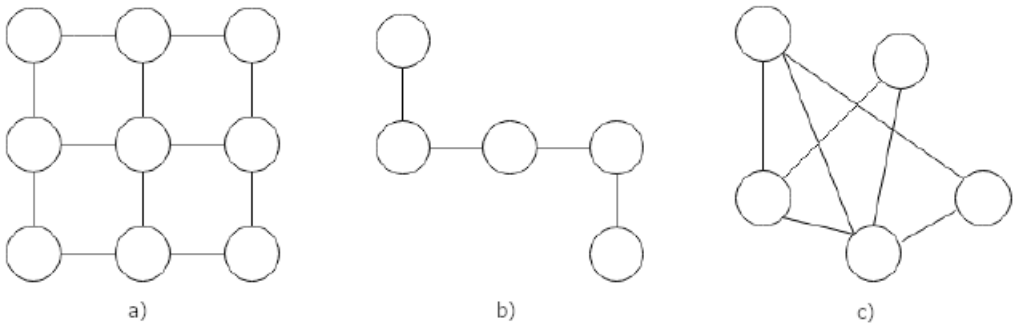


Figure 2.7: Examples of simple networks: a) grid graph, b) linear graph; c) random graph Bohumel [Boh19].

Small-world network

Theorem 53 *Sparse graph is a graph in which the number of edges is close to the minimal number of edges [Ove20].*

When we can traverse any two nodes in sparse graph with the small number of steps we call this network a small-world network. Typical distance of nodes x, y is defined as:

$$L(x, y) \sim logN, \tag{2.18}$$

where $L(x, y)$ stands for distance of randomly chosen nodes x, y and N is number of vertices in network. To further examine small-world network property we can use equation:

$$\sigma = \frac{\frac{\bar{C}}{\bar{C}_r}}{\frac{l}{l_r}}, \tag{2.19}$$

where σ is small-world coefficient, $\bar{C}$ is average clustering coefficient of examined graph, $\bar{C}_r$ is average clustering coefficient of random graph with

similar average degree like examined graph, l is average shortest path of examined graph, l_r is average shortest path of random graph with similar average degree like examined graph. If $\sigma > 1$ is satisfied, we consider network as small-world network.

Small-world networks have typically high clustering coefficient.

Clustering coefficient

Clustering coefficient C_v of a node v is a ratio of number of edges between neighbours of v to number of all possible edges between neighbours of v. Its values range from 0 for star to 1 for clique. It is defined as:

$$C_v = \frac{|E_{nv}|}{\binom{k_v}{2}} = \frac{2 \cdot |E_{nv}|}{k_v \cdot (k_v - 1)}, \tag{2.20}$$

where k_v stands for degree of node v.

Diameter

Diameter is the length of the shortest path between the most distanced nodes of a graph. It measures the extent of a graph and the topological length between two nodes. A high diameter can be observed in less linked networks.

Average path length

Average path length of graph G is defined as the sum off all shortest paths between all nodes of a graph G, divided by count of all possible paths (between all possible pairs of nodes):

$$l_G = \frac{\sum_{x \neq y} dist(x, y)}{n \cdot (n - 1)}, \tag{2.21}$$

where $dist(x, y)$ is the length of shortest path between nodes x, y and n stands for the number of nodes of graph G.

Average clustering coefficient

Average clustering coefficient $\overline{C}$ is a measure of an average probability that arbitrary two neighbours of node i are actually connected, it means they are also neighbours themselves. Average clustering coefficient is defined as:

$$\overline{C} = \frac{\sum_{i=1}^{n} C_i}{n}, \tag{2.22}$$

where n represents the number of nodes in graph G and C_i is clustering coefficient of node i.

Centrality

In theory of graphs, centrality indicators give rankings to nodes within a graph corresponding to their network position.

Betweenness centrality

Betweenness is a centrality measure of a node within a graph. Betweenness centrality measures the count a node z acts as a bridge along the shortest paths between two other nodes x, y. Betweenness centrality is defined as:

$$b(z) = \sum_{x,y} \frac{\sigma_{x,y(z)}}{\sigma_{x,y}}, \tag{2.23}$$

where $\sigma_{x,y(z)}$ represents the count of the shortest paths beginning in node x and ending in node y while traversing node z and $\sigma_{x,y}$ stands for total count of the shortest paths between nodes x and y.

Closeness centrality

Closeness centrality measures how simply a node x of graph G can reach to all the other nodes in the graph G. Closeness centrality is defined as:

$$c(x) = \frac{1}{\sigma}, \tag{2.24}$$

where σ stands for the sum of all shortest distances to all other nodes.

Stability

Stability measure describes how long the network is able to resist random and targeted attacks until the network is decomposed into more components.

Random attack

Random attack is such a node or edge removal from graph G that probability of removal is uniform.

Targeted attack

Targeted attack is such a node or edge removal from graph G that attacker intentionally choose specific node or edge.

Scale-free network

A scale-free network is a network which degree distribution is asymptotically similar to the power law function. De facto it means scale-free networks consist of large number of nodes with small degree and only a couple of nodes with large degree. In scale-free networks degree distribution resembles

asymptotically to power function:

$$P(k) \sim k^{-\gamma}, \tag{2.25}$$

where $\gamma > 1$ represents the scaling exponent. The scaling exponent of scale-free networks usually has values in interval $2 < \gamma < 3$. A power law function is the only normalizable density function $f(k)$ for node degrees in a network that is invariant under rescaling, Broido [BC19], i.e., $f(c \cdot k) = g(c) \cdot f(k)$ for any constant c, and hence that *"free"* of a natural scale. For a network's degree distribution, being scale free implies a power-law pattern, and vice versa. Across scientific domains and classes of networks, it is believed that most or all real-world networks are scale-free, so there is no typical scale that would define the network. Scale-free networks often have similar structure.

Common attributes of real complex networks

- Complex networks contain self-similar structures, it means that specific subgraphs of network are similar to whole network in the means of statistic and topological measures.

- Complex networks have power law degree distribution, that means the degree distribution of complex network asymptotically resembles to function:

$$P(k) \approx c \cdot k^{-\gamma}, \tag{2.26}$$

  where $P(k)$ stands for probability, that randomly chosen node of network will have node degree equal to degree k, c means normalizing

constant, k is node degree and γ is scaling exponent.

- Complex networks may feature attribute small-world network as well.
- Complex networks are vulnerable to targeted attacks, but are quite resistant to random attacks. It means that network will shatter easily and into many components when hubs are targeted. Hub is a node with a number of neighbours that greatly exceeds the average. In case of random attacks there is a great chance that removed node was not the hub and hence that network can resist random attacks quite solid.
- Complex networks may contain communities or cluster structure.
- Complex networks often have hierarchical structure.
- Complex networks are scale-free.

Random and preferential node attachment

Considering the specific measures of the specific network its robustness, evolution, structure and other attributes can be examined. Thanks to knowledge gained from results of various researches, mankind is able to optimize attributes of real networks, for instance: to optimize the structure of highways, electric circuits, to predict a spread of epidemics etc. Such networks are evolving on the long term basis and in accordance to some specific rules, these networks do not arise instantly. In real networks nodes are attached into the network following some preferences. We will elaborate basic principles of node attachment in consequent paragraphs, but for instance we can examine very common rule *rich gets richer* that can be observed also in real life. When some web article has many readers there is a chance that every already existing reader will recommend article to some friend and so it

gets even more readers. The chance for more readers is increasing with the number of readers who can recommend article to potential new readers.

We shall consider that we have a network which grows with time. Every time unit there comes new node x into the network and with m edges it is attached to other nodes of the network. This process may be repeated arbitrary number of times. Two basic methods of network node attachment are: random node attachment and preferential node attachment.

Random node attachment

Random node attachment is such a node attachment, when new node is attached to randomly chosen nodes of network G with m edges whereas randomly chosen nodes are chosen with equal probability of all nodes of network. It means that probability of attachment of new node with every edge of network is:

$$\frac{1}{N(t)}, \tag{2.27}$$

where $N(t)$ stands for the number of nodes in network with time t. Random node attachment is displayed on the figure 2.8.

Preferential node attachment

Preferential node attachment is such an node attachment, when new node is attaching with every edge to already attached nodes of network considering their node degree and is defined by formula:

$$\frac{k_i}{\sum_j k_j}, \tag{2.28}$$

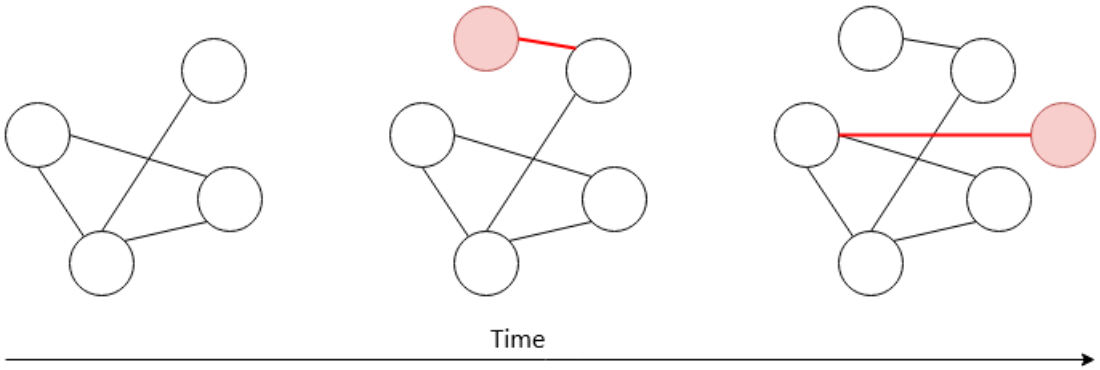


Figure 2.8: Random node attachment with time t. At first graph contains only 4 nodes. Later new nodes are attaching randomly to already attached nodes of graph G Bohumel [Boh19].

This rule is very common in real networks and is well known as the rule *rich gets richer*. It is known that preferential node attachment leads to generating scale-free networks.

Modelling of real networks

In this section we will elaborate several network models that were established to better describe and examine real networks generating. It goes without saying that some of the models are considerably different in terms of network attributes. For that reason specific network models are more suitable for modelling specific network attributes of real networks.

Erdős–Rényi model

Erdős–Rényi (ER) random graphs are generated by following these rules:

- At start of graph generating N nodes arise.
- We choose 2 nodes and we generate random number in interval $[0, 1]$.

If this randomly generated number is greater than probability p, we connect 2 chosen nodes with an edge.

- We repeat the previous rule for every pair of nodes of all possible $\frac{N\cdot(N-1)}{2}$ pairs of nodes in graph.

This model is different from real networks mainly because of:

- There are not arising small local clusters (triangles), and consequently ER graph has low clustering coefficient. On the other hand real networks usually have high clustering coefficient.
- There are no hubs, hence degree distribution asymptotically follows Poisson distribution and so ER graphs are not scale free.

Typical attributes of ER random model are defined by formulas for:

- average node degree:

$$\overline{k} = (N-1)\cdot p. \tag{2.29}$$

- average clustering coefficient:

$$\overline{C} = \frac{\overline{k}}{N}. \tag{2.30}$$

- average shortest path:

$$l \sim \frac{ln(N)}{ln(k)}. \tag{2.31}$$

Watts–Strogatz model

Watts–Strogatz (WS) model is an example of small-world network. Watts–Strogatz model begins at start with regular graph G with N nodes. We reconnect one

end of the edges to another random node defined by probability p. As a result of such a reconnection edges between distant nodes are arising whereas high average clustering coefficient is preserved and so small world network begins to form. If the probability of reconnection β is too high, the final graph begins to be similar to random graph instead of small world network. Figure 2.9 shows us Watts–Strogatz model graph generation.

