

# Time-memory trade-off in graphlet counting & Orbit vertex frequency vs. degree in the expected degree model

Miguel Santana de Freitas Amaral

Mestrado em Ciências de Computadores

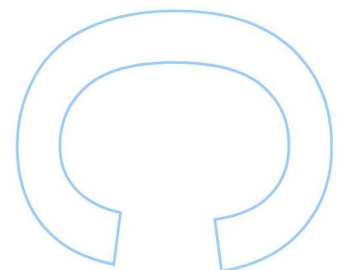
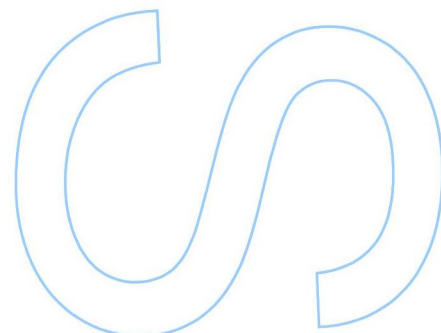
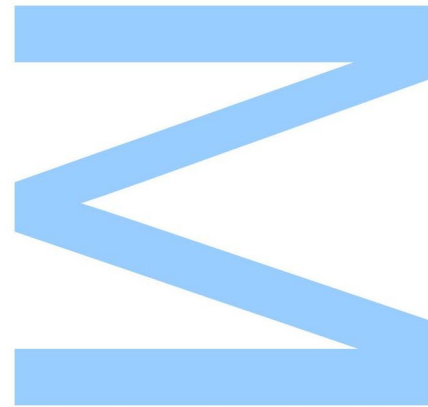
Departamento de Ciências de Computadores

2019

## Orientador

Pedro Manuel Pinto Ribeiro, Professor Auxiliar

Faculdade de Ciências da Universidade do Porto

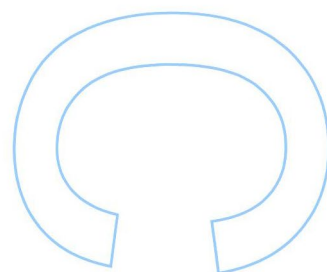
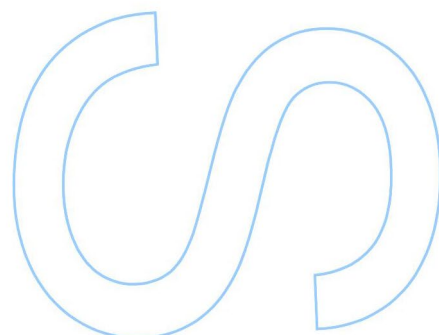
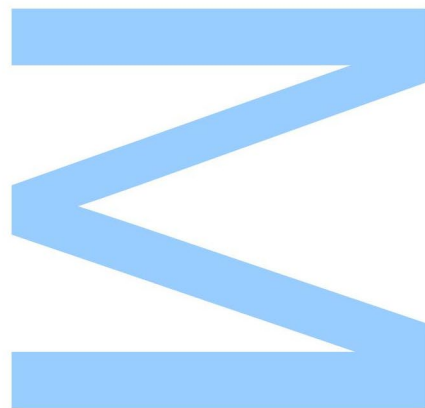




Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, \_\_\_\_ / \_\_\_\_ / \_\_\_\_



**Miguel Amaral**

Time-memory trade-off in graphlet counting &  
Orbit vertex frequency vs. degree in the expected  
degree model



Departamento de Ciência de Computadores  
Faculdade de Ciências da Universidade do Porto  
Setembro 2019

**Miguel Amaral**

Time-memory trade-off in graphlet counting &  
Orbit vertex frequency vs. degree in the expected  
degree model



*Tese submetida à Faculdade de Ciências da  
Universidade do Porto para obtenção do grau de Mestre  
em Ciência de Computadores*

**Supervisor:** Prof. Pedro Ribeiro

Departamento de Ciência de Computadores  
Faculdade de Ciências da Universidade do Porto  
Setembro 2019

Obrigado ao meu orientador, à minha família e aos meus amigos.



# Abstract

There are many important real world systems which are composed of interconnected entities: society, part of which are people and the relationships they form, chemical components and the reactions that they partake in, supply routes and their dependencies on each other, to give some examples. Studying a network of connections has proven sufficient to derive useful knowledge about the original system, and the microscale structure, i.e. patterns that appear in small groups of connected nodes, has correlated with the behavior of the entities these nodes represent. One type of microscale pattern that has been studied is the one of (induced or not) network motifs. These are small connected graphs (graphlets) that are present, if need be after a vertex relabeling (isomorphic to), as (induced or not) subgraphs in the network studied in an unexpected number versus an adequate null model.

This thesis presents two major contributions to this topic. First, we present FaSE-Jump, a modified FaSE algorithm for induced subgraph census that trades memory for time. Secondly, we derive analytical formulas for an extension to (induced) subgraph census that restrict the count via specification of both the topology of the vertex in the graphlet, via its orbit under graph isomorphism, as well as its degree in the full network, which we call graphlet-orbit degree frequency.

We developed and tested FaSE-Jump, a memory-sparing version of the subgraph census algorithm FaSE. This version places shortcuts on the data structure FaSE uses for indexing the graphlets it finds, an instance of a GTrie. These shortcuts allow FaSE-jump to jump from a node in the GTrie to a representative node in that level. This avoids building the subtrees under jump sources. FaSE-Jump was tested against FaSE on a wide range of networks. The results indicate that while memory use is sometimes reduced 3x in high memory use inputs, these are offset by a consistent 3x slowdown over the entire range of inputs. Future work might consider using a mixed strategy that places jumps at key depths of the GTrie.

The measure we present as graphlet-orbit degree frequency has, as parameters, two aspects of vertices: their topology in the graphlet, which can be interpreted as a microscale rule if statistically relevant; their degree, which often classifies their function in the network. After measuring this quantity, it is necessary to compare it with values obtained on a null-model. This can be done by Monte Carlo simulation or, more efficiently, with guaranteed accuracy, and furthering our understanding, by direct analytical calculation. We derived, and verified against Monte Carlo simulation, formulas for the mean value of graphlet-orbit degree frequency under the expected degree model.

*Keywords:* network science, network motif, subgraph census, FaSE, GTrie, graphlet, orbit, analytical, null-model, time-memory tradeoff, hidden variable model, expected degree model



# Resumo

Existem muitos sistemas reais que são compostos de entidades interconectadas: a sociedade, composta por pessoas e as relações que estas formam, compostos químicos e as reações em que tomam parte, rotas de abastecimento e as interdependências entre estas, para dar alguns exemplos. Estudar uma rede de conexões provou ser suficiente para derivar conhecimento útil sobre o sistema original, e a estrutura à microescala, isto é, os padrões que aparecem em pequenos grupos de nós conectados, tem-se correlacionado com o comportamento das entidades que estes nós representam. Um tipo de padrão à microescala que tem sido estudado é o de motivos (induzidos ou não) de rede. Isto são pequenos grafos conexos (grafetes) que estão presentes, se necessário após um renomear dos vértices (isomórficos a), como subgrafos (induzidos ou não) na rede estudada num número inesperado versus um modelo nulo adequado.

Esta tese apresenta duas contribuições principais para este tópico. Primeiro, apresentamos FaSE-Jump, uma versão modificada do algoritmo FaSE, de censo de subgrafos, em que a nossa versão troca memória por tempo. Segundo, derivamos fórmulas analíticas para uma extensão do censo de subgrafos induzidos que restringe a contagem via especificação não só da topologia do vértice no grafete, como também o seu grau na rede original, que chamamos frequência grafete-órbita grau.

Desenvolvemos e testamos FaSE-Jump, uma versão que poupa memória do algoritmo de censo de subgrafos FaSE. Esta versão coloca atalhos na estrutura de dados, uma GTrie, que FaSE usa para indexar os grafetes que encontra. Estes atalhos permitem ao FaSE-Jump saltar de um nó da GTrie para um nó representante no mesmo nível. Isto evita ter de construir as subárvores que descendem das origens dos saltos. FaSE-Jump foi testado contra FaSE numa gama variada de redes. Os resultados indicam que embora o uso de memória seja algumas vezes reduzido 3x com entradas de alto uso de memória, estes ganhos são balançados por um gasto consistente de 3x mais tempo de execução. Trabalho futuro poderá considerar usar uma estratégia que use saltos apenas a profundidades chave na GTrie.

A medida que apresentamos como frequência grafete-órbita grau tem, como parâmetros, dois aspectos dos vértices: a sua topologia no grafete, que pode ser interpretada como uma regra à microescala se estatisticamente relevante; e o seu grau, que frequentemente classifica a sua função na rede. Após medir esta quantidade, é necessário compará-la com os valores obtidos num modelo nulo. Isto pode ser feito por simulação Monte Carlo, ou, mais eficientemente, com precisão garantida e avançando o nosso conhecimento, directamente por

cálculo analítico. Derivamos, e verificamos contra simulações Monte Carlo, fórmulas para o valor médio da frequência grafete-órbita grau, no modelo do grau esperado.

*Palavras-chave:* ciência de redes, motivo de rede, censo de subgrafos, FaSE, GTrie, órbita, analítico, modelo nulo, troca tempo-memória, modelo de variável oculta, modelo de grau esperado

# Contents

<b>Acknowledgments</b>	<b>iii</b>
<b>Abstract</b>	<b>v</b>
<b>Contents</b>	<b>ix</b>
<b>List of Tables</b>	<b>xi</b>
<b>List of Figures</b>	<b>xii</b>
<b>List of Listings</b>	<b>xiii</b>
<b>Glossaries</b>	<b>xiv</b>
Glossary . . . . .	xiv
Acronyms . . . . .	xiv
Notation . . . . .	xiv
<b>1. Introduction</b>	<b>1</b>
1.1. Goals and Contributions . . . . .	3
1.2. Terminology and Definitions . . . . .	4
1.3. Organization . . . . .	4
<b>2. Time-Memory Trade-Off In FaSE Algorithm for Subgraph Census</b>	<b>5</b>
2.1. Introduction and Related Work . . . . .	5
2.2. Fast Subgraph Enumeration (FaSE) . . . . .	5
2.3. Adapting FaSE: saving memory . . . . .	9
2.4. Experimental Results . . . . .	11
2.5. Conclusions and future work . . . . .	15
<b>3. Analytical Approach to Orbit Counts</b>	<b>16</b>
3.1. Introduction and Related Work . . . . .	16
3.2. Hidden Variables Model as Graph Null Model . . . . .	18
3.3. Graphlets . . . . .	18
3.4. Orbits & Defining Graphlet-Orbit Vertex Frequency . . . . .	26
3.5. Graphlet-orbit $h$ -frequency $f_{\theta,h}$ . . . . .	27

*Contents*

3.6. Experimental Confirmation . . . . .	30
3.7. Refinements . . . . .	33
3.7.1. Correcting the high degree deviation . . . . .	33
3.7.2. Correcting the low degree deviation & extending the range beyond $w$ .	35
3.8. Conclusions and Future Work . . . . .	41
<b>Bibliography</b>	<b>43</b>
<b>A. Results Extended to All Graphlet-Orbits With <math>k \in \{3, 4, 5\}</math></b>	<b>47</b>

# List of Tables

2.1. Number of subgraphs ('occurrences'), number of isomorphism classes, number of leaf nodes in the <b>GTRIE</b> and the ratio between them, for a range of $k$ values for the network 'starwars'. Data from [10]. . . . .	6
2.2. Networks used as input for testing FASE variants. . . . .	11
2.3. Linear regression results in log-log space for runtime of new versions versus the runtime of the original implementation of FASE. $e^{\text{intercept}}$ is therefore the slowdown factor. . . . .	12
2.4. Memory data for the different FASE variations . . . . .	14
2.5. Runtime data for the different FASE variations . . . . .	15
3.1. $f_{\theta,d}$ for the example . . . . .	18
3.2. Example of $g, \sigma_c(g), I_c(g)$ for $g \in G_3^+$ . . . . .	19
3.3. Example of $c, I_g(c), \text{Aut}(c)$ for $c \in I_{k,c}$ . . . . .	19
3.4. $\{(g, \text{deg}(g)) : g \in G_k \wedge E(g) \supseteq E(c)\}$ and corresponding $M(c)$ . . . . .	25