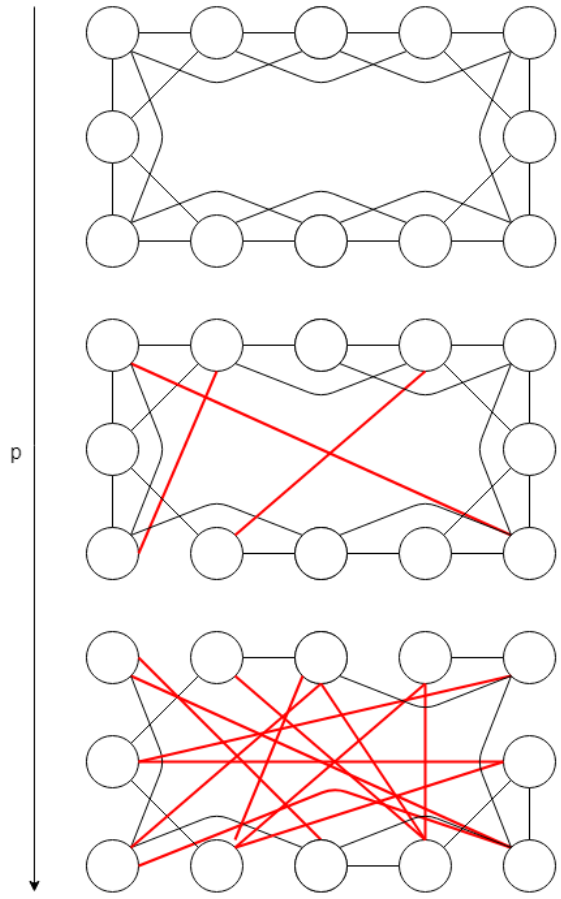


Figure 2.9: With increasing probability of edge reconnection p 3 edges are reconnected. With very high probability of edge reconnection the graph resembles random generated graph Bohumel [Boh19].

The average shortest path is small thanks to shortcuts in small world networks. In a similar way in Watts–Strogatz model such shortcuts are arising thanks to random edge reconnecting. The most significant attributes of WS model are average shortest path, clustering coefficient and degree distribution and are defined by formulas:

- Average shortest path $l(\beta)$ is equal to $l(0) = \frac{N}{2K}$ when $\beta = 0$. When $\beta \to 1$ average shortest path is decreasing, it is similar to but not converging to $l(1) = \frac{lnN}{lnK}$.
- Clustering coefficient is $C(0) = \frac{3 \cdot (K-2)}{4 \cdot (K-1)}$ for $\beta = 0$. For $\beta = 1$ is clustering coefficient the same as in the case of random geometric model graph.
- Degree distribution for $\beta = 0$ is Dirac delta function in K. When $0 < \beta < 1$ degree distribution is defined as:

$$P(k) = \sum_{n=0}^{f(k,K)} \binom{K/2}{n} (1-\beta)^n \beta^{K/2-n} \frac{(\beta K/2)^{k-n-K/2}}{(k-n-K/2)!} e^{-\beta K/2}, \quad (2.32)$$

  where $k = k_i$ is degree of node i and $f(k, K) = min(k - \frac{K}{2}, \frac{K}{2})$.

Kleinberg model

We try to examine how is it possible that people are able to traverse social networks so effectively and easily by using the Kleinberg model. We have examined in Watts–Strogatz model that in small world networks there is short average shortest path, but that does not explain why and how people are so easily traversing the networks.

Simple Kleinberg model begins with grid graph, in which every node shares 4 edges with surrounding 4 neighbours. Later every node gets one

oriented edge (x, y) which is $d(x, y)$ Manhattan-distance-wise large distance from node x to node y. This simple Kleinberg model with shortcuts is displayed on the figure 2.10. In the case of simple Kleinberg model the statistic measures of the graph are very similar like in Watts–Strogatz model.

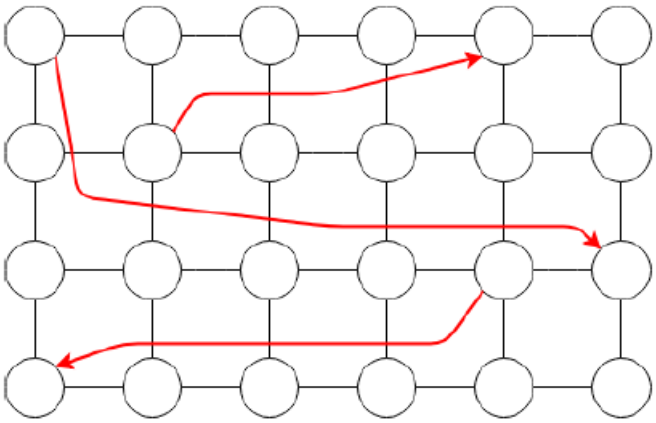


Figure 2.10: Example of adding some edges connecting distant nodes in Kleinberg model - creation of shortcuts Bohumel [Boh19].

More complex Kleinberg model begins again with the grid graph, in which every node shares 4 edges with surrounding 4 neighbours. Later every node gets one oriented edge (x, y), where probability of connection is proportional to distance of nodes x and y. Added long range random links between nodes x and y are added with a probability proportional to $d(x, y)^{-q}$ where q is the clustering exponent and $d(x, y)$ is the Manhattan distance between nodes x and y. Parameter q defines the behaviour of Kleinberg model:

- q = 0, stands for regular distribution, it means that there is same probability of shortcut connection for every node in graph. Probability of connection with distant nodes is very vague in this case.

- q $>$ 0, more distant nodes are chosen. For very high values chosen nodes are not random enough.
- q = 2 is ideal value according to Kleinberg himself for his model.

At the end of the day we are able to define final probability of edges connection in Kleinberg model with formula:

$$P(x,y) = \frac{d(x,y)^{-q}}{\sum_{x \neq y} d(x,y)^{-q}}. \tag{2.33}$$

Barabási–Albert model

Similar to random geometric model also in Barabási–Albert model (BA) the number of nodes in graph is increasing with time. We begin with small simple graph, with time t we add new node x into the graph. Node x comes into the graph with m edges. This new node x is attached with every one of m new edges to other already existing (attached) nodes of graph defined by probability p. The process of node (and edges) attachment can be repeated during the network creation an arbitrary number of times.

Probability p defining the new node with one edge attachment is defined by formula:

$$p = \frac{k_i}{\sum_j k_j}, \tag{2.34}$$

where k_i is node degree of node i. Degree distribution in BA model is asymptotically following the power function and for that reason we consider BA model network as scale free network. The figure 2.11 shows us the preferential node attachment similar to node attachment in Barabási–Albert model.

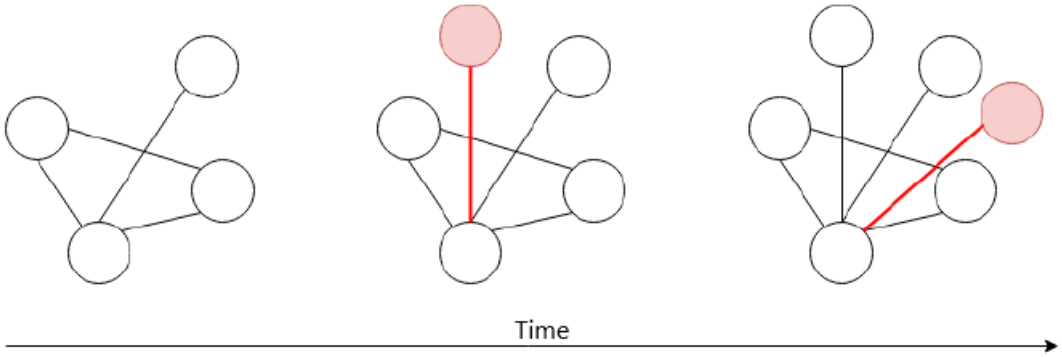


Figure 2.11: Preferential node attachment with time t, where number of edges $m = 1$ Bohumel [Boh19].

Typical attributes of Barabási–Albert model are defined by formulas:

- degree distribution

$$P(k) \sim k^{-3}, \tag{2.35}$$

- average shortest path

$$l \sim \frac{lnN}{ln\, lnN}, \tag{2.36}$$

- average clustering coefficient

$$\bar{C} \sim \frac{(lnN)^2}{N}. \tag{2.37}$$

Random geometric model

Random geometric (GEO) model is simple spatial graph, which was created by placing nodes on random coordinates (X and Y) in the metric space (plane in our case). With time t new nodes are attached into the graph.

We are connecting two nodes by a link if and only if their distance is in a given range, i.e. smaller than a certain neighborhood radius, r. Using such technique of node attachment communities are rising in the graph. To the contrary with other models random geometric model does not have to start from simple starting graph. If we generate network until it is connected we can partially estimate the size of resulting graph based on the size of plane into which we are generating new nodes. Very common modification in random geometric model is increasing the neighborhood radius r with increasing time t.

High average clustering coefficient is very common for random geometric model, but degree distribution is similar to Poisson distribution and so we do not observe scale free behaviour in random geometric model.

2.2 Functional brain networks

In the previous section we have elaborated complex networks, which can help to model networks from real world like for instance networks of interconnections of web pages, networks of personal contacts, but also biological networks - like networks of protein interactions which were examined by Nataša Pržulj [Pr7], and moreover also functional brain networks.

Creation of functional brain network (brain activity network) was possible thanks to fMRI (functional magnetic resonance imaging). FMRI works on the basis of electromagnetic resonance, which is affecting hydrogen in the molecules of water. By magnetic resonance usage an impulse is sent into the body of examined human. Hydrogen acts reactively on this impulse with signal with some strength. Strength of this signal is different for different

ambiences and so it is possible to differentiate grey matter of the brain from the white matter of the brain and from the cerebrospinal fluid on the structure images of the brain.

Neurons need oxygen for its functionality. Increased brain activity invokes increased oxygen consumption in those parts of brain where nerve cells are located. Oxygen is delivered to neurons thanks to hemoglobin in red blood cells. In dependence on the brain activity the oxygenation of the blood in the brain is changing and this oxygenation change is measured by fMRI. This method is called BOLD imaging and it means imaging dependent on the ratio of oxygenated and deoxygenated hemoglobin. At the start of brain activity ratio of oxygenated blood is slightly decreasing, after that it starts to increase and blood oxygenation climax is reached after approximately six seconds since the brain activity start.

The brain activity is measured via fMRI. Simple congnitive tasks are given to examined person, for instance to repeat heard words, to translate given sentences but also mechanical tasks are given like opening and closing the specified palm or rhythmic fingers drumming. Also in some researches examined people do not have to complete any tasks, they simply listen to the music or spoken word.

FMRI output is online imaging of recorded brain activity. FMRI scanner is able to measure signal from small part of brain called voxel. Voxel represents a value on a regular grid in three-dimensional space. 3D image of brain is divided with grid in perpendicular and also in vertical directions into voxels as we can see on the figure 2.12. Voxel is real part of brain of

size approximately $3mm^3$, and we can determine its specific coordinates and location in the brain. In the brain of healthy person there are approximately 10^{11} neurons, so in every voxel there are neurons in the order of thousands.

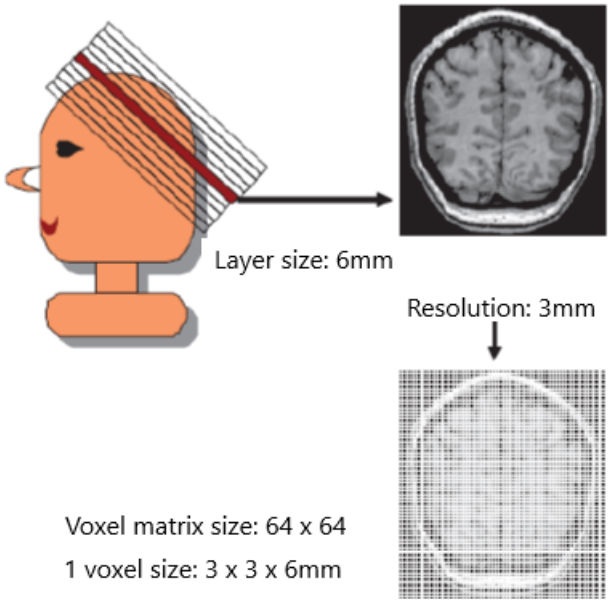


Figure 2.12: 3D voxel matrix scheme for fMRI Budinská [Bud16].

The real question there is how to create functional brain network from measured activities of voxels. Apparently particular voxels will represent nodes of the network. And if there was measured signal correlated above some specified threshold in two voxels, we will add edge between these two voxels. We determine that with calculation of correlation coefficient.

We will note activity of voxel i with time t as $s_i(t)$ and correlation coefficient of voxels i and j as $r(i, j)$. Then if $\langle . \rangle$ is notation for averaging with

time, we can define correlation coefficient with formula:

$$r(i,j) = \frac{\langle s_i(t) \cdot s_j(t) \rangle - \langle s_i(t) \rangle \cdot \langle s_j(t) \rangle}{\sigma(s_i(t)) \cdot \sigma(s_j(t))}, \tag{2.38}$$

where $\sigma(s_i(t))$ means derivation of activity of voxel i with time t. To remove random correlations and also to decrease the number of edges we can use correlation threshold r_c. Correlation threshold stands for the value that defines the minimal threshold of voxels correlation from which and edge will be created in the network. The higher correlation threshold we select the more real connections of voxels we will get, but also that will cause the creation of very small networks, which can be harder to examine for their structure and statistical measures. Lowering the correlation threshold will get us larger networks, but there will be more random correlations present. The optimal values for correlation threshold are approximately in range $[0.8, 0.9]$. Thereafter functional brain network is graph defined by formula:

$$G = (V, E) = \{(u, v) \,|\, \forall\, u, v \in V \wedge r(u, v) > r_c\}. \tag{2.39}$$

Functional brain networks are the object of study for many years. There are many approaches in study of functional brain networks' attributes, in dataset selections and consequently there are many different results from it. For instance it was discovered that functional brain networks seems to have small world networks' behaviour and later it was discovered that functional brain networks are also scale free. Nevertheless we will compare functional brain networks of examined participants between themselves and we will compare only trivially distinguishable complex networks examples of Erdős–Rényi, Barabási–Albert and random geometric models amongst themselves to validate used comparison method.

2.3 Graphlets and graphlet structure analysis

Theorem 54 *Graphlet is induced subgraph isomorphism class in a graph G.*

Induced subgraph means, that subgraph G' includes all edges incident to subset of nodes V' of original graph G (all incident edges and also non-edges, i.e. nonexistent edges between pair of nodes).

If we consider noninduced subgraph we call it motif.

Non-isomorphic graphlets G_0 to G_{29} which Nataša Pržulj [Pr7] examined in her research are shown on the figure 2.13.

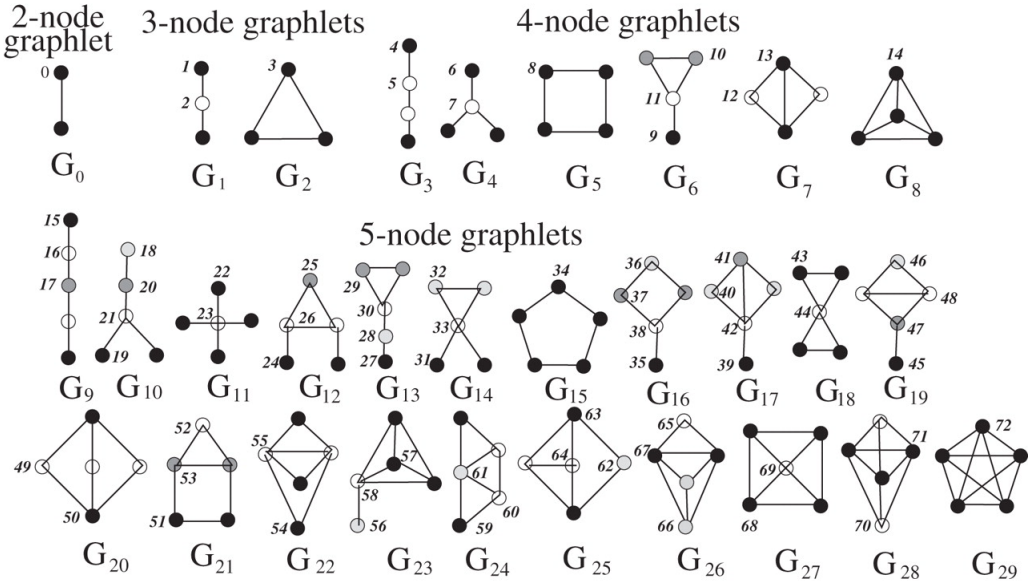


Figure 2.13: Automorphism orbits $0, 1, 2, \ldots, 72$ for the thirty $2, 3, 4$ and $5-node$ graphlets $G_0, G_1, \ldots, G_{29}$. In a graphlet $G_i, i \in \{0, 1, \ldots, 29\}$, nodes belonging to the same orbit are of the same color. Pržulj [Pr7].

2.3.1 Graphlet count

There are many ways how to use graphlets for graphs' structure comparison. The most basic method of structure comparison of two graphs using graphlets is to count graphlets of particular types in both graphs and to compare the counts. When we define $N_i(G)$ as graphlet (occurrence) count of G_i where $1 \leq i \leq 29$ in graph G, then:

$$T(G) = \sum_{i=1}^{29} N_i(G), \tag{2.40}$$

where $T(G)$ stands for the count of all graphlets in graph G.

To compare two graphs relative graphlet frequency distance can be used. To avoid results distortion by the most frequent graphlets occurring in graph G logarithmic scale can be used:

$$F_i(G) = -log\frac{N_i(G)}{T(G)}, \tag{2.41}$$

where $F_i(G)$ stands for the logarithm of ratio of count of graphlet G_i occurrences in the graph G to count of all graphlet occurrences in graph G. Consequently, relative graphlet frequency distance of networks G and H is defined as formula:

$$D(G,H) = \sum_{i=1}^{29} |F_i(G) - F_i(H)|. \tag{2.42}$$

Relative graphlet frequency distance is one of the graph structure similarity measures. The nearer to 0 values of $D(G,H)$ are approaching the more

similar are graphs G and H in the terms of their structure. G_0 is intentionally omitted from relative graphlet frequency distance formula as G_0 essentially represents just basic node degree.

Nevertheless we did not consider relative graphlet frequency distance measure in our research as we did presume it to be not worth the effort and that it would not differentiate examined graphs satisfactory and later results have proven we were right in our presumptions.

2.3.2 Graphlet degree distribution

Graphlet degree distribution (GDD) is other and the main method of structure comparison of two graphs using graphlets. Graphlet degree distribution is actually direct generalization of the notion of the degree distribution. The degree distribution measures, for each value of k, the number of nodes of degree k. It means that, for each value of k, it gives the number of nodes "touching" k edges. Note that an edge is actually the graphlet with two nodes G_0 displayed in the figure 2.13. So at the end of the day, the degree distribution measures: how many nodes "touch" one G_0, how many nodes "touch" two G_0s, ..., how many nodes "touch" k G_0s. Note that there is nothing special about graphlet G_0 and that there is no reason not to apply the same measure to other graphlets. So in graphlet degree distribution (GDD), in addition to applying this measure to an edge, i.e. graphlet G_0, as in the degree distribution, the measure is applied to the all 29 graphlets G_1, G_2, ..., G_{29} displayed in the figure 2.13 as well.

Moreover as specified graphlets from the figure 2.13 are non-isomorphic induced subgraphs, we can determine for each node the number of "touches" with particular graphlets. The "touch" of node v with graphlet G_i means,

that node v is a part of the graphlet G_i. Node v can "touch" specified graphlet G_i by particular node of graphlet G_i, this is illustrated by different colors of nodes on the figure 2.13. Particular nodes of graphlets are divided into different groups (represented by different colors on the figure 2.13) called automorphism orbits.

Automorphism orbit of node v of graph $G = (V, E)$ is defined by formula:

$$Orb(v) = \{u \in V \mid u = g(v),\ g \in Aut(G)\}, \tag{2.43}$$

where $Aut(G)$ is set of all automorphisms of graph G and $g(v)$ is bijection which projects a node to another node of the same graph (automorphism).

"Touch" of node v with orbit O_i means that node is a part of some particular graphlet G_j and this node v is "touching" graphlet G_j with particular orbit O_i. It is needless to say that definition of orbit is the direct generalization of node degree. Node degree specifies only the number of (nearest) neighbours. But orbits do specify closely the topology of node neighbourhood.

Nataša Pržulj [Pr7] specified 73 orbits $O_0, \ldots, O_{72}$ displayed in the figure 2.13 with different colours. To count graphlet degree distribution, we start by measuring the 73 GDDs for each network that we wish to compare. Let G be a network (i.e. a graph). For a particular automorphism orbit j (from the figure 2.13), let $d_G^j(k)$ be the sample distribution of the number of nodes in G "touching" the appropriate graphlet (for automorphism orbit j) k times. That is, d_G^j represents the j^{th} graphlet degree distribution (GDD).

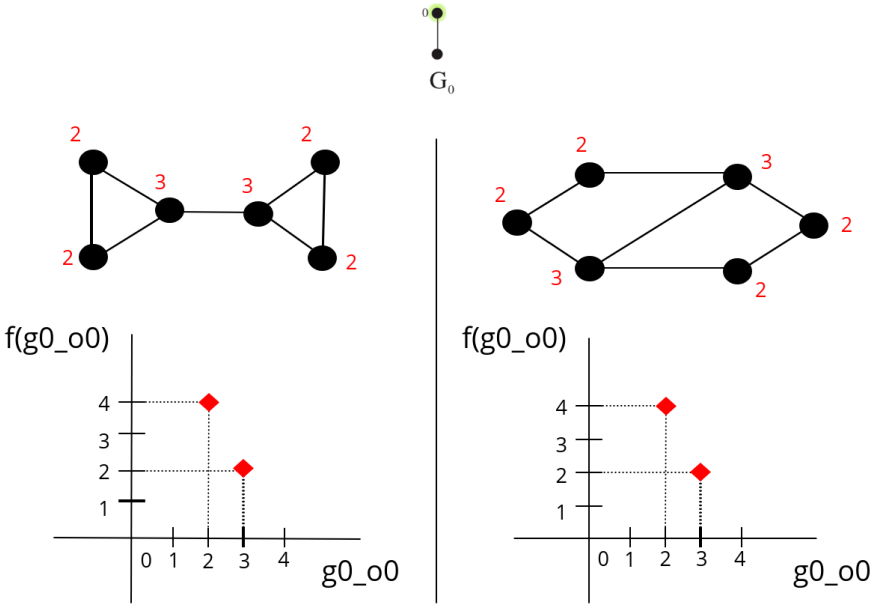


Figure 2.14: Example of graphlet degree distribution for orbit O_0 Puškel [Pu7].

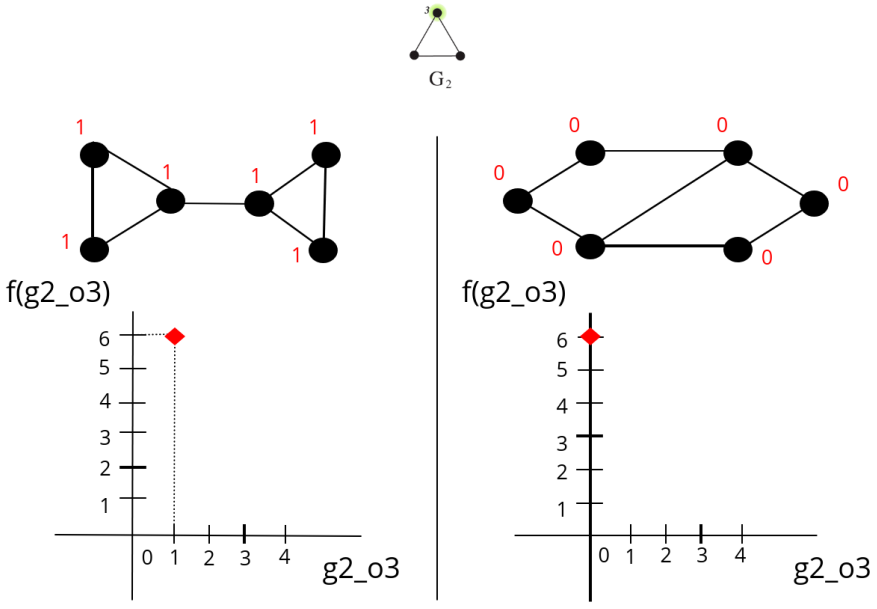


Figure 2.15: Example of graphlet degree distribution for orbit O_3 Puškel [Pu7].