# List of Figures

1.1.	In graph $G$ , the graphlet $c$ is isomorphic to subgraphs ( $a$ to $e$ ), of which only $d$ and $e$ are induced subgraphs. . . . .	2
1.2.	For each graphlet $g$ on $k = 3$ vertices, in cyclic notation, the permutation $\sigma_c(g)$ that creates them from their canonical form $c$ . On $c$ the vertices are annotated with their orbit $o$ . . . . .	3
2.2.	FASE diagram . . . . .	7
2.1.	Plot of example of FASE number of occurrences, classes and leaves versus $k$ . . . . .	7
2.3.	jump structure over FASE's . . . . .	9
2.4.	Runtime and maximum memory usage plots of new implementation of FASE and FASE-JUMP versus original FASE code. . . . .	12
2.5.	FASE vs. FASE-JUMP in number of nodes in the GTRIE. . . . .	13
2.6.	Number of nodes in GTRIE vs. memory for FASE. . . . .	13
3.1.	Example of two isomorphisms of graph $g$ , of which the bottom one is an automorphism. The isomorphism are the permutations written in cyclic notation under the $\sigma$ . . . . .	16
3.2.	$f_{\theta,v}$ and $f_\theta$ for the graph $G$ of Figure 1.1, and all graphlet-orbits $\theta$ for $k = 3$ . . . . .	17
3.3.	Diagram exemplifying $ind_{k,v}(G, k, v)$ and $o_{g,v}(g, v)$ . Grey nodes indicate both $g$ and, in a graphlet-orbit $(c, o)$ , the vertices in $o$ . . . . .	26
3.4.	Experimental orbit counts and hd approximation versus degree . . . . .	32
3.5.	Probability of existing a vertex with degree $d$ in a graph sampled from the expected degree model. . . . .	34
3.6.	High degree correction example . . . . .	35
3.7.	Diagram for Equation 3.42. Red edges are absent edges. . . . .	36
3.8.	Probability density $\rho$ corresponding to the integrand of the expression for $N_{\theta,d}$ , vs. $u$ in its full domain, for a representative discrete range of $d$ , and a $y$ -axis minimum of $10^{-7}$ . . . . .	37
3.9.	Leftmost numerator and denominator for Equation 3.41. . . . .	38
3.10.	$f_{\theta,d}(\theta, d)/P_{\exists,d}(d)$ and $f_{\theta,h}(\theta, d)$ versus $d \in \{t, \dots, n - 1\}$ . . . . .	39
3.11.	Low degree correction example. . . . .	39
3.12.	Experimental orbit counts and theoretical approximations versus degree . . . . .	40
A.1.	Results and theoretical approximations for all 72 graphlet-orbits with $k \in \{3, 4, 5\}$ . . . . .	47

# Listings

2.1. Subgraph enumerator specification . . . . .	8
2.2. FASE algorithm . . . . .	8
2.3. FASE-JUMP algorithm . . . . .	10

# Glossaries

## Glossary

**GTrie** data structure for storing and finding subgraphs. xi, 5, 6, 11

**FaSE-Jump** algorithm with the novel modification. 3, 9, 11, 15

## Acronyms

**FaSE** FAst Subgraph Enumeration. 3, 5, 6, 7, 11

**ESU** Enumerate Subgraphs. 5, 6

**WSL** Windows Subsystem for Linux. 11

**GDD** graphlet degree distribution. 16

**HVM** hidden variable model. 20, 27

## Notation

$f^\#$  when  $f : A \rightarrow B$ ,  $f^\#$  is a mapping that maps any subset of  $A$  to the multiset of the corresponding images  $\in B^\#$ .

$\#(m, e)$  the count of element  $e$  in multiset  $m$ .

$\{k_i : v_i\}_i$  **or**  $\{a : b, c : d, \dots\}$  compact notation for a map, equivalent to  $\{(k_i, v_i)\}_i$  or  $\{(a, b), (c, d), \dots\}$ .

$[n]$  the set of integers from 1 to  $n$ .

$\chi_v$  expression with value 1 if  $v$  is true or 0 if  $v$  is false.

$S_k$  the symmetric group on  $[k]$ .

$(abc, \dots, q)$  the permutation that maps  $a$  to  $b$ ,  $b$  to  $c$ , and so on, and  $q$  to  $a$ .

$\begin{pmatrix} a & b & \dots \\ \sigma(a) & \sigma(b) & \dots \end{pmatrix}$  the permutation  $\sigma$ .



$\text{deg}(v, g)$  degree of vertex  $v$  in graph  $g$ .

$\text{deg}(g)$  multiset of the degrees in graph  $g$ .

$\text{deg}^-(g, v)$  multiset of the degrees in  $g$  excluding the degree of  $v$ .

$k$  usual symbol for the number of vertices in graphlet.

$G_k$  set of graphs on vertices  $[k]$ .

$G_k^+$  set of connected graphs on vertices  $[k]$ .

$c$  usual symbol for canonical graph.

$\sigma_c$  map that takes a graph  $g$  and yields a permutation  $\sigma$  s.t  $g = \sigma c$  where  $c$  is  $g$ 's canonical form.

$o$  usual symbol for orbit.

$\theta$  graphlet-orbit, shorthand for a pair  $(c, o)$ .

$I_g(g)$  the isomorphism class of graph  $g$ .

$I_c(g)$  the canonical graph isomorphic to  $g$ .

$\text{Aut}(c)$  the automorphism group of graph  $c$ .

$I_k$  the set of isomorphism classes of graphs in  $G_k^+$ .

$I_{k,c}$  the set of canonical graphs in  $G_k^+$ .

$T_m(g)$  the probability that  $k$  arbitrary vertices from an implicit graph define a subgraph that is isomorphic to graphlet  $g$ .

$T_g(g)$  the probability that  $k$  arbitrary vertices from an implicit graph define an induced subgraph that is isomorphic to graphlet  $g$ .



# 1. Introduction

There are many important real world systems which are composed of interconnected entities. Representing the entities as nodes and the connections as edges in a graph results in what is called a complex network, so-called for their often non-trivial topology [1]. Studying this topology has proven sufficient to derive useful knowledge in many occasions, spawning the field of complex networks. A survey of applications can be found in [2], which covers systems such as society, part of which are people and the relationships they form; chemical components and the reactions that they partake in; economic supply routes and their dependencies on each other.

This field of study can be roughly split into the problems of representation, characterization and modelling. The problem of characterization consists in the creation of measures that can differentiate networks. The theoretical basis being the same, i.e. studying the topology of a network, regardless of application, any network measure can have multidisciplinary impact. A survey of such measures can be found in [3].

The focus of this thesis is one these measures, called network motifs [4]. They can be interpreted to be building blocks that by presence or absence characterize a network. Therefore, they can shed light on the first principles that rule the micro-scale structure of the network. Specifically, network motifs are small connected graphs (which we will call graphlets) that are present, if need be after a vertex relabeling (that is, isomorphic to), as subgraphs in the network studied at an unexpected rate versus an adequate null model, that is, their number needs to be statistically relevant. [Figure 1.1](#) illustrates the concept of subgraphs isomorphic to an example graphlet.

## 1. Introduction

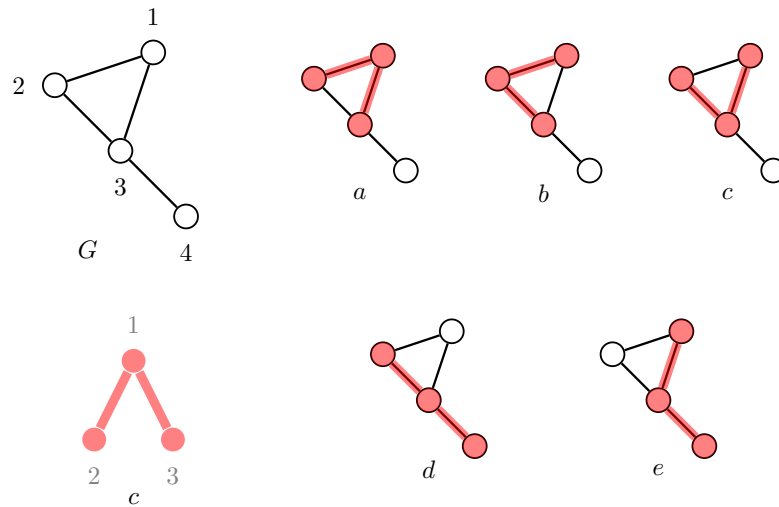


Figure 1.1.: In graph  $G$ , the graphlet  $c$  is isomorphic to subgraphs ( $a$  to  $e$ ), of which only  $d$  and  $e$  are induced subgraphs.

Network motifs have been successfully used in multiple areas, such as biology [5], sociology [6] and computer science [7]. However, they are computationally difficult, and therefore most work has been done with small sizes, with motifs containing at most 5 vertices. While it is possible to generalize the concepts in this work to directed graphs, this carries increased costs in performance and analysis, therefore we restrict ourselves to undirected graphs, i.e. edges have no direction.

Therefore, by definition, determining if a graphlet is a network motif implies two steps: performing a census over the network, which is an algorithmic problem named subgraph census, and determining if it is statistical relevant, which we will call the statistical relevance issue.

The subgraph census problem is related to the subgraph isomorphism problem, i.e., given two graphs  $G$  and  $H$ , determine if  $H$  is a subgraph of  $G$ , in the sense that it is a simplification that checks if the frequency of  $H$  is  $> 0$ . The subgraph isomorphism problem is NP-complete [8], making the subgraph census problem of at least the same difficulty.

Solving the statistical relevance issue can be achieved by computationally expensive generation and measurement of enough samples of the null-model to obtain statistics, or it can be done by analytical derivation of the expected measures.

A refinement on this concept is the concept of topologically equivalent vertices, in the context of a particular graphlet. This equivalence partitions the vertices into what are called orbits. A vertex in  $G$  then can be associated with all the graphlet-orbit pairs in which it appears. Subgraph counting algorithms can then collapse this information into statistics that relate graphlet-orbit appearances to vertex properties. This concept has been successfully applied in biology [9]. Figure 1.2 illustrates the concept of orbits.

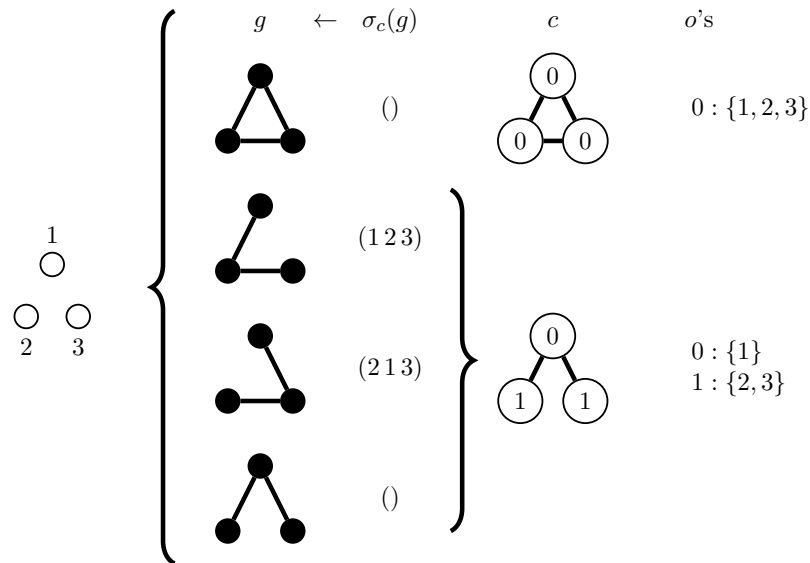


Figure 1.2.: For each graphlet  $g$  on  $k = 3$  vertices, in cyclic notation, the permutation  $\sigma_c(g)$  that creates them from their canonical form  $c$ . On  $c$  the vertices are annotated with their orbit  $o$ .

## 1.1. Goals and Contributions

Our intention with this thesis is to contribute in general to the subfield of network science concerning network motifs. We approached this by first delving into the subgraph census problem, and then by focusing on the statistical relevance issue via analytical methods.

Our approach to the subgraph census problem started by taking a state-of-the-art exact subgraph enumeration algorithm named [FAst Subgraph Enumeration \(FASE\)](#) [10] which main limitation is high memory usage. Our research goal was to find memory usage reduction strategies and then quantify their impact on runtime. We designed, implemented and tested a variant we called [FASE-JUMP](#) that reduces memory usage by as much as 3x, incurring a consistent 3x runtime increase.

As for the statistical relevance issue, we intended to derive new analytical formulas for statistical measures of network motifs, that would avoid time-consuming measures by random sampling. We focused on an apparently unexplored research space, that of analytical formulas for orbit counts aggregated by vertex degree on the expected degree model, the most common and real-world applicable null model. By extending pre-existent work on graphlet counts, we derived formulas for this novel measure which we successfully validated by systematic simulation for  $k \leq 5$ .

## 1. Introduction

### 1.2. Terminology and Definitions

We proceed by going over some terminology.

**Functions and symbols** In general, we use subscripts as fixed symbols to identify a specific function. For instance,  $f_a : A \rightarrow B$  would be a function from  $A$  to  $B$ , and  $f_a(a)$  would be the image of  $a$  by  $f_a$ .

**Edge representation** We represent an edge between vertices  $a$  and  $b$  as the set  $\{a, b\}$ , or the pair  $(a, b)$  or  $(b, a)$ , interchangeably, since we are dealing exclusively with undirected graphs.

**Subgraphs, induced or not** We proceed to clarify what we mean by subgraph. A graph  $g = (v, e)$ , with vertices  $v$  and edges  $e$ , is a subgraph of  $G = (V, E)$  when  $v$  is a subset of  $V$ , and the condition on  $e$  depends on whether or not the subgraph is induced. For a not induced subgraph,  $e$  can be any subset of the set of edges between vertices  $v$  that appear in  $G$ , while for an induced subgraph, it must include all of those edges. When performing measures on induced subgraphs, we will prefix those measures with the term 'induced'. [Figure 1.1](#) illustrates the concept of (induced) subgraphs.

**Graph isomorphism** We say two graphs are isomorphic when disregarding the labels assigned to their vertices makes them equal. Mathematically speaking, two graphs  $g = (v, e)$ ,  $g' = (v', e')$  are isomorphic when there is a bijection between its vertices such that their edges are the same, i.e. when  $\exists \sigma : v \rightarrow v'$  s.t.  $e' = \{\{\sigma a, \sigma b\} \mid \{a, b\} \in e\}$ .

### 1.3. Organization

The rest of the work is divided in two parts. First, the implementation and performance analysis of a modification to an existing subgraph census algorithm. Secondly, the derivation and confirmation by simulation of formulas for graphlet-orbit counts aggregated by vertex degree under the expected degree model.

## 2. Time-Memory Trade-Off In FaSE Algorithm for Subgraph Census

### 2.1. Introduction and Related Work

The problem of computing a subgraph census has been researched since Milo et al. [4] formulated network motifs and introduced the program MFINDER for calculating subgraph counts. Next came [Enumerate Subgraphs \(ESU\)](#) by Wernicke [11], which steps through each subgraph once while eschewing search paths that lead to overcounting. Each subgraph found needs to be mapped to its canonical graphlet. In general, the preferred tool for this classification task is the highly optimized program suite NAUTY[12]. By pre-matching partially constructed subgraphs into the same isomorphism class, improvements such as [FASE](#)[10] cut down on the number of classification calls. If the motif graphlet search space is limited, the GTRIE data structure [13] adapts to any subset of interest out of all possible  $k$  vertices graphlets, taking advantage of the corresponding symmetries to short-circuit the census.

These approaches are generalizable in graphlet size and, although less explored, in vertex and edge labelling. However, there is a distinction between algorithms that must run the census over the entire network to function, such as GTSCANNER [13], based directly on the GTRIE data structure, and algorithms that are more flexible in the sense that they can scan for subgraphs that intersect a given subset of vertices, such as [FASE](#). This last case, for instance, is more applicable to dynamic graphs, to situations where graph access is costly, or a global census is impractical. This motivates our research towards improving [FASE](#), that can be easily modified to achieve this query functionality. For a recent survey on subgraph counting see [14].

### 2.2. Fast Subgraph Enumeration (FaSE)

[FASE](#) is described by the pseudocode in [Listing 2.2](#) and illustrated in [Figure 2.2](#); in the following paragraph we will make an informal description of the pseudocode, enclosing literal quotes inside parenthesis. In our pseudo-code we use the notation  $a..b$  for the list  $a$  with  $b$  appended. [FASE](#) maintains a [GTRIE](#) [13], i.e. a trie of graphs (`gtrie`). Subgraphs are fed

## 2. Time-Memory Trade-Off In FASE Algorithm for Subgraph Census

	occurrences	classes	leaves	classes/leaves
k				
3	1.4e+03	2.0e+00	3.0e+00	6.7e-01
4	1.3e+04	6.0e+00	1.7e+01	3.5e-01
5	9.8e+04	2.1e+01	1.7e+02	1.2e-01
6	6.3e+05	1.1e+02	2.4e+03	4.4e-02
7	3.4e+06	7.0e+02	2.7e+04	2.6e-02
8	1.6e+07	5.6e+03	2.0e+05	2.7e-02
9	6.8e+07	4.2e+04	1.1e+06	3.7e-02

Table 2.1.: Number of subgraphs ('occurrences'), number of isomorphism classes, number of leaf nodes in the `GTRIE` and the ratio between them, for a range of  $k$  values for the network 'starwars'. Data from [10].

vertex by vertex (`n`) by a subgraph enumerator (`gen`) to the `GTRIE` resulting in a depth-first traversal. The subgraph enumerator, is in practice the `ESU` algorithm but in theory can be anything that respects the specification in [Listing 2.1](#). Once a leaf node is reached, i.e. once depth  $k$  is attained, the leaf's counter (`node.count`), initially zero, is incremented. After enumerating all the subgraphs of interest, each leaf is mapped by its corresponding graphlet (`graph`) to its canonical graphlet (`getCanonical(graph)`). In this way, `FASE` saves runtime by doing one classification for all subgraphs that matched a leaf. More specifically, at the  $d$ -th (`depth`) level of this tree, from each node, descend zero or more edges, each corresponding to a binary vector of size  $d$  (`makeLabel(n, subgraph)`). Each vector is the adjacency vector of the  $d + 1$ -th vertex (`n`) of some subgraph (`subgraph .. n`) that was scanned, and thus for any  $d$  each node of the tree at depth  $d$  can be made to correspond to a graph of size  $d$ .

Loosely speaking, the number of canonical graphs with  $k$  vertices grows in proportion to the number of graphs with  $k$  vertices, of which there are  $2^k$ . This happens since as  $k$  grows the symmetries that map several graphs to the same canonical form are easily broken by small changes. However, the number of leaves in the `GTRIE` that `FASE` constructs is larger still, since for any specific graph there are roughly  $k!$  ways of the algorithm arriving to it as it discovers it vertex by vertex. In practice, we ran `FASE` on an example network to get the magnitude of this combinatorial explosion, of the results are in [Table 2.1](#) and plotted in [Figure 2.1](#); for this example, the ratio of leaves to canonical graphs is around  $10^2$ . The situation worsens if considering any kind of combinatoric complication such as directed edges, labelled edges or vertices in general, or multilayer motifs, for instance, making memory usage a major limitation to the applicability of `FASE`.



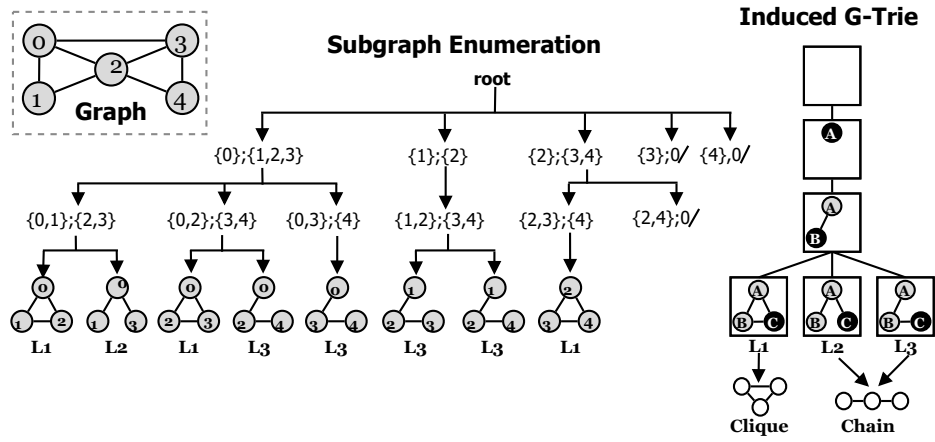


Figure 2.2.: FaSE diagram taken from [15].

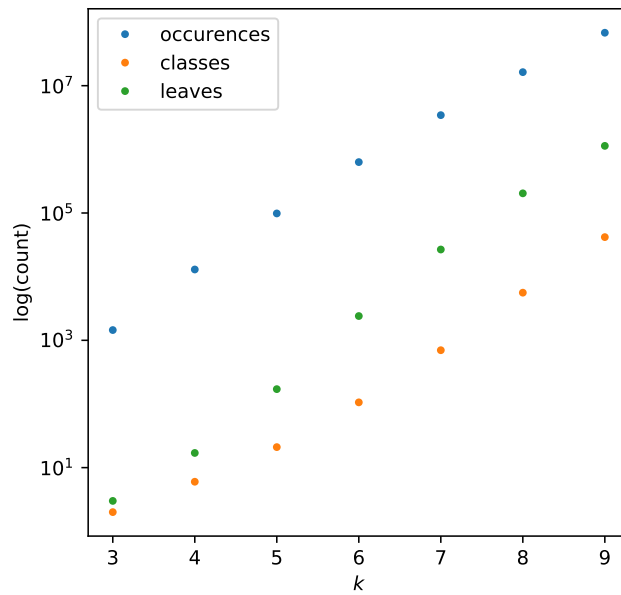


Figure 2.1.: Plot of FaSE number of occurrences, classes and leaves versus  $k$ , using data from Table 2.1.

## 2. Time-Memory Trade-Off In FASE Algorithm for Subgraph Census

### Listing 2.1: Subgraph enumerator specification

```
1 generates subset of induced subgraphs  $S$  of  $G$ , each identified by a set of vertices
2 enumerated so that the enumeration state  $s$  corresponds to a subgraph  $g =$ 
  ↪  $\text{subgraph}(s)$  of a member of  $S$ 
3 vertices of  $G$  must have an ordering
4 initial state is  $\emptyset$  so that  $\text{subgraph}(\emptyset) = \emptyset$ 
5 methods
6  $\text{next}(s:\text{state}) \rightarrow s':\text{state}, v:\text{vertex} \mid \text{null}$ 
7 goes over all valid vertices  $v$  s.t calling  $\text{super}(s,v)$  is valid
8  $\text{super}(s:\text{state},v:\text{vertex}) \rightarrow s':\text{state}$ 
9 generates a state so that:
10  $\text{subgraph}(s') = \text{subgraph}(s) \cup v$ 
```

### Listing 2.2: FASE algorithm

```
1 input
2 global gen : generator
3 global k : size of graphlets
4 output
5 map (canonical form of size k) -> count
6 code
7 global gtrie = empty Gtrie
8 recurse(
9   state =  $\emptyset$ ,
10  subgraph= empty list,
11  node = gtrie.root)
12 m = empty map with default value 0
13 for graph,count in gtrie.leaves_of_size(k):
14   m[getCanonical(graph)]+=count
15 return m
16
17 function recurse(
18   state:gen state,
19   subgraph : list of vertices,
20   node:gtrie node
21 )
22 depth = length(subgraph)
23 if depth == k:
24   node.count++
25 else:
26   while s',n  $\leftarrow$  gen.next(state)
27     recurse(
28       state = gen.super(s',n),
29       subgraph = subgraph .. n,
30       node = gtrie.child(node,makeLabel(n,subgraph))
31     )
```

## 2.3. Adapting FaSE: saving memory

We investigated the memory-savings obtained by, at each tree level, mapping on-demand all nodes that correspond to the same isomorphic graph to a representant node. The pseudocode in Listing 2.3 describes the version with the modification, which we named **FASE-JUMP**; through the following paragraph we will make an informal description of the pseudocode, enclosing literal quotes inside parenthesis. The algorithm is modified so that, after adding a new vertex ( $n$ ), it performs an horizontal jump from a node (`node_child`) to its representant (`node_target`). This eliminates the need for the subtrees under jump origins and thus reduces the number of nodes in the tree, thus saving memory. An example of this modification can be found in Figure 2.3. Table 2.1 shows the ratio between classes and leaves and gives an idea of the number of paths that can be avoided at each level. However, this modification requires additional bookkeeping. A traversal must maintain in memory a permutation (`permutation`) that transforms the partially constructed subgraph (`subgraph`) to its canonical form. At each node, corresponding to the mapping that brings it to its representant node, there must be a pointer (`node_child.target`) and a permutation (`node_child.permutation`). To initialize a new node with its jump target (`buildTarget`), after its canonical form  $c$  has been calculated we need to check whether or not there is already a representant for  $c$ . This is done via a map at each level from canonical graphlets to representant nodes (`depth_to_canonical_to_node[depth]`) All these additional structures requirements counter memory savings and increase runtime.

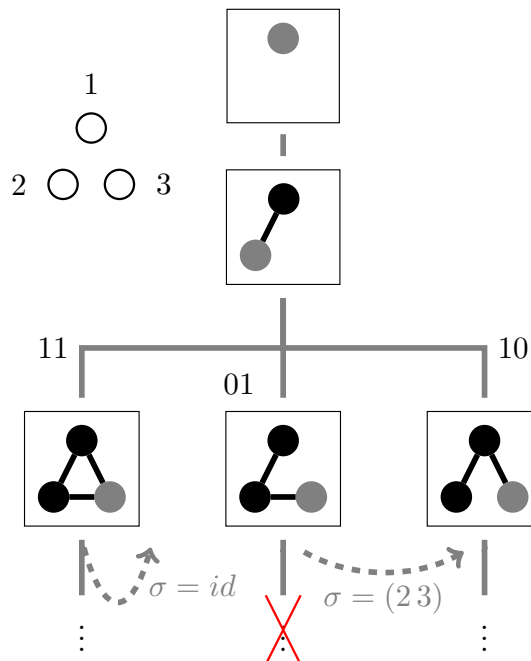


Figure 2.3.: In dashed lines, jump structure, over the data structure, in gray lines.