As we can see in the figures 2.14, 2.15 only graphlet degree distribution itself i.e. distribution (i.e. frequency table) d_G^j for some particular orbit j can be sufficient measure to distinguish graph structure in some cases.

But in most cases just simple orbits' comparison is not sufficient enough and more elaborated comparison techniques are required.

To decrease the contribution of larger degrees in a GDD $d_G^j(k)$ is scaled as follows:

$$S_G^j(k) = \frac{d_G^j(k)}{k}, \tag{2.44}$$

and later is normalized with respect to its total area:

$$T_G^j = \sum_{k=1}^{\infty} S_G^j(k), \tag{2.45}$$

$$N_G^j(k) = \frac{S_G^j(k)}{T_G^j}. \tag{2.46}$$

With given normalized distribution $N_G^j(k)$ it is straightforward to compute distance of normalized distributions of two graphs G and H for orbit j:

$$D^j(G,H) = \left(\left(\sum_{i=1}^{\infty} \left[N_G^j(i) - N_H^j(i) \right]^2 \right)^{\frac{1}{2}} \right) \cdot \frac{1}{\sqrt{2}}. \tag{2.47}$$

Distance of normalized distributions $D^j(G,H)$ reach values from range $[0,1]$, where 0 means that networks are topologically identical, whereas 1

means that they are different.

We can invert these value to be more intuitive, we count GDD agreement as follows:

$$A^j(G,H) = 1 - D^j(G,H), \tag{2.48}$$

where j^{th} GDD agreement represents the similarity of two networks according to orbit j.

We count A^j for all 73 orbits. And finally, the agreement between two networks G and H is either the arithmetic 2.49 or geometric 2.50 mean of $A^j(G,H)$ over all j.

$$A_a(G,H) = \frac{1}{73}\sum_{j=0}^{72} A^j(G,H), \tag{2.49}$$

$$A_g(G,H) = \left(\prod_{j=0}^{72} A^j(G,H)\right)^{\frac{1}{73}}. \tag{2.50}$$

The nearer to 1 values of GDD agreement are approaching the more similar are graphs G and H in the terms of their structure. It is needless to say that we did use GDD to compare graphs in this thesis.

2.3.3 Combinatorial approach to graphlet counting

To compute GDD is an computationally challenging task. And even we counted GDD by brute force in our previous thesis [Pu7] which worked considerably well given the large input graphs which where counted by computer stations with rather low operating memory resources at their disposal, there is a better way.

Tomaž Hočevar and Janez Demšar proposed a combinatorial approach to graphlet counting: Orbit Counting Algorithm (ORCA) in their research [HD14a].

ORCA counts vector of (all 73) orbit "touches" for each node u of graph G. As we already know, it means we get for each node u the number of occurrences in respective orbits $0, \ldots, 72$ of graphlets $G_0, \ldots, G_{29}$ in graph G.

The algorithm uses system of linear equations which describes relations between graphlets of smaller number of nodes with graphlets of higher number of nodes. For instance when node $x \in V$ of graph $G = (V, E)$ is part of some k node graphlet G_i, then it is also part of some other $k - 1$ graphlet G_j.

As we can see in the figure 2.16 when we remove any most distant node y from node x which is part of 5-node graphlet G_{17} then we get 4-node graphlet G_7 which also includes node x. ORCA uses this knowledge in reverse direction. Every 4-node graphlet can be composed by adding one node to some specific 3-node graphlet (to some node of specific orbit in graphlet) and every 5-node graphlet can be composed by adding one node to some specific 4-node graphlet, ...

The number of these linear equations is $n - 1$, where n is the number of orbits, so we can directly enumerate only just one orbit for every node and the rest is derived, calculated from the equation system and as a result we get for every node the number of "touches" (occurrences) in respective orbits.

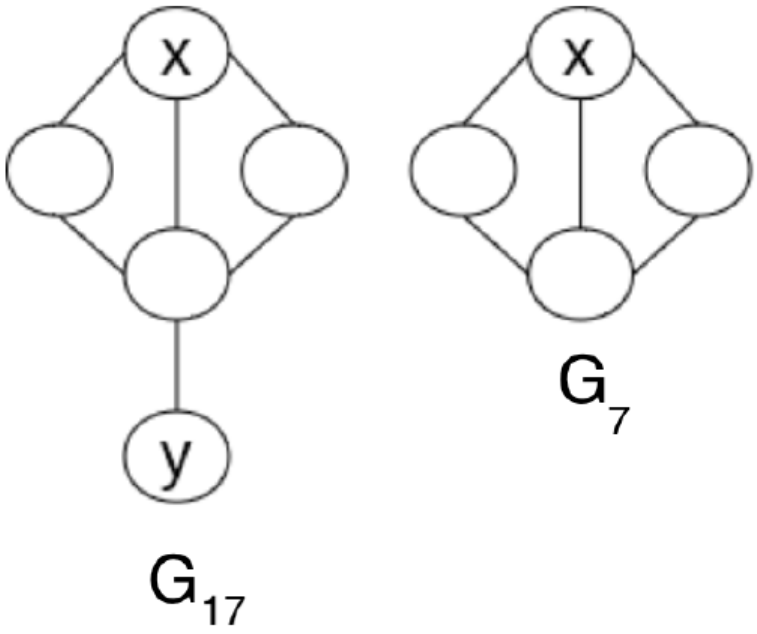


Figure 2.16: Decomposition of graphlet G_{17} into graphlet G_7 Vyslúžil [Vys17].

Let $c(u, v) = |N(u) \cap N(v)|$ denotes the number of common neighbours of nodes u and v. Let $p(u, v)$ denotes the number of paths on three nodes that start at node u, continue with v and end with some node t, which is not connected to u. We can compute $p(u, v)$ as $d(v) - 1 - c(u, v)$.

Let nodes x, y and z of examined graph G induce graphlet G_1, a path on 3 nodes; we will observe its extensions to 4-node graphlets with the fourth node, w, connected to y and z (dashed lines). The number of such nodes w is $c(y, z)$. In the example shown in the figure 2.17(a) there are $c(y, z) = 3$

such nodes, which are marked by w_1, w_2 and w_3.

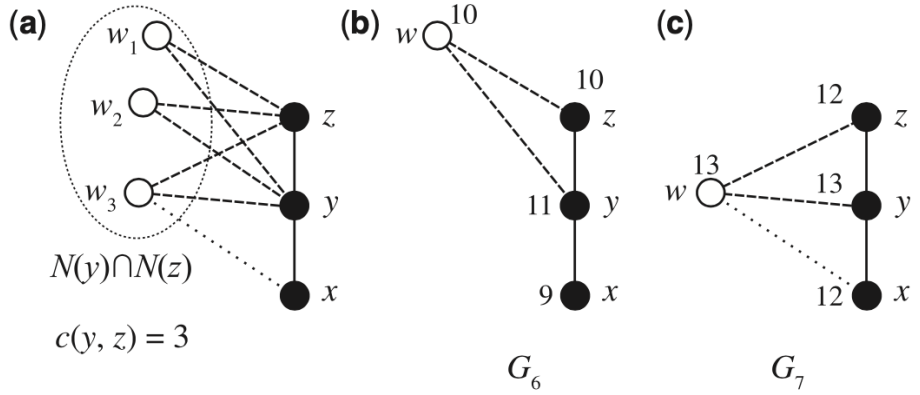


Figure 2.17: Relation between orbits O_9 and O_{12}. Solid lines represents edges in the 3-node graphlet G_1 being extended. Dashed lines exist by definition: w (or w_i) are the common neighbours of y and z. Dotted lines are optional edges that make the resulting 4-node graphlet on x, y, z and w_i isomorphic to G_6 or G_7 Hočevar [HD14a].

The edge $\{x, w\}$ might exist in the graph G (as in the case of w_3, the dotted line) or not (as for w_1 and w_2). With no edge, nodes x, y, z and w form a paw (G_6) with x in orbit O_9 2.17(b). With an edge between x and w, they form a diamond (G_7) with x in orbit O_{12} 2.17(c). Because all $c(y, z)$ nodes in $N(y) \cap N(z)$ must participate either in G_6 or in G_7, which puts x in O_9 or in O_{12}, this gives $o_9 + o_{12} = c(y, z)$ for the particular triplet x, y and z. We sum this over all possible three-node paths starting at x. Summation must account for symmetries: each graphlet G_6 appearing in the graph is counted twice with roles of z and w reversed, and G_7 is counted twice with reversed roles of y and w. As a result we get formula for relation between

counts of $c(y, z)$ for some specific x and its occurrences in orbits O_9 and O_{12}:

$$2o_9 + 2o_{12} = \sum_{\substack{y,z:\, x,z \in N(y) \\ G[\{x,y,z\}] \cong G_1]}} c(y, z), \tag{2.51}$$

where $\cong$ denotes graph isomorphism (e.g. $G[\{x, y, z\}]$, a subgraph on nodes x, y and z is isomorphic to G_1, a path with three nodes).

There are only two 3-node graphlets and relatively few possible extensions. Investigating all possibilities in a similar manner yields 10 linearly independent equations with 11 variables that correspond to counts of 11 orbits in 4-node graphlets, as Tomaž Hočevar and Janez Demšar proposed [HD14a]:

$$\begin{aligned}
o_{12} + 3o_{14} &= \sum_{y,z:\, y<z, G[\{x,y,z\}] \cong G_2} c(y, z) - 1 \\
2o_{13} + 6o_{14} &= \sum_{y,z:\, y<z, G[\{x,y,z\}] \cong G_2} (c(x, y) - 1) + (c(x, z) - 1) \\
o_{10} + 2o_{13} &= \sum_{y,z:\, y<z, G[\{x,y,z\}] \cong G_2} p(y, z) + p(z, y) \\
2o_{11} + 2o_{13} &= \sum_{y,z:\, y<z, G[\{x,y,z\}] \cong G_2} p(y, x) + p(z, x) \\
6o_7 + 2o_{11} &= \sum_{y,z:\, y<z, y,z \in N(x), G[\{x,y,z\}] \cong G_1} (p(y, x) - 1) + (p(z, x) - 1) \\
o_5 + 2o_8 &= \sum_{y,z:\, y<z, y,z \in N(x), G[\{x,y,z\}] \cong G_1} p(x, y) + p(x, z) \\
2o_6 + 2o_9 &= \sum_{y,z:\, x,z \in N(y), G[\{x,y,z\}] \cong G_1} p(x, y) - 1 \\
2o_9 + 2o_{12} &= \sum_{y,z:\, x,z \in N(y), G[\{x,y,z\}] \cong G_1} c(y, z) \\
o_4 + 2o_8 &= \sum_{y,z:\, x,z \in N(y), G[\{x,y,z\}] \cong G_1} p(y, z) \\
2o_8 + 2o_{12} &= \sum_{y,z:\, x,z \in N(y), G[\{x,y,z\}] \cong G_1} c(x, z) - 1
\end{aligned}$$

Figure 2.18: Equations for orbit counts in 4-graphlets Hočevar [HD14a].

Right sides depend on the graph G and need to be computed for each point x. To accelerate their computation, values of $c(u, v)$ and $p(u, v)$ are precomputed. In all equations, except for the last one, $c(u, v)$ is computed on pairs of connected nodes (u, v); in $p(u, v)$, they are connected by the definition of p. Therefore, it suffices to precompute $c(u, v)$ and $p(u, v)$ only for all pairs of connected nodes u and v; the last equation, in which the new node closes a cycle, is treated separately.