## 2. Time-Memory Trade-Off In FASE Algorithm for Subgraph Census

Listing 2.3: FASE-JUMP algorithm

```

1  input
2  global gen : generator
3  output
4  map ( size s → (map (canonical form of size s → count)) )
5  code
6  global gtrie = empty Gtrie
7  global depth_to_canonical_to_node = empty map with default value
8  ↪ (empty map)
9  recurse(
10     state = ∅,
11     subgraph= empty list,
12     node = gtrie.root,
13     permutation = empty list
14 )
15 m = empty map with default value (empty map)
16 for depth,m in depth_to_canonical_to_node.pairs():
17     for graph,node in m.pairs():
18         m[depth][graph]=node.count
19 return m
20
21 function recurse (
22     s:gen state,
23     subgraph : list of vertices,
24     node : gtrie node,
25     permutation : list of indices
26 )
27     depth = length(subgraph)
28     while s',n ← gen.next(s)
29         label = makeLabel(n,permute(permutation,subgraph))
30         node_child = gtrie.child(node,label)
31
32         if node_child.target is null:
33             buildTarget(depth+1,node_child,node,label)
34
35         node_target = node_child.target
36         node_target.count++
37
38         recurse(
39             state = gen.super(s',n),
40             subgraph = subgraph .. n,
41             node = node_target,
42             permutation = permute(
43                 node_child.permutation,
44                 permutation .. depth
45             )
46         )
47
48 function buildTarget(
49     depth : level of node_child,
50     node_child : gtrie node,
51     node : gtrie node parent of node_child,
52     label : adjacency vector
53 )
54     canonical_subgraph,canonical_to_input = getCanonical(
55         induced_subgraph(node.canonical,label)
56     )
57     node_child.permutation = canonical_to_input
58
59     node_target = depth_to_canonical_to_node[depth][canonical_subgraph]
60     if (node_target is null){
61         node_child.canonical = canonical_subgraph
62         depth_to_canonical_to_node[depth][canonical_subgraph] = node_child
63         node_target = node_child
64     }
65     node_child.target = node_target

```

## 2.4. Experimental Results

A pre-existent C++ code base implementing `FASE` and designed for LINUX was made cross-platform, and a PYTHON API was constructed to facilitate automated testing and metering. This resulted in three new versions of the software being tested, in addition to the original `FASE` implementation (referred as "original" in this text) run on the `Windows Subsystem for Linux (WSL)`, and used as reference: the cross-platform `FASE` version run on the `WSL`, the cross-platform `FASE-JUMP` version run on `WINDOWS`, and the cross-platform `FASE-JUMP` version run on the `WSL`. We tested the new versions against the same set of networks that `FASE` was tested against in its original paper [10]. Node and edge numbers can be found in [Table 2.2](#). For directed networks edge direction was ignored. The graphlets searched for were all with  $k$  vertices, for  $k$  starting at 3 and stopping at the largest value that took more than 20 seconds of runtime with the original implementation of `FASE`. All tests were conducted on the same machine, on a single-thread, with a CPU clocking at 3.8GHz and 16GB of RAM. For runtime analysis, data points for runs which took less than 15ms were discarded due to accuracy limitations.

Results are summarized visually in [Figure 2.4](#), for which the data is on [Table 2.5](#) and [Table 2.4](#). These verify qualitatively that our cross-platform version of `FASE` is as performant as the original, and therefore give us some assurance that the modifications that were necessary will not affect our evaluation of `FASE-JUMP`. The maximum memory usage plot, together with the numerical data, indicates that for inputs with original high memory usage, `FASE-JUMP` either did not increase the memory usage, or reduced it, by at most 3x. Regarding runtime values, the slow-down values are consistent enough to make a regression, which can be found in [Table 2.3](#). These reveal that `FASE-JUMP` is roughly 2.7x slower than `FASE`. We confirmed that `FASE-JUMP` reduced the number of nodes in the `GTRIE` compared to `FASE` (see [Figure 2.5](#)). It was also observed that the number of nodes in the `GTRIE` eventually directly correlates with the memory used by the original version of `FASE` (see [Figure 2.6](#)).

name	directed	#nodes	#edges
starwars	no	48	157
gloss	yes	67	118
jazz	no	198	2742
neural	yes	297	2148
email	yes	1133	5452
power	no	4941	6594
foldoc	yes	13356	91471
astroph	no	18772	198110

Table 2.2.: Networks used as input for testing `FASE` variants.

## 2. Time-Memory Trade-Off In FASE Algorithm for Subgraph Census

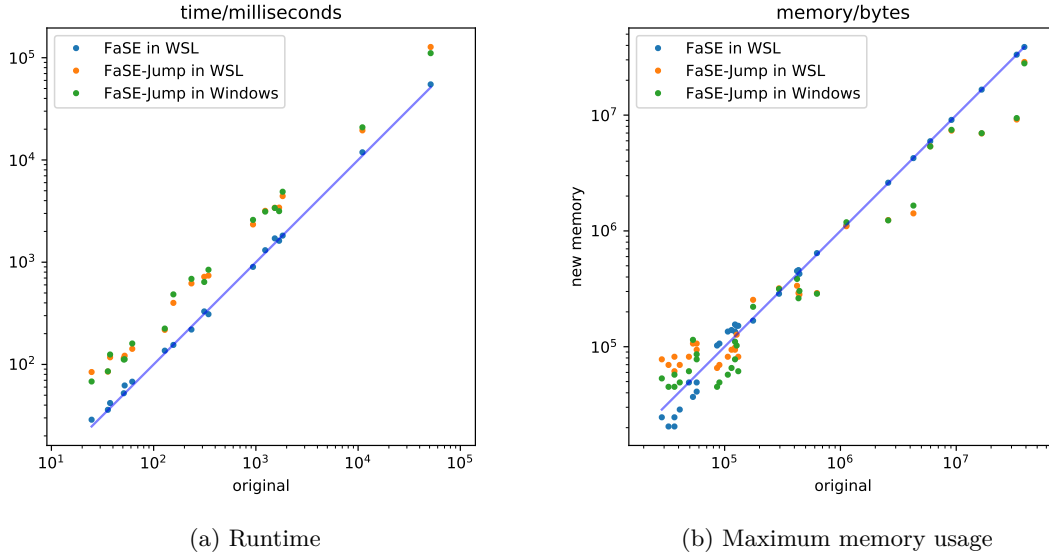


Figure 2.4.: Runtime and maximum memory usage plots of new implementation of FASE and FASE-JUMP versus original FASE code.

spec	slope	$e^{\text{intercept}}$
FaSE in WSL	0.99	1.09
FaSE-Jump in WSL	0.97	2.79
FaSE-Jump in Windows	0.97	2.86

Table 2.3.: Linear regression results in log-log space for runtime of new versions versus the runtime of the original implementation of FASE.  $e^{\text{intercept}}$  is therefore the slowdown factor.

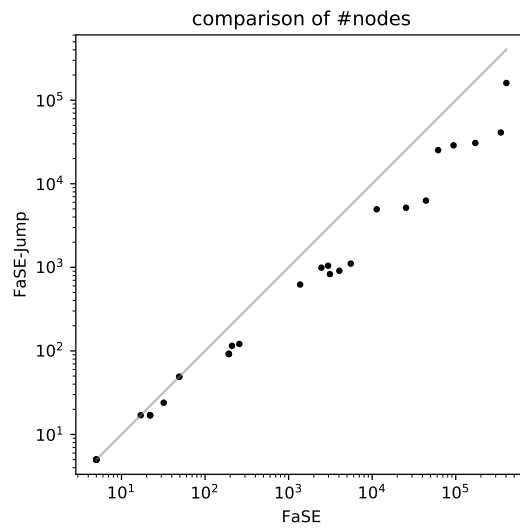


Figure 2.5.: FASE vs. FASE-JUMP in number of nodes in the GTRIE.

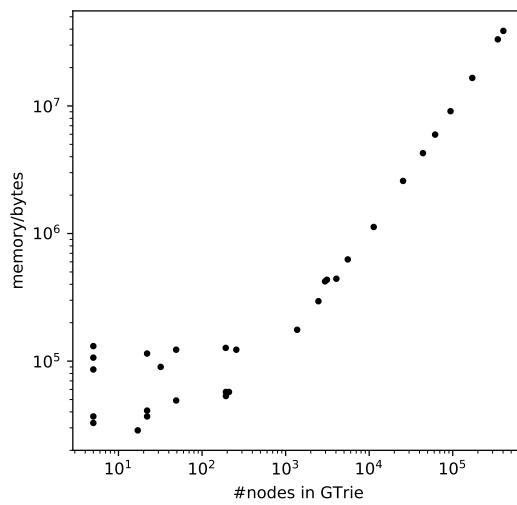


Figure 2.6.: Number of nodes in GTRIE vs. memory for FASE.

## 2. Time-Memory Trade-Off In FASE Algorithm for Subgraph Census

network	k	original/bytes	FaSE in WSL	FaSE-Jump in WSL	FaSE-Jump in Windows
gloss	3	2.9e+04	1.2e+00	3.7e-01	5.4e-01
starwars	3	3.3e+04	1.6e+00	4.7e-01	7.3e-01
jazz	3	3.7e+04	1.5e+00	6.0e-01	8.2e-01
starwars	4	3.7e+04	1.8e+00	4.5e-01	6.4e-01
jazz	4	4.1e+04	1.4e+00	5.9e-01	8.3e-01
neural	3	4.9e+04	1.0e+00	6.0e-01	8.0e-01
starwars	5	5.3e+04	1.4e+00	5.0e-01	4.6e-01
jazz	5	5.7e+04	1.4e+00	6.1e-01	7.4e-01
gloss	4	5.7e+04	1.2e+00	5.4e-01	6.7e-01
email	3	8.6e+04	8.4e-01	1.3e+00	1.9e+00
email	4	9.0e+04	8.5e-01	1.3e+00	1.8e+00
power	3	1.1e+05	7.9e-01	1.3e+00	1.9e+00
power	4	1.1e+05	8.2e-01	1.2e+00	1.8e+00
email	5	1.2e+05	9.1e-01	1.3e+00	1.1e+00
foldoc	3	1.2e+05	7.9e-01	1.3e+00	1.6e+00
power	5	1.3e+05	8.4e-01	1.0e+00	1.2e+00
astroph	3	1.3e+05	8.6e-01	1.6e+00	<b>2.1e+00</b>
gloss	5	1.8e+05	1.0e+00	6.9e-01	8.0e-01
neural	4	2.9e+05	1.0e+00	9.2e-01	9.4e-01
foldoc	4	4.2e+05	9.4e-01	1.3e+00	1.1e+00
power	6	4.3e+05	9.5e-01	1.5e+00	1.7e+00
starwars	6	4.4e+05	1.0e+00	1.6e+00	1.5e+00
email	6	6.3e+05	9.7e-01	<b>2.2e+00</b>	<b>2.2e+00</b>
gloss	6	1.1e+06	1.0e+00	1.0e+00	9.5e-01
power	7	2.6e+06	9.9e-01	<b>2.1e+00</b>	<b>2.1e+00</b>
starwars	7	4.3e+06	1.0e+00	<b>3.0e+00</b>	<b>2.6e+00</b>
gloss	7	6.0e+06	1.0e+00	1.1e+00	1.1e+00
neural	5	9.1e+06	1.0e+00	1.2e+00	1.2e+00
power	8	1.7e+07	1.0e+00	<b>2.4e+00</b>	<b>2.4e+00</b>
starwars	8	3.3e+07	1.0e+00	<b>3.6e+00</b>	<b>3.5e+00</b>
gloss	8	3.9e+07	1.0e+00	1.3e+00	1.4e+00

Table 2.4.: Maximum memory usage data points, in the form of memory gain factors for new versions, relative to the original version, which is in bytes. Entries are sorted by the original version values. Relative memory gains greater than 2 are bolded.



name	k	original/ms	FaSE in WSL	FaSE-Jump in WSL	FaSE-Jump in Windows
starwars	6	2.5e+01	1.2e+00	3.4e+00	2.8e+00
neural	4	3.6e+01	1.0e+00	2.4e+00	2.4e+00
gloss	7	3.8e+01	1.1e+00	3.1e+00	3.3e+00
jazz	4	5.1e+01	1.0e+00	2.3e+00	2.2e+00
email	4	5.2e+01	1.2e+00	2.3e+00	2.2e+00
power	6	6.2e+01	1.1e+00	2.3e+00	2.6e+00
foldoc	3	1.3e+02	1.1e+00	1.7e+00	1.7e+00
starwars	7	1.6e+02	9.9e-01	2.6e+00	3.1e+00
gloss	8	2.3e+02	9.4e-01	2.6e+00	2.9e+00
astroph	3	3.1e+02	1.1e+00	2.3e+00	2.0e+00
power	7	3.4e+02	9.0e-01	2.2e+00	2.5e+00
starwars	8	9.4e+02	9.6e-01	2.5e+00	2.8e+00
neural	5	1.2e+03	1.1e+00	2.6e+00	2.5e+00
email	5	1.5e+03	1.1e+00	2.2e+00	2.2e+00
jazz	5	1.7e+03	9.6e-01	2.0e+00	1.9e+00
power	8	1.8e+03	1.0e+00	2.4e+00	2.7e+00
foldoc	4	1.1e+04	1.1e+00	1.8e+00	1.9e+00
email	6	5.1e+04	1.1e+00	2.5e+00	2.2e+00

Table 2.5.: Runtime data points, in the form of slowdown values for new versions, relative to the original version, which is in milliseconds. Entries are sorted by the original version values.

## 2.5. Conclusions and future work

There is a memory improvement, but not always and not very significant with the inputs used. **FaSE-Jump** is always more expensive on time. Depending on runtime/memory constraints on a parallel version this might be advantageous. It might be interesting to experiment with a mixed approach that only uses the new strategy at key depths.

# 3. Analytical Approach to Orbit Counts

## 3.1. Introduction and Related Work

The concept of graphlets can be refined by considering orbits, i.e. the equivalence classes under automorphism of the vertices in a subgraph. Here, by automorphism we mean an isomorphism of a graph that does not change it: under such a transformation, vertices swap to topologically equivalent positions. An example of isomorphisms and an automorphism can be seen in [Figure 3.1](#). The concept of orbits has been previously illustrated in [Figure 1.2](#).

In this work we will call graphlet-orbit to a pair  $(g, o)$ , where  $g$  is a graphlet and  $o$  is a subset of vertices of  $g$  that form an orbit of  $g$ . We will often write  $\theta$  to represent a generic graphlet-orbit.

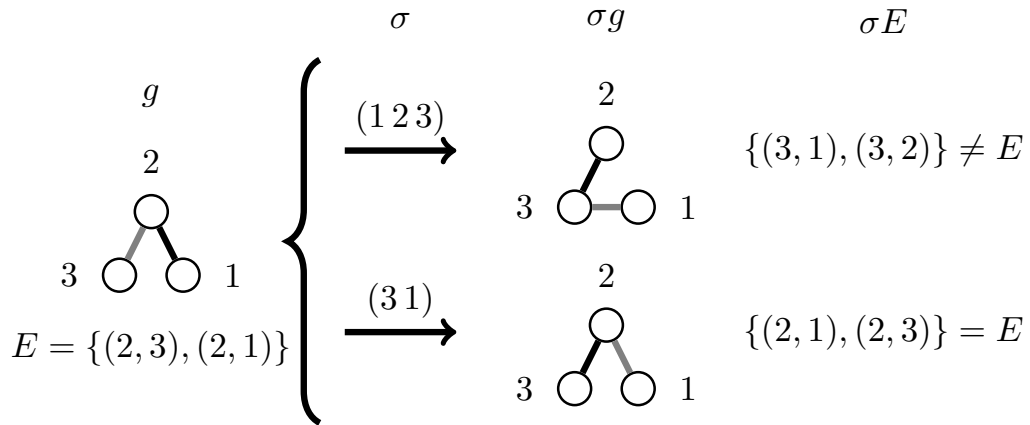


Figure 3.1.: Example of two isomorphisms of graph  $g$ , of which the bottom one is an automorphism. The isomorphisms are the permutations written in cyclic notation under the  $\sigma$ .

For a graphlet-orbit  $\theta = (g, o)$ , for each vertex  $v$  in  $V$ , one can consider what we will call the graphlet-orbit vertex frequency  $f_{\theta,v}(\theta, v)$  defined as the number of induced subgraphs  $g' \subseteq G$  isomorphic to  $g$  such that  $v$  maps to orbit  $o$ .

This measure can be further collapsed by dropping the dependency on  $v$  and instead having a count for each distinct frequency, obtaining what we will call the graphlet-orbit multiset, and is referred by others as the [graphlet degree distribution \(GDD\)](#) [16]:  $f_{\theta}(\theta) := \text{multiset}([f_{\theta,v}(\theta, v)]_{v \in V})$ .

To represent a multiset  $m$ , we will use a map, where each element  $v$  in  $m$  is mapped to its count  $\#(m, v)$ . The values corresponding to the graph in Figure 1.1 can be seen in Figure 3.2.

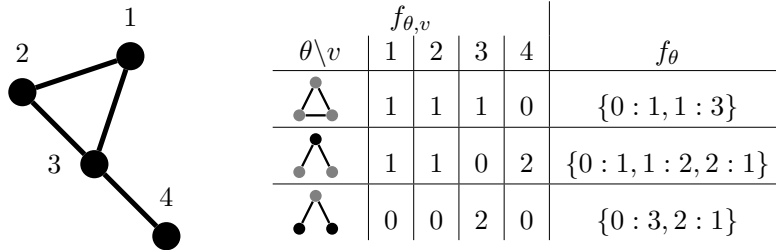


Figure 3.2.:  $f_{\theta, v}$  and  $f_{\theta}$  for the graph  $G$  of Figure 1.1, and all graphlet-orbits  $\theta$  for  $k = 3$ .

Computing these measures on data remains costly in both memory and time, despite substantial research and improvement [10, 13]. Therefore there is interest in obtaining analytical formulas for their statistical behavior under some null-model.

There is a decision in how much of the network being measured is to be inputted into the null-model and how much is to be randomized. The model choice also constrains the type of networks that can be fitted. An analysis of the asymptotic behavior of some of these measures in the preferential attachment model can be found in [17]. We focus our efforts on the expected degree model [18]. Our motivations are that it can conserve the degree distribution of the input, and is mathematically simple and suited to the ensuing calculations of moments of the measures being considered.

To our knowledge there are no exact formulas for the distributions of these measures. Formulas for the mean and variance for counts of (induced) subgraphs isomorphic to a graphlet in the expected degree model have been derived several times [19, 20, 21]. An analytical approximation to the real distribution that can be fitted using these two moments can be found in [22]. In this chapter, we make our own derivation of the formulas for the mean, and then proceed to our contribution, extending the calculation of the mean to graphlet-orbit counts aggregated by vertex degree. We will call this measure the graphlet-orbit degree frequency  $f_{\theta, d}$ . Defining:

$$V_d(d) := \{v \in V : \text{deg}(v) = d\} \quad (3.1)$$

We can then write out the formula for  $f_{\theta, d}$ :

$$f_{\theta, d}(\theta, d) = \begin{cases} \frac{\sum_{v \in V_d(d)} f_{\theta, v}(\theta, v)}{\#V_d(d)} & V_d(d) \neq \emptyset \\ 0 & V_d(d) = \emptyset \end{cases} \quad (3.2)$$

### 3. Analytical Approach to Orbit Counts

Inadvertently, although this formula allows us to have a defined value no matter the graph  $G$ , this confuses the cases where  $f_{\theta,v}$  is zero for all vertices with those where there are no vertices with degree  $d$ . As we will see, this will actually slightly complicate our analysis, but qualitatively appear to result in a smooth behavior versus degree.

Values for this quantity for the example in [Figure 1.1](#) can be checked at [Table 3.1](#).




$\theta \backslash d$	0	1	2	3	4
	0	0	1	1	0
	0	2	1	0	0
	0	0	0	2	0

Table 3.1.:  $f_{\theta,d}$  for the example

## 3.2. Hidden Variables Model as Graph Null Model

We start by using an hidden variables model [23] as null-model. Let the undirected random graph being generated be  $G = (V, E)$ , where the vertices  $V = \{v_1, v_2, \dots, v_n\}$ . Let  $D_G$  be the graph distribution. This model is sampled by drawing for each vertex  $v_i$  a hidden variable  $h_i$  from a distribution  $D_h$ . Let the hidden variables vector  $H = [h_1, h_2, \dots, h_n], h_i \sim D_h$ . Then for each possible edge  $(v_a, v_b)$ , create it with probability:

$$\begin{aligned} P((v_a, v_b) \in E) &= p_e((v_a, v_b), H) \\ &= p_{x,y}(h_a, h_b) \end{aligned} \tag{3.3}$$

Where  $p_{x,y}$  is a fixed, symmetric function with values in  $[0, 1]$ .

## 3.3. Graphlets

### Defining Graphlets

Define the transformation  $\sigma g$  of a graph  $g = (V, E)$  by a bijection  $\sigma : V(g) \rightarrow B$  as  $\sigma g = (B, \{(\sigma a, \sigma b) \forall (a, b) \in E\})$ .

Define graph isomorphism as the relation  $\sim: a \sim b \Leftrightarrow \exists$  bijection  $\sigma : V(a) \rightarrow V(b) : b = \sigma a$ .

Let  $G_k^+$  be the set of connected graphs on vertices  $[k]$ . Consider  $\sim$  restricted to  $G_k^+$ . For these graphs the isomorphisms are generated by applying  $S_k$ , the set of permutations of  $[k]$ , in the form of bijections from  $[k]$  to  $[k]$ . For a graph  $g \in G_k^+$  let  $I_g(g)$  be its isomorphism class and  $I_c(g)$  the corresponding representant, i.e. its canonical form. Then by definition there is  $\sigma_c : G_k^+ \rightarrow S_k$  s.t.  $g = \sigma_c(g)I_c(g)$ . For a canonical graph  $c$  with automorphism group  $Aut(c) := \{\sigma \in S_k : c = \sigma c\}$ , the orbit stabilizer theorem [24, Proposition 6.4 on p. 179] gives us the size of its isomorphism class:  $|I_g(c)| = \frac{k!}{|Aut(c)|}$ .

Let  $I_k$  be the set of isomorphism classes on  $G_k^+$ . Let  $I_{k,c}$  be the set of representants. These are the graphlets of size  $k$  of  $G$ .

Examples of these objects can be found in Table 3.2 and Table 3.3.

In practice we use NAUTY [12] to generate all connected graphs of size  $k$ , their canonical forms, and the size of the corresponding automorphism group.

$g$	$\sigma_c(g)$	$I_c(g)$
	$()$	
	$()$	
	$(213)$	
	$(123)$	

Table 3.2.: Example of  $g, \sigma_c(g), I_c(g)$  for  $g \in G_3^+$

$c$	$I_g(c)$	$Aut(c)$
	$\left\{ \begin{array}{c} \text{triangle graph} \\ \text{triangle graph} \end{array} \right\}$	$S_3$
	$\left\{ \begin{array}{c} \text{V-shaped graph} \\ \text{path graph of length 2} \\ \text{path graph of length 2} \end{array} \right\}$	$\{(23)\}$

Table 3.3.: Example of  $c, I_g(c), Aut(c)$  for  $c \in I_{k,c}$

### Mean graphlet count

We proceed to derive a formula for the mean number of occurrences of a graphlet  $c \in I_{k,c}$  with  $G \sim D_G$  drawn from the previously described null-model. While formulas for this measure have been previously derived, ours guided the implementation of a general evaluation routine and is the basis for the next section on orbit counts.

Consider the number of induced subgraphs isomorphic to a graphlet  $c$  in a graph  $G$ :

### 3. Analytical Approach to Orbit Counts

$$\#_c(G, c) := \sum_{s \in \binom{V}{k}} \chi_{\text{ind}(G, s) \sim c} \quad (3.4)$$

Where  $k = \#V(c)$ ,  $\binom{V}{k}$  is the set of subsets of  $V$  with size  $k$  and  $\text{ind}(G, s)$  is the subgraph of  $G$  induced by  $s$ . We use the notation  $\chi_v$  to mean 1 if  $v$  is true, and 0 otherwise.

Averaging over  $G \sim D_G$ , by linearity of expectation we obtain:

$$\langle \#_c(G, c) \rangle_G = \binom{n}{k} \cdot \langle \chi_{g \sim c} \rangle_g \quad (3.5)$$

Where  $g$  is a  $k$ -vertex graph on vertices  $[k]$  that follows the same [hidden variable model \(HVM\)](#). Expanding the average and splitting on the ways  $g$  can match  $c$ :

$$\langle \chi_{g \sim c} \rangle_g = \sum_{\substack{g \in G_k \\ g \sim c}} \langle p_g(g, H) \rangle_H \quad (3.6)$$

Where  $H = [h_1, \dots, h_k]$ ,  $p_H(H) = \prod_{i=1}^k p_h(h_i)$  and:

$$p_g(g, H) = \prod_{e \in E(C_k)} p_\chi(e \in E(g), x) \quad \left| \begin{array}{l} e = (i, j) \\ x = p_{x,y}(h_i, h_j) \end{array} \right. \quad (3.7)$$

And in this last formula,  $C_k$  is the complete graph on  $k$  vertices and:

$$p_\chi(b, x) := \begin{cases} x & b \\ 1 - x & -b \end{cases} \quad (3.8)$$

For non-induced subgraphs, indicated by a suffix  $m$ , the same formula applies, and  $p_g$  simplifies to:

$$p_m(g, H) = \prod_{(i,j) \in E(g)} p_{x,y}(h_i, h_j) \quad (3.9)$$

Let  $S_k$  be the set of permutations of  $[k]$ .

If  $g \sim c$  then  $\exists \sigma \in S_k : g = \sigma c$ .

In general we have:

$\forall \sigma \in S_k :$

$$\begin{aligned}
p_g(\sigma g, H) &= \prod_{(i,j) \in E(C_k)} p_\chi((i, j) \in E(\sigma g), p_{x,y}(h_i, h_j)) \\
&= \prod_{(\sigma^{-1}i, \sigma^{-1}j) \in E(g)} p_\chi((\sigma^{-1}i, \sigma^{-1}j) \in E(g), p_{x,y}(h_{\sigma\sigma^{-1}i}, h_{\sigma\sigma^{-1}j})) \\
&= p_g(g, \sigma^{-1}H) \quad (3.10)
\end{aligned}$$

So  $p_g(g, H) = p_g(\sigma c, H) = p_g(c, \sigma^{-1}H)$ .

For a continuous hidden variable distribution  $D_h$  with support  $[h_{min}, h_{max}]$ , we would have:

$$\langle X(H) \rangle_H = \int_{i=1}^k \int_{h_{min}}^{h_{max}} dh_i p_H(H) X(H) \quad (3.11)$$

For  $\sigma \in S_k$  let  $\sigma H := [h_{\sigma^{-1}i}]_i$ . The hidden variables are drawn independently so we have:

$$\forall \sigma \in S_k : \langle X(H) \rangle_H = \langle X(\sigma H) \rangle_H \quad (3.12)$$

Therefore:

$$\begin{aligned}
\langle \chi_{g \sim c} \rangle_g &= \sum_{\substack{g \in G_k \\ g \sim c \\ \sigma : g = \sigma c}} \langle p_g(c, \sigma^{-1}H) \rangle_H \\
&= \sum_{\substack{g \in G_k \\ g \sim c}} \langle p_g(c, H) \rangle_H \\
&= |I_g(c)| \langle p_g(c, H) \rangle_H
\end{aligned} \quad (3.13)$$

Let  $T_g(c)$  be the right factor  $\langle p_g(c, H) \rangle_H$ .

For a non-induced subgraphs, we get the same formula, changing the suffix  $g$  by  $m$ . Let  $T_m(c)$  be the right factor  $\langle p_m(c, H) \rangle_H$ .