In a very similar manner equations for orbit counts in 5-graphlets were proposed:

$$
\begin{aligned}
P_{14}(x,u,v,t) &= u < v < t \wedge G[\{x,u,v,t\}] \cong G_8 \\
P_{13}(x,u,v,t) &= v < t \wedge (v,t) \notin E \wedge G[\{x,u,v,t\}] \cong G_7 \\
P_{12}(x,u,v,t) &= u < v \wedge (x,t) \notin E \wedge G[\{x,u,v,t\}] \cong G_7 \\
P_{11}(x,u,v,t) &= u < v \wedge u,v \notin N(t) \wedge G[\{x,u,v,t\}] \cong G_6 \\
P_{10}(x,u,v,t) &= x,u \notin N(t) \wedge G[\{x,u,v,t\}] \cong G_6 \\
P_{9}(x,u,v,t) &= v < t \wedge v,t \notin N(x) \wedge G[\{x,u,v,t\}] \cong G_6 \\
P_{8}(x,u,v,t) &= u < v \wedge u,v \in N(x) \wedge G[\{x,u,v,t\}] \cong G_5 \\
P_{7}(x,u,v,t) &= u < v < t \wedge u,v,t \in N(x) \wedge G[\{x,u,v,t\}] \cong G_4 \\
P_{6}(x,u,v,t) &= v < t \wedge x,v,t \in N(u) \wedge G[\{x,u,v,t\}] \cong G_4 \\
P_{5}(x,u,v,t) &= u,v \in N(x) \wedge t \in N(v) \wedge G[\{x,u,v,t\}] \cong G_3 \\
P_{4}(x,u,v,t) &= x,v \in N(u) \wedge t \in N(v) \wedge G[\{x,u,v,t\}] \cong G_3
\end{aligned}
$$

Figure 2.19: Conditions for 5-graphlet orbit count equations, P_i, define the order of nodes and put x in orbit O_i; e.g. in P_{13} node x is in orbit O_{13}. Some right-hand sides refer to the number of common neighbours of three nodes $|N(u) \cap N(v) \cap N(t)|$; with some abuse of notation, it is written as $c(u, v, t)$. For consistency, $c(u)$ is also used to denote the degree of a point, $|N(u)|$. Hočevar [HD14a].

$$2o_{71} + 12o_{72} = \sum_{u,v,t:\ P_{14}(x,u,v,t)} (c(x,u,v) - 1) + (c(x,u,t) - 1) + (c(x,v,t) - 1)$$

$$o_{70} + 4o_{72} = \sum_{u,v,t:\ P_{14}(x,u,v,t)} c(u,v,t) - 1$$

$$4o_{69} + 2o_{71} = \sum_{u,v,t:\ P_{13}(x,u,v,t)} c(x,v,t) - 1$$

$$o_{68} + 2o_{71} = \sum_{u,v,t:\ P_{13}(x,u,v,t)} c(u,v,t) - 1$$

$$o_{67} + 12o_{72} + 4o_{71} = \sum_{u,v,t:\ P_{14}(x,u,v,t)} (c(x,u) - 2) + (c(x,v) - 2) + (c(x,t) - 2)$$

$$o_{66} + 12o_{72} + 2o_{71} + 3o_{70} = \sum_{u,v,t:\ P_{14}(x,u,v,t)} (c(u,v) - 2) + (c(u,t) - 2) + (c(v,t) - 2)$$

$$2o_{65} + 3o_{70} = \sum_{u,v,t:\ P_{12}(x,u,v,t)} c(u,v,t)$$

$$o_{64} + 2o_{71} + 4o_{69} + o_{68} = \sum_{u,v,t:\ P_{13}(x,u,v,t)} c(v,t) - 2$$

$$o_{63} + 3o_{70} + 2o_{68} = \sum_{u,v,t:\ P_{12}(x,u,v,t)} c(x,t) - 2$$

$$2o_{62} + o_{68} = \sum_{u,v,t:\ P_{8}(x,u,v,t)} c(u,v,t)$$

$$2o_{61} + 4o_{71} + 8o_{69} + 2o_{67} = \sum_{u,v,t:\ P_{13}(x,u,v,t)} (c(x,v) - 1) + (c(x,t) - 1)$$

$$o_{60} + 4o_{71} + 2o_{68} + 2o_{67} = \sum_{u,v,t:\ P_{13}(x,u,v,t)} (c(u,v) - 1) + (c(u,t) - 1)$$

$$o_{59} + 6o_{70} + 2o_{68} + 4o_{65} = \sum_{u,v,t:\ P_{12}(x,u,v,t)} (c(u,t) - 1) + (c(v,t) - 1)$$

$$o_{58} + 4o_{72} + 2o_{71} + o_{67} = \sum_{u,v,t:\ P_{14}(x,u,v,t)} c(x) - 3$$

$$o_{57} + 12o_{72} + 4o_{71} + 3o_{70} + o_{67} + 2o_{66} = \sum_{u,v,t:\ P_{14}(x,u,v,t)} (c(u) - 3) + (c(v) - 3) + (c(t) - 3)$$

Figure 2.20: Equations for orbit counts in 5-graphlets Hočevar [HD14a].

$$3o_{56} + 2o_{65} = \sum_{u,v,t:\ P_9(x,u,v,t)} c(u,v,t)$$

$$3o_{55} + 2o_{71} + 2o_{67} = \sum_{u,v,t:\ P_{13}(x,u,v,t)} c(x,u) - 2$$

$$2o_{54} + 3o_{70} + o_{66} + 2o_{65} = \sum_{u,v,t:\ P_{12}(x,u,v,t)} c(u,v) - 2$$

$$o_{53} + 2o_{68} + 2o_{64} + 2o_{63} = \sum_{u,v,t:\ P_8(x,u,v,t)} c(x,u) + c(x,v)$$

$$2o_{52} + 2o_{66} + 2o_{64} + o_{59} = \sum_{u,v,t:\ P_{10}(x,u,v,t)} c(u,t) - 1$$

$$o_{51} + 2o_{68} + 2o_{63} + 4o_{62} = \sum_{u,v,t:\ P_8(x,u,v,t)} c(u,t) + c(t,v)$$

$$3o_{50} + o_{68} + 2o_{63} = \sum_{u,v,t:\ P_8(x,u,v,t)} c(x,t) - 2$$

$$2o_{49} + o_{68} + o_{64} + 2o_{62} = \sum_{u,v,t:\ P_8(x,u,v,t)} c(u,v) - 2$$

$$o_{48} + 4o_{71} + 8o_{69} + 2o_{68} + 2o_{67} + 2o_{64} + 2o_{61} + o_{60} = \sum_{u,v,t:\ P_{13}(x,u,v,t)} (c(v) - 2) + (c(t) - 2)$$

$$o_{47} + 3o_{70} + 2o_{68} + o_{66} + o_{63} + o_{60} = \sum_{u,v,t:\ P_{12}(x,u,v,t)} c(x) - 2$$

$$o_{46} + 3o_{70} + 2o_{68} + 2o_{65} + o_{63} + o_{59} = \sum_{u,v,t:\ P_{12}(x,u,v,t)} c(t) - 2$$

$$o_{45} + 2o_{65} + 2o_{62} + 3o_{56} = \sum_{u,v,t:\ P_9(x,u,v,t)} c(v,t) - 1$$

$$4o_{44} + o_{67} + 2o_{61} = \sum_{u,v,t:\ P_{11}(x,u,v,t)} c(x,t)$$

$$2o_{43} + 2o_{66} + o_{60} + o_{59} = \sum_{u,v,t:\ P_{10}(x,u,v,t)} c(v,t)$$

$$o_{42} + 2o_{71} + 4o_{69} + 2o_{67} + 2o_{61} + 3o_{55} = \sum_{u,v,t:\ P_{13}(x,u,v,t)} c(x) - 3$$

$$o_{41} + 2o_{71} + o_{68} + 2o_{67} + o_{60} + 3o_{55} = \sum_{u,v,t:\ P_{13}(x,u,v,t)} c(u) - 3$$

$$o_{40} + 6o_{70} + 2o_{68} + 2o_{66} + 4o_{65} + o_{60} + o_{59} + 4o_{54} = \sum_{u,v,t:\ P_{12}(x,u,v,t)} (c(u) - 3) + (c(v) - 3)$$

$$2o_{39} + 4o_{65} + o_{59} + 6o_{56} = \sum_{u,v,t:\ P_9(x,u,v,t)} (c(u,v) - 1) + (c(u,t) - 1)$$

$$o_{38} + o_{68} + o_{64} + 2o_{63} + o_{53} + 3o_{50} = \sum_{u,v,t:\ P_8(x,u,v,t)} c(x) - 2$$

$$o_{37} + 2o_{68} + 2o_{64} + 2o_{63} + 4o_{62} + o_{53} + o_{51} + 4o_{49} = \sum_{u,v,t:\ P_8(x,u,v,t)} (c(u) - 2) + (c(v) - 2)$$

$$o_{36} + o_{68} + 2o_{63} + 2o_{62} + o_{51} + 3o_{50} = \sum_{u,v,t:\ P_8(x,u,v,t)} c(t) - 2$$

$$2o_{35} + o_{59} + 2o_{52} + 2o_{45} = \sum_{u,v,t:\ P_4(x,u,v,t)} c(u,t) - 1$$

$$2o_{34} + o_{59} + 2o_{52} + o_{51} = \sum_{u,v,t:\ P_4(x,u,v,t)} c(x,t)$$

$$2o_{33} + o_{67} + 2o_{61} + 3o_{58} + 4o_{44} + 2o_{42} = \sum_{u,v,t:\ P_{11}(x,u,v,t)} c(x) - 3$$

Figure 2.21: Equations for orbit counts in 5-graphlets Hočevar [HD14a].

$$2o_{32} + 2o_{66} + o_{60} + o_{59} + 2o_{57} + 2o_{43} + 2o_{41} + o_{40} = \sum_{u,v,t:\ P_{10}(x,u,v,t)} c(v) - 3$$

$$o_{31} + 2o_{65} + o_{59} + 3o_{56} + o_{43} + 2o_{39} = \sum_{u,v,t:\ P_{9}(x,u,v,t)} c(u) - 3$$

$$o_{30} + o_{67} + o_{63} + 2o_{61} + o_{53} + 4o_{44} = \sum_{u,v,t:\ P_{11}(x,u,v,t)} c(t) - 1$$

$$o_{29} + 2o_{66} + 2o_{64} + o_{60} + o_{59} + o_{53} + 2o_{52} + 2o_{43} = \sum_{u,v,t:\ P_{10}(x,u,v,t)} c(t) - 1$$

$$o_{28} + 2o_{65} + 2o_{62} + o_{59} + o_{51} + o_{43} = \sum_{u,v,t:\ P_{9}(x,u,v,t)} c(x) - 1$$

$$2o_{27} + o_{59} + o_{51} + 2o_{45} = \sum_{u,v,t:\ P_{4}(x,u,v,t)} c(v,t)$$

$$o_{26} + 2o_{67} + 2o_{63} + 2o_{61} + 6o_{58} + o_{53} + 2o_{47} + 2o_{42} = \sum_{u,v,t:\ P_{11}(x,u,v,t)} (c(u) - 2) + (c(v) - 2)$$

$$2o_{25} + 2o_{66} + 2o_{64} + o_{59} + 2o_{57} + 2o_{52} + o_{48} + o_{40} = \sum_{u,v,t:\ P_{10}(x,u,v,t)} (c(u) - 2)$$

$$o_{24} + 4o_{65} + 4o_{62} + o_{59} + 6o_{56} + o_{51} + 2o_{45} + 2o_{39} = \sum_{u,v,t:\ P_{9}(x,u,v,t)} (c(v) - 2) + (c(t) - 2)$$

$$4o_{23} + o_{55} + o_{42} + 2o_{33} = \sum_{u,v,t:\ P_{7}(x,u,v,t)} c(x) - 3$$

$$3o_{22} + 2o_{54} + o_{40} + o_{39} + o_{32} + 2o_{31} = \sum_{u,v,t:\ P_{6}(x,u,v,t)} c(u) - 3$$

$$o_{21} + 3o_{55} + 3o_{50} + 2o_{42} + 2o_{38} + 2o_{33} = \sum_{u,v,t:\ P_{7}(x,u,v,t)} (c(u) - 1) + (c(v) - 1) + (c(t) - 1)$$

$$o_{20} + 2o_{54} + 2o_{49} + o_{40} + o_{37} + o_{32} = \sum_{u,v,t:\ P_{6}(x,u,v,t)} c(x) - 1$$

$$o_{19} + 4o_{54} + 4o_{49} + o_{40} + 2o_{39} + o_{37} + 2o_{35} + 2o_{31} = \sum_{u,v,t:\ P_{6}(x,u,v,t)} (c(v) - 1) + (c(t) - 1)$$

$$2o_{18} + o_{59} + o_{51} + 2o_{46} + 2o_{45} + 2o_{36} + 2o_{27} + o_{24} = \sum_{u,v,t:\ P_{4}(x,u,v,t)} c(v) - 2$$

$$2o_{17} + o_{60} + o_{53} + o_{51} + o_{48} + o_{37} + 2o_{34} + 2o_{30} = \sum_{u,v,t:\ P_{5}(x,u,v,t)} c(u) - 1$$

$$o_{16} + o_{59} + 2o_{52} + o_{51} + 2o_{46} + 2o_{36} + 2o_{34} + o_{29} = \sum_{u,v,t:\ P_{4}(x,u,v,t)} c(x) - 1$$

$$o_{15} + o_{59} + 2o_{52} + o_{51} + 2o_{45} + 2o_{35} + 2o_{34} + 2o_{27} = \sum_{u,v,t:\ P_{4}(x,u,v,t)} c(t) - 1$$

Figure 2.22: Equations for orbit counts in 5-graphlets Hočevar [HD14a].