Expanding the missing edges in  $p_g$ , we have:

### 3. Analytical Approach to Orbit Counts

$$\begin{aligned}
p_g(c, H) &= \sum_{\substack{g \in G_k : \\ E(g) \supseteq E(c)}} (-1)^{|E(g)/E(c)|} \prod_{e \in E(g)} p_e(e, H) \\
&= (-1)^{|E(c)|} \sum_{\substack{g \in G_k : \\ E(g) \supseteq E(c)}} (-1)^{|E(g)|} p_m(g, H)
\end{aligned} \tag{3.14}$$

Where  $G_k$  is the set of graphs on vertices  $[k]$ .

An example of the expansion of  $p_g$  can be seen in [Equation 3.15](#).

$$\begin{aligned}
p_g \left( \begin{array}{c} \bullet \\ \nearrow \quad \searrow \\ \bullet \end{array} \right) &= + p_m \left( \begin{array}{c} \bullet \\ \nearrow \quad \searrow \\ \bullet \end{array} \right) \\
&\quad - p_m \left( \begin{array}{c} \bullet \\ \nearrow \quad \leftarrow \quad \searrow \\ \bullet \end{array} \right) - p_m \left( \begin{array}{c} \bullet \\ \leftarrow \quad \searrow \\ \bullet \end{array} \right) \\
&\quad + p_m \left( \begin{array}{c} \bullet \\ \nearrow \quad \leftarrow \quad \searrow \\ \bullet \end{array} \right)
\end{aligned} \tag{3.15}$$

Therefore  $T_g$  can be written as signed sum of  $T_m$  terms:

$$T_g(c) = (-1)^{|E(c)|} \sum_{\substack{g \in G_k : \\ E(g) \supseteq E(c)}} (-1)^{|E(g)|} T_m(g) \tag{3.16}$$

If  $p_{x,y}$  is separable in  $x, y$ , then  $p_g$  can be separated in a factor per hidden variable, making  $T_m$  a product of the expected values of these factors. Otherwise, as a last resort it can be estimated by Monte Carlo methods, or variational methods [21] might be employed to determine the region of probability space that dominates the expectation being calculated in  $T_m$ .

**Picking  $p_{x,y}$**  We make the common choice of setting:

$$p_{x,y}(x, y) = x.y/w^2 \tag{3.17}$$

Where  $w$  is a constant so that  $w \geq h_{max}$ . This is possible for all input distributions with finite support, and is separable in  $x, y$ .

**Using the expected degree model** By using an intended degree distribution  $D_d$  as  $D_h$ , and setting  $w = \sqrt{\langle h \rangle n}$ , as long as  $\max(h) \leq \sqrt{\langle h \rangle n}$ , we obtain the expected degree model,



named so in [18] since, in the limit of large  $n$ , we get:

$$\langle \text{deg}(v_i) \rangle = \sum_{j=1}^n \frac{h_i h_j}{\langle h \rangle n} = h_i \frac{\frac{1}{n} \sum_{j=1}^n h_j}{\langle h \rangle} \approx h_i \quad (3.18)$$

However, this forces  $h_{\max} \leq w$ .

**Fitting an arbitrary distribution to the expected degree model** One can ensure  $\max(h) \leq \sqrt{\langle h \rangle n}$  by truncating and re-normalizing the hidden variable distribution, restricting the analysis to the vertices with degree  $\leq w$ . We will call the resulting distribution  $D^c(D_h)$ . It has probability density  $\rho(x) = \frac{\rho(h=x|h \sim D_h)}{P(h \leq w|h \sim D_h)}$ , where  $w = \sqrt{\langle h \rangle_{h \sim D^c(D_h)} n}$ . Its support is  $\{h \in \text{sup}(D_h) : h \leq w\}$ . Sampling the graph distribution now means generating a graph not with  $n$  vertices but with  $nP(h \leq w|h \sim D)$  vertices. In order to include the most out of the original distribution  $D_h$ , we want to maximize  $w$ . This requires solving an implicit equation, which in practice we solved numerically.

**Developing  $T_m$**  Using

$$p_{x,y}(x, y) = x.y/w^2 \quad (3.19)$$

, we get:

$$p_m(g, H) = \prod_{(i,j) \in E(g)} p_{x,y}(h_i, h_j) = \prod_{i=1}^k \left( \frac{h_i}{w} \right)^{\text{deg}(i,g)} \quad (3.20)$$

Where  $\text{deg}(i, g)$  is the degree of vertex  $i$  in graph  $g$ .

$$\begin{aligned} T_m(c) &= \langle p_g(c, H) \rangle_H \\ &= \prod_{i=1}^k \left\langle \left( \frac{h}{w} \right)^{\text{deg}(i,g)} \right\rangle_h \\ &= \prod_{i \in m} \left( \left\langle \left( \frac{h}{w} \right)^i \right\rangle_h \right)^{\#(m,i)} \\ &=: \Gamma(m) \end{aligned} \quad (3.21)$$

Where  $m = \text{deg}(g)$  is the multiset of the degrees of  $g$ , and  $\#(m, i)$  is the count of element  $i$  in multiset  $m$ . With this we can write the expected number of induced subgraphs as:

### 3. Analytical Approach to Orbit Counts

$$\langle \#_c(G, c) \rangle = \binom{n}{k} |I_g(c)| (-1)^{|E(c)|} \sum_{\substack{g \in G_k : \\ E(g) \supseteq E(c)}} (-1)^{|E(g)|} \Gamma(\deg(g)) \quad (3.22)$$

$$\Gamma(\deg(g)) = \prod_{i=1}^k \left\langle \left( \frac{h}{w} \right)^{\deg(i,g)} \right\rangle_h \quad (3.23)$$

We conclude this section by going over a practical concern when doing the summation in Equation 3.22, presenting and justifying our exploratory choice for the degree distribution  $D_h$  and studying the corresponding limits of  $T_m$  and  $T_g$  for large  $n$ .

**Gathering identical terms in  $T_g$**  Given a function  $f : A \rightarrow B$ , let  $f^\# : 2^A \rightarrow B^\#$  be the function that maps any subset of  $A$  to the multiset of the corresponding images, which we will call  $B^\#$ . In short, it turns a function into a counting function.

We use this formulation to express our gathering of identical terms in  $T_g$ . This gathering is a precaution against numerical inaccuracy. However, we could not avoid iterating over all the  $2^{k-|E(c)|}$  possible supergraphs in order to generate the multisets, which become a computational hurdle for larger  $k$ . The formula with gathered terms becomes:

$$T_g(c) = (-1)^{|E(c)|} \sum_{m \in M(c)} \#(M(c), m) (-1)^{E_{\#,m}(m)} \Gamma(m) \quad (3.24)$$

Where:

$$M(c) := (g \rightarrow \deg(g))^\# (\{g \in G_k : E(g) \supseteq E(c)\}) \quad (3.25)$$

$$E_{\#,m}(m) := \frac{\sum_{d \in m} d \cdot \#(m, d)}{2} \quad (3.26)$$

An example of calculating  $M(c)$  can be found in Table 3.4.





$g$	$deg(g)$	
$c =$ 	$\{1 : 1, 2 : 2, 3 : 1\}$	$\rightarrow M(c) = \left\{ \begin{array}{l} \{1 : 1, 2 : 2, 3 : 1\} : 1 \\ \{2 : 2, 3 : 2\} : 2 \\ \{3 : 4\} : 1 \end{array} \right\}$
	$\{2 : 2, 3 : 2\}$	
	$\{2 : 2, 3 : 2\}$	
	$\{3 : 4\}$	

Table 3.4.:  $\{(g, deg(g)) : g \in G_k \wedge E(g) \supseteq E(c)\}$  and corresponding  $M(c)$ 

**$\Gamma(m)$  in the limit of large  $n$  for power-law distributions** In order to deduce more specific results, we specialize our analysis to graphs with a power law distribution. We make this choice for its close connection to scale-free networks, which degree distributions asymptotically follow a power-law distribution. The discovery of scale-free networks in a myriad of real world systems catalyzed network science [25] and thus using a power-law as degree distribution allows us to work on well-established theoretical and practical ground.

This means we draw hidden variables from the truncated distribution  $D^c (D_\alpha^{pw})$ , where  $D_\alpha^{pw} : p(x) = (\alpha - 1)x^{-\alpha}, x \in [1, +\infty]$ , i.e. the power law distribution starting at 1 with parameter with  $\alpha \in ]2, 3[$ .

We have:

$$\begin{aligned}
M_h(d) &:= \left\langle \left( \frac{h}{w} \right)^d \right\rangle_h \\
&= \int_1^w \frac{\alpha - 1}{1 - w^{-(\alpha-1)}} h^{-\alpha} \left( \frac{h}{w} \right)^d dh \\
&= \frac{(\alpha - 1)w^{-d}}{1 - w^{-(\alpha-1)}} \int_1^w h^{d-\alpha} dh \\
&= \frac{(\alpha - 1)}{1 - w^{-(\alpha-1)}} \frac{(w^{-(\alpha-1)} - w^{-d})}{d - (\alpha - 1)}
\end{aligned} \tag{3.27}$$

In the limit of large  $n$ , i.e.  $n \rightarrow \infty \Leftrightarrow w \rightarrow \infty$ , the leftmost denominator goes to 1, and the rightmost numerator simplifies to  $w^{-\min(\alpha-1, d)}$ . Therefore, since  $\alpha \in ]2, 3[$ :

$$\Gamma(m) = (\alpha - 1)^k \left( \prod_{d \in m} \left( \frac{1}{d - (\alpha - 1)} \right)^{\#(m, d)} \right) w^{-(\#(m, 1) + (\alpha - 1)(k - \#(m, 1)))} \tag{3.28}$$

This means  $\Gamma(m)$  is proportional to  $\omega$  raised to some fixed power.

### 3. Analytical Approach to Orbit Counts

$T_g$  in the limit of large  $n$   $T_g$  is a sum of terms, one for each supergraph  $g'$  of  $g$ , each scaled by a factor of the form  $T_m(g') = \Gamma(\deg(g'))$ . Therefore, in the limit of large  $n$ , the dominant terms are those that do not add edges touching vertices with degree 1.

### 3.4. Orbits & Defining Graphlet-Orbit Vertex Frequency

The orbit  $o$  of a vertex  $v$ ,  $Orb(g, v)$  in a graph  $g$  is the right coset of  $Aut(g)$  with respect to  $v$ , i.e.  $Aut(g)v$ , where  $Aut(g)$  is the group of automorphisms of  $g$ . In general terms, an orbit contains vertices with the same topology in the graph. The set of orbits  $O(g)$  of a graph form a partition of its vertices. Orbits are conserved by isomorphisms in the sense that  $\sigma Orb(g, v) = Orb(\sigma g, \sigma v)$ .

Let  $O_k := \{(c, o) : c \in I_k, o \in O(c)\}$ , i.e., the set of the distinct graphlet-orbits on  $k$  vertices.

Consider the set  $ind_k(G, k)$  of all induced subgraphs of size  $k$  on a graph  $G$ , and for a vertex  $v$ , the subset of graphs that contain it,  $ind_{k,v}(G, k, v)$ .

For each subgraph  $g$ , get its canonical form  $c$ , and map the graph-orbits  $\subseteq O_k$  back to the vertices on  $G$ . In this fashion, each vertex  $v$  in  $V$  gets assigned an element  $o_{g,v}(g, v)$  in  $O_k$  per induced subgraph  $g$  that contains  $v$ . See Figure 3.3 for an example.

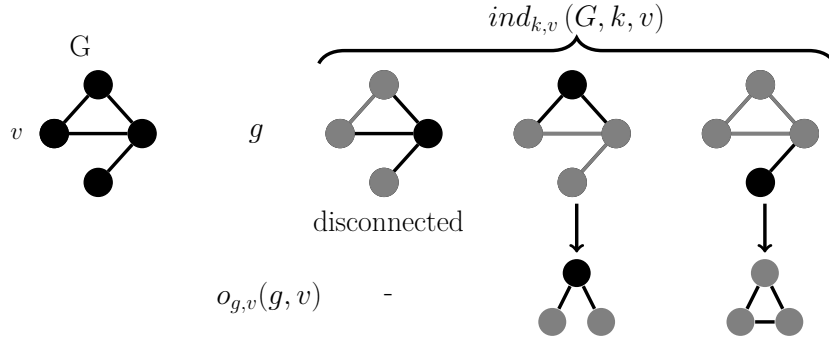


Figure 3.3.: Diagram exemplifying  $ind_{k,v}(G, k, v)$  and  $o_{g,v}(g, v)$ . Grey nodes indicate both  $g$  and, in a graphlet-orbit  $(c, o)$ , the vertices in  $o$ .

Let the multiset of these graphlet-orbits be:

$$O_{G,v,k}(G, v, k) = (g \rightarrow o_{g,v}(g, v))^{\#} (ind_{k,v}(G, k, v)) \quad (3.29)$$

This is the same as a function from graphlet-orbit  $\theta$  to graphlet-orbit vertex frequency  $f_{\theta,v}(\theta, v)$ , with the domain restricted to those  $\theta$  with at least one occurrence in  $G$ . For the example in Figure 3.3, we would have:

$$O_{G,k,v}(G, v, k) = \left\{ \begin{array}{l} \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array} : 1 \\ \begin{array}{c} \bullet \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \end{array} : 1 \end{array} \right\} \quad (3.30)$$

### 3.5. Graphlet-orbit $h$ -frequency $f_{\theta,h}$

We proceed to derive a formula for the mean graphlet-orbit vertex frequency, or a vertex  $v$  in  $G$ ,  $G$  drawn from the random model previously described, with a fixed hidden variable  $h$ . We name this  $f_{h,\theta}$  the graphlet-orbit  $h$ -frequency, where  $\theta$  is shorthand for the graphlet-orbit  $(c, o)$ :

$$f_{\theta,h}(\theta, h) := \langle \#(O_{G,v,k}(G, v, k), \theta) \rangle_G = \binom{n-1}{k-1} \langle \chi_{g \in G_\theta} \rangle_{g|h_1=h} \quad (3.31)$$

Where  $g$  is a random graph on vertices  $[k]$  that follows the same [HVM](#) as  $G$ , with the constraint that  $h_1$  is fixed and set to  $h$ , and  $G_\theta$  are the graphs that have vertex 1 in graphlet-orbit  $\theta$ :

$$G_\theta = \{g \in G_k^+ : g \sim c \wedge \sigma_c(g)^{-1}(1) \in o\} \quad (3.32)$$

By symmetry we can change vertex 1 in this definition to any other in  $[k]$ .

We will call  $f_{h,\theta}$  the graphlet-orbit  $h$ -frequency.

Let  $H^- := H|h_1 = h$ , that is, the distribution  $H$  with variable  $h_1$  fixed. Expanding the right factor of [Equation 3.31](#):

$$\langle \chi_{g' \in G_\theta} \rangle_{g'} = \sum_{g \in G_{k,\theta}} \langle p_g(g, H) \rangle_{H^-} \quad (3.33)$$

In general, letting  $S_k^-$  be the set of permutations on  $[k]$  that leave 1 unchanged, we have  $\forall \sigma \in S_k^- : \langle X(H) \rangle_{H^-} = \langle X(\sigma H) \rangle_{H^-}$ . For any  $g \in G_\theta$ ,  $\exists \sigma_g \in S_k^- : g = \sigma_g \gamma c$ , where  $\gamma$  is a fixed permutation, function of  $c, o$ , such that the vertex  $v_o := \gamma^{-1}(1)$  is in orbit  $o$  in graph  $c$ . Using this fact and [Equation 3.10](#), we obtain:

### 3. Analytical Approach to Orbit Counts

$$\begin{aligned}
\langle \chi_{g' \in G_\theta} \rangle_{g'} &= \left\langle \sum_{g \in G_\theta} p_g(g, H) \right\rangle_{H^-} \\
&= \sum_{g \in G_\theta} \langle p_g(\sigma_g \gamma c, H) \rangle_{H^-} \\
&= \sum_{g \in G_\theta} \langle p_g(\gamma c, \sigma_g^{-1} H) \rangle_{H^-} \\
&= \sum_{g \in G_\theta} \langle p_g(\gamma c, H) \rangle_{H^-} \\
&= |G_\theta| \langle p_g(\gamma c, H) \rangle_{H^-} \\
&= |G_\theta| \langle p_g(c, H) \rangle_{H|h_{v_o}=h}
\end{aligned} \tag{3.34}$$

We now address calculating  $|G_\theta|$ .

**Formula for  $|G_\theta|$  as a function of known quantities** Consider vertices  $[1, \dots, k] =: V$ , forming a graph  $g$ . We want the number of ways  $G_\theta$  of setting edges in  $g$  so that it is isomorphic to  $c$ , and such that vertex 1 is in orbit  $o$  ( $\theta = (c, o)$ ). The first condition we can rephrase as  $g \in I := I_g(c)$ . As for the second condition, by symmetry,  $|G_\theta|$  will be the same if we exchange 1 by any other of the  $k$  vertices. Formally, let  $A(v) := \{g \in I : ((\sigma_c(g))^{-1} v) \in o\}$ . Then by symmetry  $|A(v)| = |A(1)| \forall v \in V$ . We can therefore sum all occurrences of  $\theta$  over these  $k$  cases, which will be the number of vertices in orbit  $o$  summed over all  $g \in I$ , i.e.:

$$|G_\theta| * k = |I| * |o| \implies |G_\theta| = \frac{|I| |o|}{k}$$

The right factor  $\langle p_g(c, H) \rangle_{H|h_{v_o}=h}$  of Equation 3.34 can be simplified for non-induced subgraphs, by using Equation 3.20 and extracting the dependency on the fixed hidden variable  $h$ :

$$\begin{aligned}
\langle p_m(g, H) \rangle_{H|h_{v_o}=h} &= \left(\frac{h}{w}\right)^{\deg(v_o, g)} \prod_{i=v_o}^k \left\langle \left(\frac{h}{w}\right)^{\deg(i, g)} \right\rangle_h \\
&= \left(\frac{h}{w}\right)^{\deg(v_o, g)} \Gamma(\deg(g)/\{v_o\})
\end{aligned} \tag{3.35}$$

For a multiset  $m$  and a set  $s$ , by  $m/s$  we mean the multiset where all counts for the elements of  $s$  were dropped; therefore in Equation 3.35,  $\deg(g)/\{v\} = (v \rightarrow \deg(v, g))^{\#}(V(g)/\{v\})$ .

By using Equation 3.14 to expand the  $p_g$  factor in Equation 3.34, we can write Equation 3.31 in calculable terms as:

$$f_{\theta,h}((c, o), h) = \binom{n-1}{k-1} |G_{\theta}| (-1)^{|E(c)|} \sum_{\substack{g \in G_k : \\ E(g) \supseteq E(c)}} (-1)^{|E(g)|} \left(\frac{h}{w}\right)^{\deg(v_o, g)} \Gamma(\deg(g)/\{v_o\}) \quad (3.36)$$

We finish this section by first specifying how to aggregate identical terms when calculating the coefficients for the powers of  $h$  in Equation 3.36 for  $f_{\theta,h}$ .

**Gathering identical terms in the computation of coefficients of  $f_{\theta,h}$**  To prevent numerical errors, for the summands in Equation 3.36 with  $h$  raised to the same power  $d$ , that is, vertex  $v_o$  with the same degree  $d$ , with exact integer calculations we can count those with the same argument for  $\Gamma$ , which can be computed once and the result can be multiplied by the count. For given  $\theta = (c, o)$ , we define the multiset that stores those tallies as:

$$M(c, d, v) = (g \rightarrow \deg(g)/\{v\})^{\#} (\{g \in G_k : E(g) \supseteq E(c) \wedge \deg(v, g) = d\})$$

We can then write:

$$f_{\theta,h}(\theta, h) = \binom{n-1}{k-1} |G_{\theta}| (-1)^{|E(c)|} \sum_{d=1}^k \left(\frac{h}{w}\right)^d \sum_{m \in M(c, d, v_o)} \left( \#(M(c, d, v_o), m) (-1)^{E_{\#,m}(m)+d/2} \Gamma(m) \right) \quad (3.37)$$

### 3.6. Experimental Confirmation

We measured the average over  $D_G$  of  $f_{\theta,d}$ .

**Theoretical approximation “using  $h \approx d$ ”** A vertex’s assigned hidden variable  $h$ , determines the statistics of its degree; since it is the sum of  $n - 1$  independently drawn random variables, in general it will follow a normal distribution centered at its mean  $h$ , with a comparatively small variance. Reversing this relationship, a vertex with degree  $d$  will very likely have hidden variable  $h$  near  $d$ , and therefore for a graphlet-orbit  $\theta$  we can approximate the expected graphlet-orbit degree frequency for degree  $d$  by the graphlet-orbit  $h$ -frequency,  $f_{\theta,h}(d, \theta)$ . We will call this approximation “using  $h \approx d$ ”.

**Null model parameters** We used as hidden variable distribution  $D_h = D^c(D_{2.5}^{pw})$ , that is the truncated continuous power law distribution with minimum value 1 and parameter  $\alpha = 2.5$ . Solving for  $w$  resulted in  $w \approx 50$ . Choosing a power law makes it possible to have closed formulas for the moments of  $D_h$  and therefore closed formulas for the orbit counts.

**Sampling parameters**  $10^4$  graphs  $G$  generated, each with  $10^3$  vertices. These parameters were increased in tandem until the resulting measurements were qualitatively smooth.

**Graphlet-orbits measured** For practical reasons, we restricted ourselves to measuring the graphlet-orbit degree frequencies for the 72 graphlet-orbits with  $k \in \{3, 4, 5\}$ . This kept measuring runtime under 20 minutes and allowed us to visually inspect each graphlet-orbit result in a short time-frame.

**Implementation details** Coding was done in the PYTHON scientific stack. The hidden variable distribution was handled as an opaque input parameter, so that it can easily be swapped by arbitrary input. A version of a subgraph counting algorithm with graphlet-orbit counting capabilities based on the data structure was modified to compute graphlet-orbit vertex frequencies aggregated by degree. Generating and measuring the samples took in total  $\approx 1$ h.

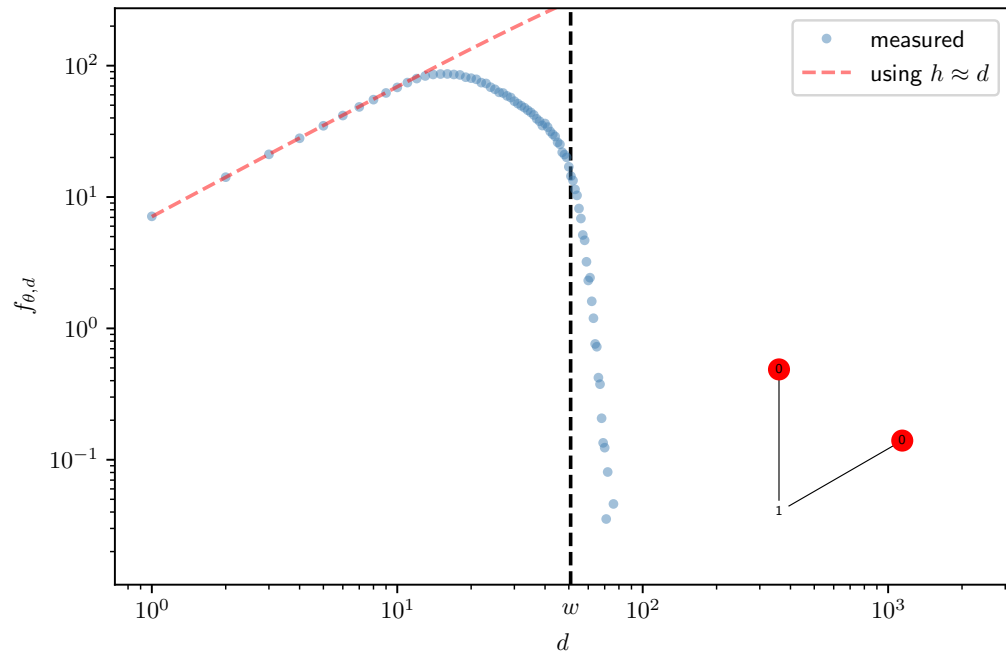
**Results** In [Figure 3.4](#) we show, versus degree, the experimental graphlet-orbit degree frequency  $f_{\theta,d}$  (see [Equation 3.2](#)), together with the “using  $h \approx d$ ” approximation for two graphlet-orbits. The same plots for all 72 graphlet-orbits with  $k \in \{3, 4, 5\}$  can be found in the appendix, [Figure A.1](#), plotted together with the better theoretical approximations described ahead.



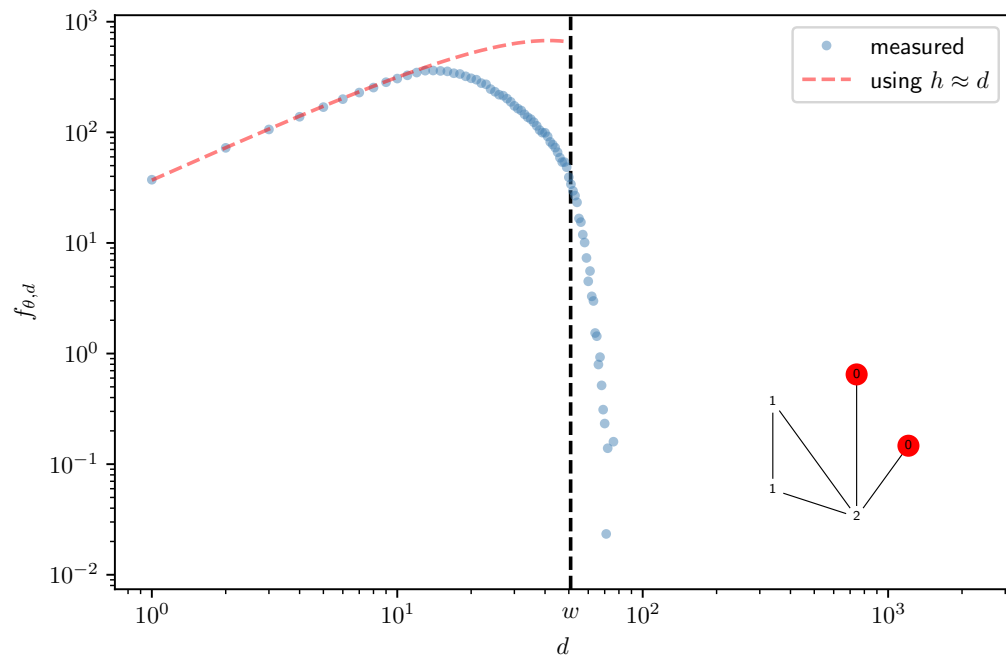
### 3.6. *Experimental Confirmation*

The plots show qualitative deviation from our approximation, for both low degree and close to  $w$ , after which our approximation is not defined while the results present values. We detail two cumulative improvements, that together match the results exactly, further ahead in [section 3.7](#). While their derivation is general to our model, these refinements were developed in response to this simulation.