All source code to ORCA is available on website [HD14b].

2.4 Other similar theses

Our thesis is based on the fundamental knowledge of other previous similar theses.

2.4.1 Graphlets in the functional brain networks

Student Mgr. Lucia Budinská has analysed real data of functional brain networks by graphlet count of $G_1, \ldots, G_8$ and graphlet degree distribution (GDD) of 3-node graphlets and G_8 in her Master's thesis Graphlets in the functional brain networks [Bud16]. This solution uses RAGE algorithm. She has compared measured results of functional brain networks with results of BA and WS models. Her results has proven that by GDD comparison small world networks are similar to functional brain networks only by 30% but her real data of functional brain networks are similar to BA model by 60%.

2.4.2 Graphlet in complex networks

Student Mgr. Marián Vyslúžil has analysed real data of functional brain networks in his Master's thesis Graphlet in complex networks [Vys17]. Dataset was divided into 3 groups of participants: young, healthy elderly and elderly participants with diagnosed Alzheimer's disease. He used GDD to compare graphs between themselves with his own implementation of ORCA algorithms. It was somewhat successful even groups of participants could not be differentiated by GDD as their topology seems to be very similar. Also his own implementation was better in some cases than original ORCA implementation, though further improvements could be done in some later research as CUDA parallelism was not good enough because of problems with shared memory between CUDA threads.

2.4.3 Individual social network analysis

Student Mgr. Martin Bohumel has analysed real data of social networks in his Master's thesis Individual social network analysis [Boh19]. He compared social networks between themselves and with artificial data of well known models as BA, ER, WS, GEO. He used standard graph statistics as well as graphlet count with usage of available ORCA software to perform comparisons. Measured data was reviewed and behaviour of graphs explained by theoretical graph models.

2.4.4 Probability comparison of functional brain networks

In our Bachelor's thesis Probability comparison of functional brain networks [Pu7] we have created our own software to count GDD and we have analysed real data of functional brain networks. Dataset of 40 participants was divided into 3 groups of participants: young, healthy elderly and elderly participants with diagnosed Alzheimer's disease and we got it from Buckner's research [BSS+00].

Dataset was rather large, for instance the biggest graph had $|V| = 8394$, $|E| = 450042$ and $max\, d(G) = 651$ and unfortunately we did not manage to compare GDD of these large graphs by the scope of Bachelor's thesis for various problems e.g. there was problem with overflowing integer and other bugs in our software and even we have managed to fix it we could not perform the measurement once again for the tight time schedule and even GDD agreement comparison module was not yet fully implemented by that time so all this and even more remained as subject of research for this Master's thesis.

It is needless to say that our own software (or rather API / terminal application) was quite successful, it uses our own brainstormed brute force likewise algorithms, and it is network distributed system with master and workers' computer stations which even uses processor multi threading and is platform independent and moreover in some very special cases under some very special circumstances it features better performance results than ORCA.

Chapter 3

Solution

3.1 Goals

To finish our project we have these goals:

- Implement GDD agreement comparison feature in our existing software.
- Make our own implementation of ORCA.
- Measure GDD agreements of 3 groups of participants divided by Alzheimer's disease diagnosis.
- Measure GDD agreements of some ER, BA, GEO graphs.
- Compare and evaluate measured GDD agreements, and later decide if GDD agreement measure is reliable for graph comparison, and if and only if yes, then later decide whether Alzheimer's disease is projected to functional brain networks graph structure.

The main purpose of this chapter is to clarify what and how was done in our research. Since it all started as Bachelor's thesis many new features were

added, huge heap of code was refactored, many bugs were fixed and tremendous computationally complex amount of graphs were again or additionally measured for statistics and reviewed. With the exception of first measurements that were carried out in labs of our university, all other experiments were performed by our home desktop PC with `Intel© Core™ i5-3570 CPU @ 3.40GHz`, operating memory `16.00 GB RAM` and operating system 64 bit Windows 10.

3.2 Timeline

In the summer 2017 we have repeated the measurement experiment in labs I-H3, I-H6 and F1-248 of the Faculty of Mathematics, Physics and Informatics of Comenius University Bratislava with allegedly fixed software application. It consumed countless amount of electric power and took a long time. With finally allegedly correct input data for GDD agreement comparison we have implemented GDD agreement comparison feature in our software during winter semester 2017 and consequently we have evaluated the results. But the final comparison results were very unsatisfactory as we could have not differentiate the 3 groups of input functional brain networks by the comparison results with any reasonable statistics methods (we have examined arithmetic mean and standard deviation of GDD agreement values between graphs of same group or between graphs of different groups, etc. but nothing feasible would come out of it). But fortunately just in that time we have took a class for advanced *Numerical methods* with dear Mrs. professor Mgr. Jela Babušíková, PhD. where we have come to deeply learn about floating-point and rounding arithmetic and errors in computations with computers with

fixed amount of memory (which is essentially every known computer on this planet) and we were absolutely devastated about how many potential bugs there could be in our computations and that we could never be able to reveal the truth about whether the property of functional brain network to suffer from Alzheimer's disease is projected to graph structure.

And so it did not took us long (even it did) to rewrite the whole application to use *math/big.Float* package to be able to enumerate float numbers with arbitrary precision. It is needless to say that this modification has deteriorated code readability very much and it even has slowed down the application performance significantly as all float numbers (so all numbers) in the application are treated as text strings from this point.

What was even worse is that *math/big.Float* package does have not implemented numeric operation methods for counting and extracting the nth root of a positive real number and so we were forced to implement it ourselves as we wanted to upgrade our already existing application and not to combine too many different tools and software for many different things. After extensive and extremely tedious study and comprehension of the Shifting n^{th} root algorithm [Wik21] and consequent implementation of this new feature, our software was finally allegedly ready to solve the mystery of Alzheimer's disease graph structure projection in the winter 2017 / 2018.

In the summer semester 2018 we have begun the study of ORCA algorithm to be able to improve our software even more and we have created our first ORCA prototype counting only 4-node graphlets in python.

In the winter semester 2018 we have begun to be suspicious as we could still not differentiate 3 groups of graphs even after results evaluation with an arbitrary precision. After very deep review of our code we have revealed the

bug in official *math/big.Float* library when creating a number not from string but rather directly from number literal. Of course we have fixed this or rather have found an workaround for our purposes but we have unfortunately not reported this bug as we could not prove it by sharing our code repository, as our repository was moved from public github repository to private one [Pu2] on gitlab when transitioning from Bachelor's thesis to Master's thesis on the suggestion of our research group to protect the intellectual property of Comenius University Bratislava. Nevertheless we could still not differentiate 3 groups and while still being suspicious we have found and error of our software in counting orbit 55 in the winter 2018 / 2019. In the summer semester 2019 comparison of measured results from original ORCA software with results measured by our software confirmed fortunately an error to be present only in counting of orbit 55. But even after comparison orbits excluding orbit 55 we have not been able to differentiate 3 groups... and moreover after the implementation of module integrating original ORCA GDD measurements into GDD agreement comparison tool of our software we have still not been able to differentiate 3 groups.

This revelation of Alzheimer's disease graph structure projection absence mystery has been confirmed also by our colleague Mgr. Andrej Jursa, PhD. and his independent implementation and evaluation of results.

The bug in counting orbit 55 was solved in the winter semester 2019 by implementing (standalone) ORCA module *_diplomka/app/orca.go* in our software counting 4 and 5-node graphlets just like original ORCA software.

3.3 Source code description

Along all this process a lot of refactoring and fixing was done in our software and we even have tried our best to make it a proper golang package with *bin*, *pkg* and *src* folder structure. We describe briefly the content of source code located in *app/src/gdd* folder:

- There are many folders including input and output data of measurements. Some of them like *cmp_group_v1*, ..., *cmp_group_v7* might even contain short *README* file to clarify experiment iteration details (like set precision, etc.). Some folders may be empty, they exist just for app to not crash, and they might have not been used yet with current configuration setting, there are many modules and to use a module it have to be set i.e. called from *main()* function.
- *gdd.go* is the main package and includes *main()* function.
- *gdd_test.go* includes some basic unit tests for the software.
- *server.go*, *client.go*, *worker.go*, *message.go* includes fundamental functionality for distributed tasks system.
- stat_tasks.go includes some basic statistics logging for tasks.
- *desync_patch.go* was (as the name of a file suggests) written for one time use only. It has fixed and saved some data that were corrupted by some communication protocol error. Fortunately we could identify the error and saved counted data. It was used in the first experiments during Bachelor's thesis era.

- After heavy refactoring during Master's thesis era most of the functionality have been moved to *packages* that can be compiled and built and should be able to been included later as standalone packages (libraries) even without source code, nevertheless our application, of course, always has access to all source code so it can be rebuilt always when needed. Particular packages may include one or more files but for the abstract purposes we describe just briefly the purpose of whole package, all the particular files can be studied separately as they are included just like all the source code on the attachment USB flash drive:
  - Package *packages/a_u_x* includes some auxiliary generic helper functions used across all the app. Name *aux* is a keyword and thus was reserved.
  - Package *packages/agreement* includes functionality of GDD agreement.
  - Package *packages/concurrent_int* includes custom class (struct in go programming language) for work with numbers featuring processor multi cores and threads i.e. concurrency. Handling some basic locking and mutexes and processor clock time synchronization is needed...
  - Package *packages/config* is the most important package as it contains configuration for the whole app.
  - Packages *packages/graph*, *packages/graph_isomorpher*, *packages/vertex* includes basic functionality and classes for data representation of graphs.
  - Package *packages/orbit_distribution* manages graphlet degree distribution data representation.

- Package *packages/orca_verifier* is very important package. It enables comparison of ORCA gained GDD and our original GDD data. It was used for debugging our original software and for orbit 55 bug detection as well as for validation of our own ORCA implementation that it works correct like it would be expected.
- Package *packages/pca* contains some data normalization for principal component analysis (even this could not differentiate 3 groups).
- Package *packages/results* contains some measurement results data management code and code for writing results in JSON and YAML text files.
- Packages *packages/task* and *packages/task_blueprint* contains code for task management for worker's stations.
- And finally package *packages/vertices_interval* is a class for our divide and conquer count GDD task gimmick we used in our original distributed software.

And even we did not achieve faster implementation than original ORCA software after all, our software is still good enough and can be used to count graphlet degree distribution of graphs just as good as original ORCA can.

3.4 Input dataset

We did use combination of our ORCA module and original ORCA software to count and compare these artificial BA, ER, GEO graphs provided by [HD14b] to verify GDD agreement measure:

id	file name	|V|	|E|	model
0	ba_1k_100k.in	1000	100231	BA
1	ba_1k_10k.in	1000	9900	BA
2	ba_1k_150k.in	1000	150144	BA
3	ba_1k_15k.in	1000	14775	BA
4	ba_1k_20k.in	1000	19600	BA
5	ba_1k_25k.in	1000	25324	BA
6	ba_1k_2k.in	1000	1996	BA
7	ba_1k_30k.in	1000	30039	BA
8	ba_1k_40k.in	1000	40236	BA
9	ba_1k_4k.in	1000	3984	BA
10	ba_1k_50k.in	1000	50191	BA
11	ba_1k_6k.in	1000	5964	BA
12	ba_1k_8k.in	1000	7936	BA
13	er_1k_100k.in	1000	100000	ER
14	er_1k_10k.in	1000	10000	ER
15	er_1k_12k.in	1000	12000	ER
16	er_1k_14k.in	1000	14000	ER
17	er_1k_150k.in	1000	150000	ER
18	er_1k_200k.in	1000	200000	ER

Table 3.1: Table shows stats of compared BA, ER, GEO graphs. In table $|V|$ stands for vertex count, $|E|$ stands for edge count and *id* is important just for short label in comparison infographic displayed in the next chapter 4.

id	file name	|V|	|E|	model
19	er_1k_20k.in	1000	20000	ER
20	er_1k_25k.in	1000	25000	ER
21	er_1k_30k.in	1000	30000	ER
22	er_1k_35k.in	1000	35000	ER
23	er_1k_40k.in	1000	40000	ER
24	er_1k_4k.in	1000	4000	ER
25	er_1k_50k.in	1000	50000	ER
26	er_1k_60k.in	1000	60000	ER
27	er_1k_6k.in	1000	6000	ER
28	er_1k_8k.in	1000	8000	ER
29	geo1k_100k.in	1000	100000	GEO
30	geo1k_10k.in	1000	10000	GEO
31	geo1k_12k.in	1000	12000	GEO
32	geo1k_14k.in	1000	14000	GEO
33	geo1k_150k.in	1000	150000	GEO
34	geo1k_200k.in	1000	200000	GEO
35	geo1k_20k.in	1000	20000	GEO
36	geo1k_25k.in	1000	25000	GEO
37	geo1k_30k.in	1000	30000	GEO
38	geo1k_35k.in	1000	35000	GEO
39	geo1k_40k.in	1000	40000	GEO
40	geo1k_4k.in	1000	4000	GEO
41	geo1k_50k.in	1000	50000	GEO
42	geo1k_6k.in	1000	6000	GEO
43	geo1k_8k.in	1000	8000	GEO

Table 3.2: Table shows stats of compared BA, ER, GEO graphs. In table $|V|$ stands for vertex count, $|E|$ stands for edge count and *id* is important just for short label in comparison infographic displayed in the next chapter 4.

Tables 3.3, 3.4 display functional brain networks data input for 3 groups of participants with / without Alzheimer's disease measurement:

file name	|V|	|E|	maximum d(G)
awith_40_std_edges.txt	5677	17200	43
awout_15_std_edges.txt	5439	20276	111
awout_24_std_edges.txt	5924	22315	65
awith_12_std_edges.txt	5335	23059	121
awith_06_std_edges.txt	5575	26204	93
awout_19_std_edges.txt	5287	30043	183
awout_07_std_edges.txt	6681	30443	121
young_25_std_edges.txt	7292	42740	176
awith_35_std_edges.txt	7347	45094	95
awith_31_std_edges.txt	6270	50549	162
awith_36_std_edges.txt	5752	55169	135
awout_11_std_edges.txt	6725	59929	242
awith_02_std_edges.txt	7529	66494	184
awith_33_std_edges.txt	7969	71869	167
awith_37_std_edges.txt	6218	73880	202
young_20_std_edges.txt	7163	83431	261
awith_30_std_edges.txt	6528	85443	193
awout_38_std_edges.txt	6847	87145	321
young_22_std_edges.txt	6482	93436	366
young_34_std_edges.txt	7131	93643	335

Table 3.3: Table displays functional brain networks data input for 3 groups of participants with / without Alzheimer's disease measurement. In table $|V|$ stands for vertex count, $|E|$ stands for edge count and maximum d(G) is the maximum node degree present in the graph.

file name	|V|	|E|	maximum d(G)
young_41_std_edges.txt	6941	101885	328
young_29_std_edges.txt	6977	103778	292
awith_09_std_edges.txt	7558	102091	205
young_23_std_edges.txt	6836	106575	332
awout_27_std_edges.txt	6600	110564	323
young_21_std_edges.txt	6736	112851	378
young_16_std_edges.txt	7032	131887	422
young_17_std_edges.txt	7553	134565	412
awout_13_std_edges.txt	7416	139482	416
young_18_std_edges.txt	7478	151486	433
awout_14_std_edges.txt	6849	172969	468
young_32_std_edges.txt	7539	180380	409
awout_28_std_edges.txt	8151	182021	345
awout_04_std_edges.txt	8000	183940	375
young_26_std_edges.txt	8183	211232	519
young_39_std_edges.txt	8582	307112	504
awout_05_std_edges.txt	8382	319165	591
awout_10_std_edges.txt	8131	361948	648
awout_08_std_edges.txt	8044	395861	778
awith_01_std_edges.txt	8394	450042	651

Table 3.4: Table displays functional brain networks data input for 3 groups of participants with / without Alzheimer's disease measurement. In table $|V|$ stands for vertex count, $|E|$ stands for edge count and maximum d(G) is the maximum node degree present in the graph.

Chapter 4

Results

After days of processor's runtime we were very dissapointed to evaluate and show comparison results of these trivially distinguishable graphs from tables 3.1 and 3.2 in summary infographic comparison tables 4.1 and 4.2.

Measured GDD agreements of compared graphs are exported in JSON format file so they can be evaluated easily later by any software / script. Visual way is the most intuitive and compact and easy to comprehend way. Hence that, we have interpreted the result visually with simple HTML, CSS table written in REACT framework (of course also all this bonus source code is included on attached USB flash drive).

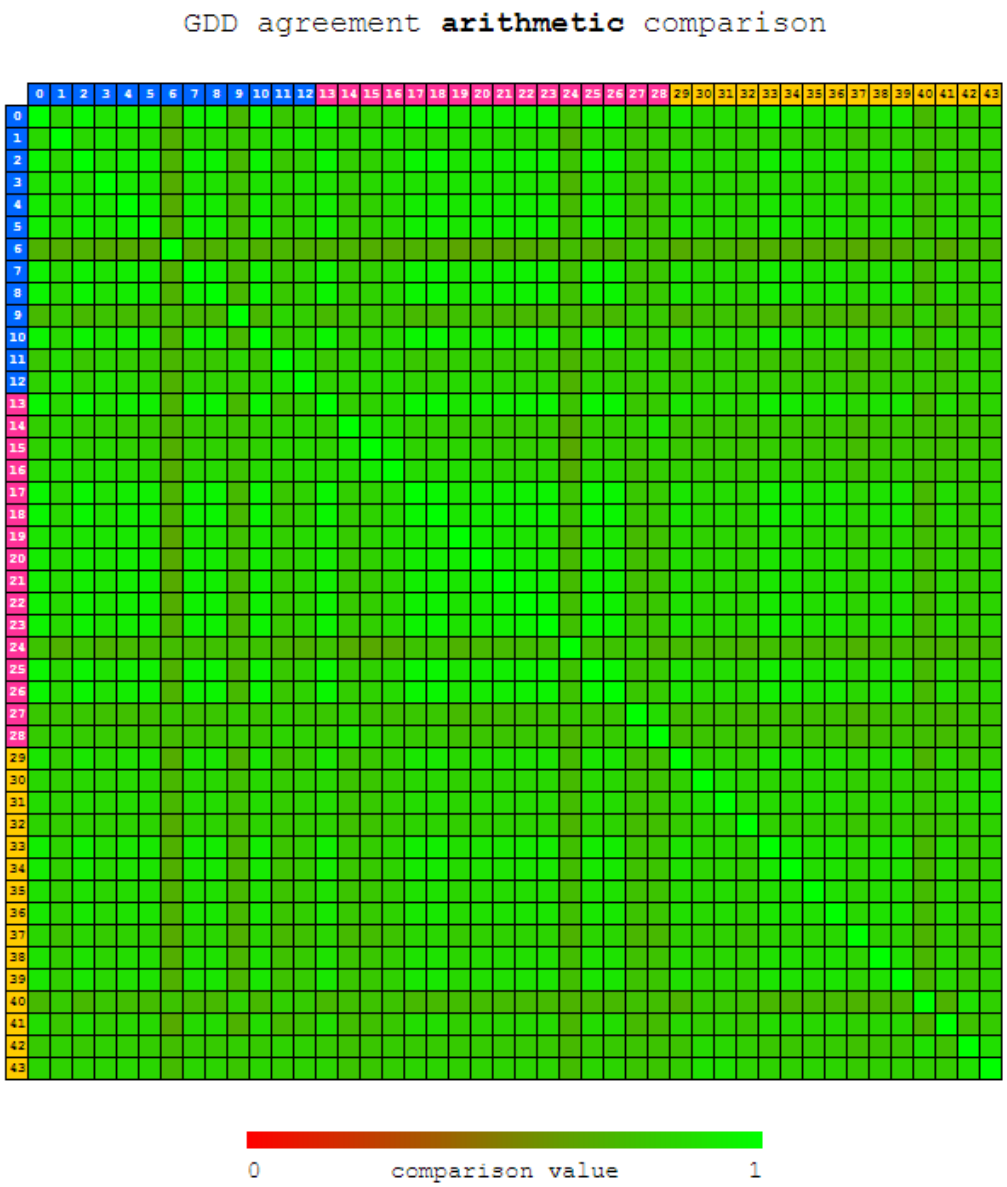


Figure 4.1: Comparison infographic table displays arithmetic GDD agreement between all graphs from tables 3.1 and 3.2. For convenient display only *id* labels of graphs are shown in head row / column. Classification of graph to models: Barabási–Albert (BA), Erdős–Rényi (ER), Random geometric (GEO) is illustrated by different colors of head row / column. The more similar are 2 compared graphs in terms of GDD agreement, the closer their GDD agreement comparison value is to 1 and this shows off as more green in infographic. Similarly, the less similar compared graphs are, the closer is their GDD agreement value to 0 and this shows off as more red in the table, i.e. x and y coordinates in the table defines compared graphs and the color of cell represents value of their GDD agreement.

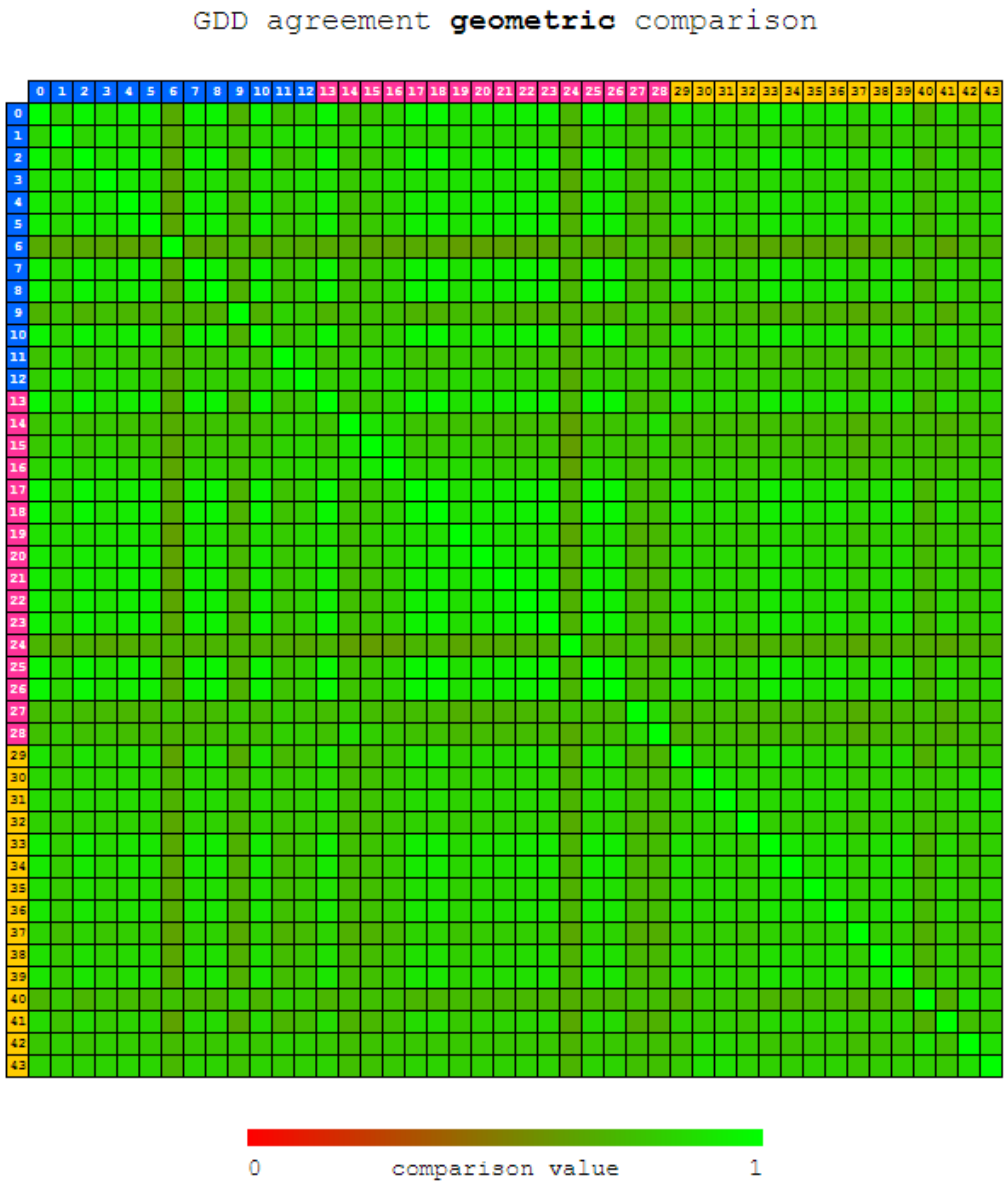


Figure 4.2: Comparison infographic table displays geometric GDD agreement between all graphs from tables 3.1 and 3.2. For convenient display only *id* labels of graphs are shown in head row / column. Classification of graph to models: Barabási–Albert (BA), Erdős–Rényi (ER), Random geometric (GEO) is illustrated by different colors of head row / column. The more similar are 2 compared graphs in terms of GDD agreement, the closer their GDD agreement comparison value is to 1 and this shows off as more green in infographic. Similarly, the less similar compared graphs are, the closer is their GDD agreement value to 0 and this shows off as more red in the table, i.e. x and y coordinates in the table defines compared graphs and the color of cell represents value of their GDD agreement.

We have analyzed many comparisons of GDD agreements of 3 groups of participants from tables 3.3 and 3.4. All source code and enumerations are included on attached USB flash drive. Final GDD enumerations (counted with ORCA algorithm) are included in *app/src/gdd/orca*. All final GDD agreement enumerations are included in *app/src/gdd/cmp_group_orca*. We have not compared GDD agreement of every graph with each other from this dataset. We have compared only graphs in groups of interest from this dataset. Names of files with graphs and result enumerations for 3 groups represent their classification to group:

- *awith* stands for elder people with diagnosed tendency for Alzheimer's disease.
- *awout* stands for elder people without Alzheimer's disease diagnosis.
- *young* stands for young people (obviously healthy people).

For instance we would expect graphs from similar groups like all graphs among one group or all graphs among union of similar groups (awout and young) to have similar GDD agreement, so values of either arithmetic or geometric GDD agreement between compared graphs should be close to 1. Whereas comparison of graphs between different groups (included amongst result files with *VERSUS* tag in name) should prove graphs are significantly less similar, or their GDD agreement is close to 0 in ideal situation.

Unfortunately we were not able to differentiate 3 groups of participants with GDD agreement measure. There was done vast amount of enumerations over many years with multiple iterations of software improvements as we stated in section 3.2.

For convenient display we show only *arithmetic mean* and *standard deviation* of measured GDD agreement values for groups of interest from this dataset in tables 4.1 and 4.2. But all data, results and source code can (and should) be examined from attached USB flash drive.

group of graphs	arithmetic mean	standard deviation
awith	0.958951	0.009460
awout	0.955163	0.010079
young	0.963415	0.008977
awout ∪ young	0.959025	0.009187
awith VERSUS awout	0.957246	0.010419
awith VERSUS young	0.959280	0.009668
awout VERSUS young	0.958780	0.007890

Table 4.1: Table displays stats for values of arithmetic GDD agreement for subset groups of interest of original input dataset from tables 3.3 and 3.4.

group of graphs	arithmetic mean	standard deviation
awith	0.958696	0.009565
awout	0.954893	0.010151
young	0.963208	0.009084
awout ∪ young	0.958783	0.009277
awith VERSUS awout	0.956986	0.010510
awith VERSUS young	0.959035	0.009771
awout VERSUS young	0.958535	0.007979

Table 4.2: Table displays stats for values of geometric GDD agreement for subset groups of interest of original input dataset from tables 3.3 and 3.4.

When comparing graphs with VERSUS tag: each graph from first group is compared with each graph from another group. Graphs of same group are not compared among themselves in VERSUS comparison.

We have fulfilled all goals of this thesis stated in section 3.1.

We patched and significantly improved our own distributed software solution featuring parallelism for graphlet degree distribution enumeration of arbitrary simple, connected, undirected graph. We implemented GDD agreement comparison module in our software (in package *app/src/gdd/packages/agreement*) as elaborated in section 3.3. Moreover, we have improved and rewritten our whole software to count with numbers with arbitrary precision to get most possible reliable results as stated in section 3.2.

We have implemented our own standalone ORCA module (*_ diplomka/app/orca.go*) based on original ORCA (combinatorial approach to graphlet counting) algorithm and software introduced by Tomaž Hočevar [HD14a], [HD14b].

We have measured and evaluated GDD agreements of 3 groups of participants divided by Alzheimer's disease diagnosis as summarized in tables 4.1 and 4.2.

We have measured and evaluated GDD agreements of trivially distinguishable ER, BA, GEO graphs as summarized in infographic comparison tables 4.1 and 4.2.

We carried out series of multiple experiments in computer science labs I-H3, I-H6 and F1-248 FMFI UK with functional brain networks of participants divided into 3 groups: young people, elder people with diagnosed tendency for Alzheimer's disease and elder people without this diagnosis. And later

we performed many enumerations by our home desktop PC.

We compared gained graphlet degree distribution agreements multiple times over multiple iterations with improving our software.

We compared even some trivially different networks to verify graphlet degree distribution agreement as a valid measure suitable for graph structure comparison.

Despite counting with arbitrary precision we were not able to distinguish 3 groups of participants or to distinguish groups of ER, BA, GEO graphs reliably.

Finally, we came to conclusion that GDD agreement is not reliable universally to compare graphs for their structure.

Chapter 5

Conclusion

It is clear from the figures 4.1 and 4.2 that GDD agreement can not differentiate these trivially distinguishable ER, BA and GEO graphs universally. We would expect the comparison figures resembling something like greenish squares on the diagonal and rest should be red so only graphs from similar groups should have high values of GDD agreement, yet unfortunately GDD agreement seems to be not reliable universally for graph comparison.

Even though our colleague Mgr. Andrej Jursa, PhD. has performed similar comparison experiments and has proved that GDD agreement can be used to differentiate trivially distinguishable graphs, unfortunately we were not able to reproduce demonstration of GDD agreement reliably differentiating distinct subsets of graphs.

We are terribly devastated by this conclusion but we do not recommend to use GDD agreement universally.

We can peacefully proclaim that we have fulfilled all goals of this thesis and definitely have finished our research.

Main additional contribution of our work is that we have researched deeply the possibilities and influence of arbitrary precision on GDD agreement results and even though it did not help to differentiate groups of graphs after all, if we would not try it, we could never know.

We have created vast amount of custom utilizable software for our research and even though it is not application with nice GUI (as it was never meant to be) for non computer literate user, every researcher with at least some basic computer skills may use our API or even extend it for custom purposes.

We have done everything we could to differentiate 3 groups of graphs (young people, elder people with diagnosed tendency for Alzheimer's disease and elder people without this diagnosis) by GDD agreement. But unfortunately GDD agreement is not able to reliably universally differentiate even ER, BA and GEO graphs.

We don't suggest examining 6 node graphlets for graphlet degree distribution comparison as future work, as we deem it the pure waste of time and energy.

Even our previous solution *the brute force, but as good as even possible algorithms* [Pu7] achieved surprisingly good performance results for specific graphs, we recommend using combinatorial approach to graphlet counting (ORCA) for graphlet degree distribution enumeration.

We assumne Alzheimer's disease is not projected to functional brain networks graph structure, as we were not able to distinguish graphs from different participants' groups by graphlet degree distribution agreement.

Finally, we came to conclusion that GDD agreement is not reliable universally to compare graphs for their structure.

GDD may be used to compare graphs only under very special strict circumstances as e.g. node and edge count of compared graphs may significantly influence GDD comparison results. Unfortunately such wide study is far beyond scope of our Master's thesis, but was elaborated and explained by our colleague Mgr. Andrej Jursa, PhD. in his Dissertation thesis Properties of Real and Artificial Complex Networks [Jur21].

Bibliography

[BC19] Anna D. Broido and Aaron Clauset. Scale-free networks are rare. *Nature Communications*, 10(1):1017, Mar 2019.

[Boh19] Martin Bohumel. *Individual Social Network Analysis (Master's thesis)*. Comenius University Bratislava, 2019.

[BSS+00] Randy L. Buckner, Abraham Z. Snyder, Amy L. Sanders, Marcus E. Raichle, and John C. Morris. Functional brain imaging of young, nondemented, and demented older adults. *Journal of Cognitive Neuroscience*, 12(supplement 2):24–34, 2000.

[Bud16] Lucia Budinská. *Graphlets in the Functional Brain Networks (Master's thesis)*. Comenius University Bratislava, 2016.

[Fer18] Bence Ferdinandy. What's the difference between a graph and a network?, 2018. `https://bence.ferdinandy.com/2018/05/27/whats-the-difference-between-a-graph-and-a-network/`, last accessed on 2020-12-30.

[Gri04] Ralph P. Grimaldi. *Discrete and Combinatorial Mathematics: an Applied Introduction*. Pearson Addison Wesley, Boston, 2004.

[GYA18] Jonathan Gross, Jay Yellen, and Mark Anderson. *Graph Theory and Its Applications*. 11 2018.

[HD14a] Tomaž Hočevar and Janez Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 12 2014.

[HD14b] Tomaž Hočevar and Janez Demšar. A combinatorial approach to graphlet counting, 2014. `https://file.biolab.si/biolab/supp/orca/`, last accessed on 2022-04-12.

[Jur21] Andrej Jursa. *Properties of Real and Artificial Complex Networks (Dissertation thesis)*. Comenius University Bratislava, 2021.

[LK14] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection, 2014. `https://snap.stanford.edu/data/`, last accessed on 2021-1-1.

[MBF14] Paul McCarthy, Lubica Benusková, and Elizabeth A. Franz. The age-related posterior-anterior shift as revealed by voxelwise analysis of functional brain networks. In *Front. Aging Neurosci.*, 2014.

[Ove20] Stack Overflow. What is the distinction between sparse and dense graphs?, 2020. `https://stackoverflow.com/questions/12599143/what-is-the-distinction-between-sparse-and-dense-graphs`, last accessed on 2022-05-03.

[Pr7] Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 01 2007.

[Pr0] Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 26(6):853–854, 03 2010.

[Pu7] Michal Puškel. *Probability Comparison of Functional Brain Networks (Bachelor's thesis)*. Comenius University Bratislava, 2017.

[Pu2] Michal Puškel. Gitlab repository gdd, 2022. `https://gitlab.com/michal.puskel/gdd`, last accessed on 2022-04-23.

[Sta11] Alexander Stanoyevitch. *Discrete Structures with Contemporary Applications*. Chapman Hall/CRC, 1st edition, 2011.

[Vys17] Marián Vyslúžil. *Graphlet in Complex Networks (Master's thesis)*. Comenius University Bratislava, 2017.

[Wes00] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2 edition, September 2000.

[Wik21] Wikipedia. Shifting n^{th} root algorithm, 2021. `https://en.wikipedia.org/wiki/Shifting_nth_root_algorithm`, last accessed on 2022-04-23.

List of Figures

Appendix

Attachment consists of USB flash drive containing all source codes as well as supplementary materials.