### 3. Analytical Approach to Orbit Counts

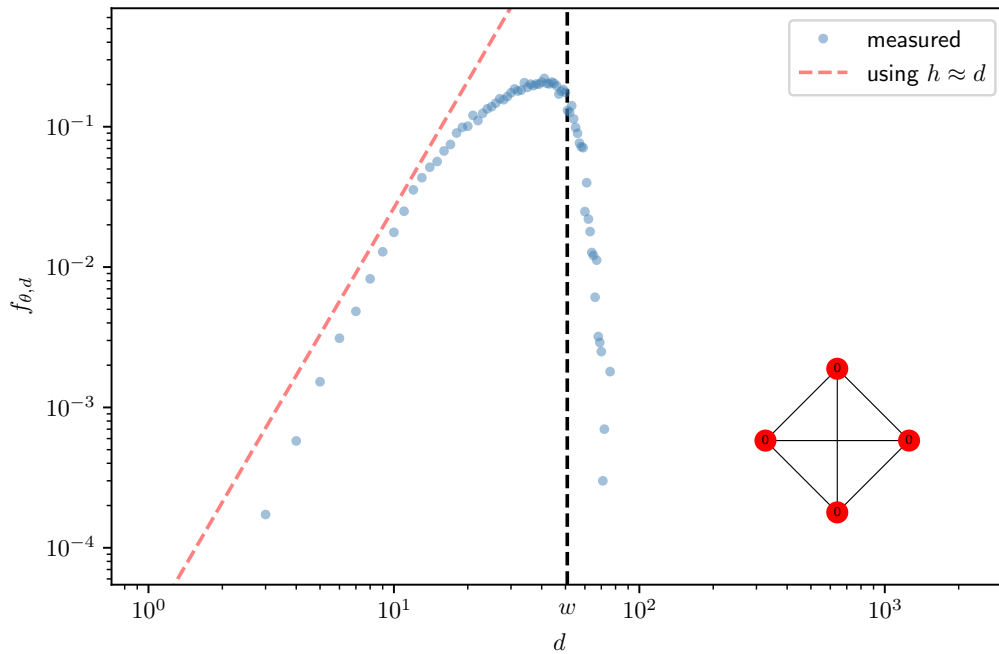


(a)



(b)

Figure 3.4.: Experimental orbit counts and  $h \approx d$  approximation versus degree. In the lower right, graphlet, with nodes labeled by orbit, and in red the orbit being counted.



(c)

Figure 3.4 continued

### 3.7. Refinements

Simulation results for the graphlet-orbit degree frequency showed a large deviation from  $f_{\theta,h}(\theta, d)$  for high degree, and a smaller one for low degree (see Figure 3.4). Additionally, the full range of measured degrees is outside the range of the hidden variable. In this section we correct these deviations in two steps.

#### 3.7.1. Correcting the high degree deviation

In a graph  $G$ , for a fixed degree  $d$  and graphlet-orbit  $\theta$ , the experimental graphlet-orbit degree frequency is the average orbit count over all vertices with degree  $d$  (see Equation 3.2). However, this specific sample  $G$  of  $D_G$  will not contribute to the average over  $D_G$  if there is not a single vertex with degree  $d$ .

The correction to be made is therefore a multiplication of  $f_{\theta,h}$  by:

$$P_{\exists,d}(d) := P(\exists v : \text{deg}(v, G) = d | G \sim D_G) \quad (3.38)$$

This can be approximated from the probability that a random vertex has degree  $d$ :

### 3. Analytical Approach to Orbit Counts

$$P_d(d) := \langle P(x = d | x \sim \text{Binomial}(n - 1, u \langle u \rangle)) \rangle_u \quad (3.39)$$

Where  $u := \frac{h}{w}$  and consequently  $u \langle u \rangle$  is the probability that a vertex with hidden variable  $u$  forms an edge with some other random vertex.

We can use  $P_d$  to calculate the probability that none of the  $n$  vertices has degree  $d$ , and negate that again to obtain:

$$P_{\exists,d}(d) \approx 1 - (1 - P_d(d))^n \quad (3.40)$$

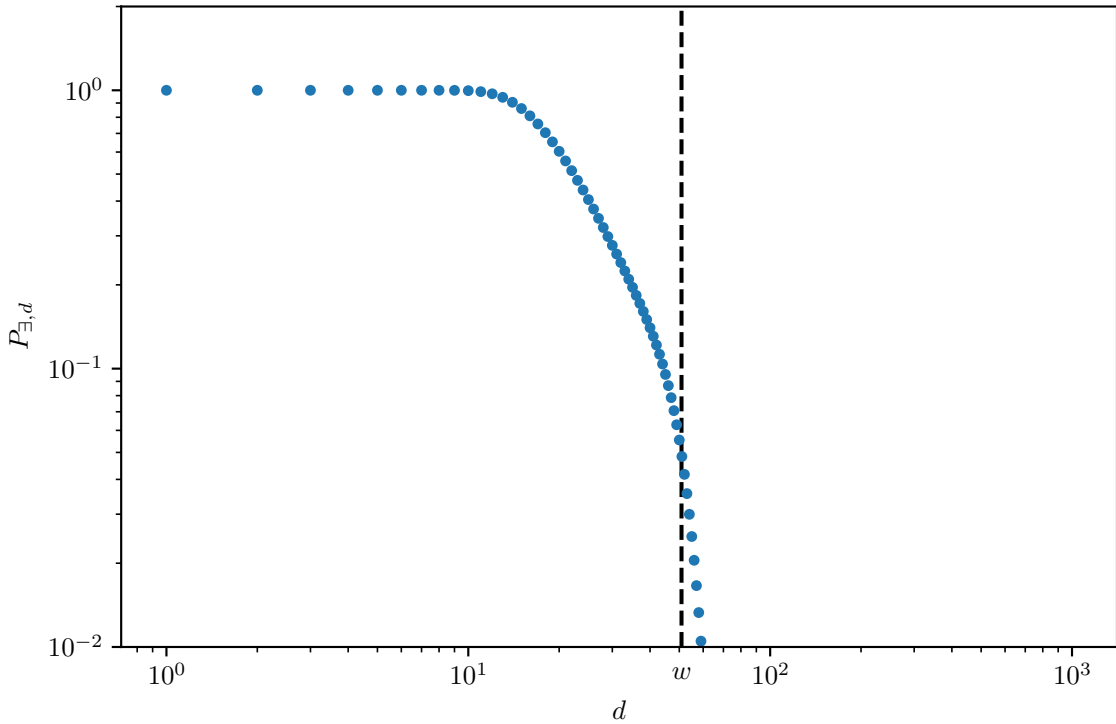


Figure 3.5.: Probability of existing a vertex with degree  $d$  in a graph sampled from the expected degree model.

Using the parameters of our experimental confirmation, as can be seen in [Figure 3.5](#),  $P_d$  is 1 for low  $h$  and then goes exponentially down when arriving near  $w$ , the limit we established for the value of the hidden variable.

As can be seen in [Figure 3.6](#), specifically the “using  $P_{\exists,d}$ ” line, this factor corrects the high degree deviation, but does not extend the range beyond  $w$ .

If we had discarded samples without vertices with degree  $d$  and made the average of  $f_{\theta,d}$  over the rest, this correction would not be necessary; however this might have resulted in more noise in the experimental results.

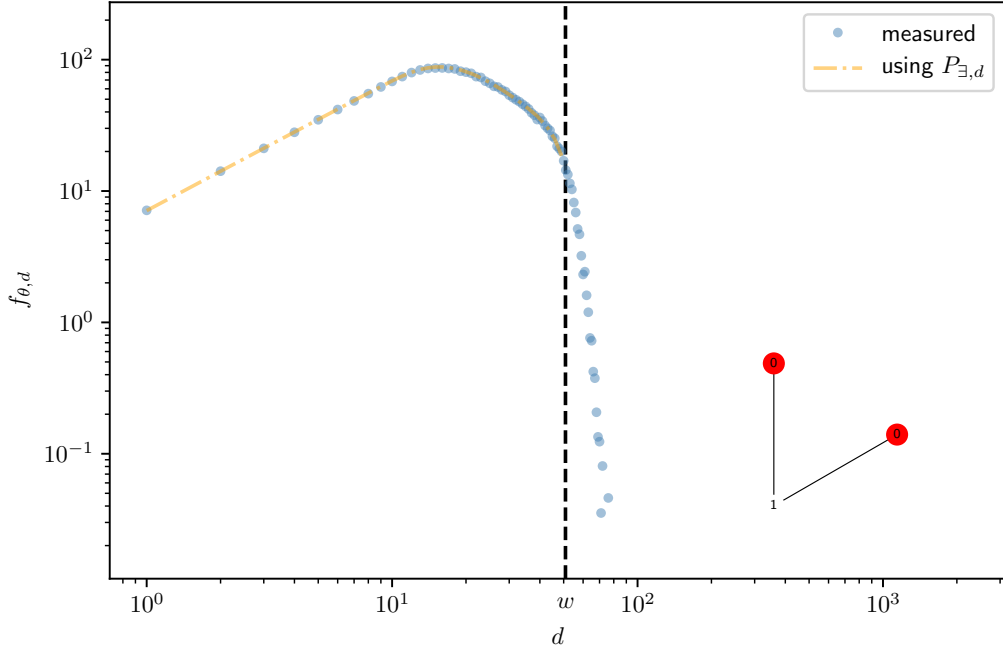


Figure 3.6.: High degree correction example

### 3.7.2. Correcting the low degree deviation & extending the range beyond $w$

The low degree deviation and the missing range beyond  $w$  can be explained by dropping the approximation  $h \approx d$ . Consider that we are measuring the average number of times vertex  $v_1$  ends up in a graphlet-orbit  $(c, o)$ . Since any set of  $k - 1$  vertices, put together with  $v_1$ , will statistically form the same mean number of said occurrences, we can consider that these vertices are  $\{v_2, \dots, v_k\} =: v_{g^-}$ , and multiply the ensuing result by pre-factor  $\binom{n-1}{k-1}$ . There are  $|G_\theta|$  ways of placing edges between  $v_g$  so that the subgraph  $g$  induced by  $v_1, \dots, v_k$  is isomorphic to the graph  $c$  and  $v_1$  is in orbit  $o$ . For any possible concretization  $g'$  of  $g$  that is isomorphic to  $c$ , the odds of  $g = g'$  and  $\deg(v_1, G) = d$  are the same. Without loss of generality, we relabel the vertices so that  $g' = c$  and  $v_1$  goes to  $v_o$ , a vertex in the orbit  $o$  of  $c$ , with hidden variable  $h_{v_o}$ . All that is left is to do is to calculate  $N_{\theta,d}$ , the joint probability of  $g = c$  and of  $v_o$  having degree  $d$  in  $G$ . For now we can write:

$$f_{\theta,d}(\theta, d) = \frac{\binom{n-1}{k-1} |G_\theta| N_{\theta,d}(\theta, d)}{P_d(d)} P_{\exists,d}(d) \quad (3.41)$$

Where  $P_d(d)$  is the probability of a vertex in  $G$  having degree  $d$ , and we have already taken the first correction in account with the  $P_{\exists,d}$  factor.

We proceed by determining  $N_{\theta,d}$ . The vertex  $v_o$  in  $g$  will have a fixed degree which we name  $t$ . Therefore for  $v_o$  to have degree  $d$  in  $G$ , it must have  $d - t$  edges to  $\{v_{k+1}, \dots, v_n\} =: v_{G/g}$ .

### 3. Analytical Approach to Orbit Counts

Refer to [Figure 3.7](#) for a diagram.

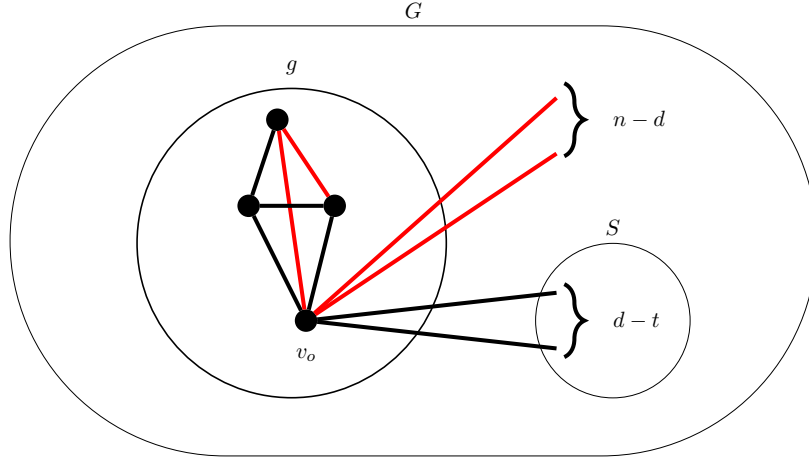


Figure 3.7.: Diagram for [Equation 3.42](#). Red edges are absent edges.

There is no condition imposed on the edges between the vertices  $v_{g^-}$  and  $v_{G/g}$ , or in-between vertices in  $v_{G/g}$  so the corresponding factors are separable and through integration on the hidden variables we obtain:

$$\begin{aligned}
 N_{\theta,d}(\theta, d) &:= \left\langle p_g(c, [h_1, h_2, \dots, h_k]) \sum_{\substack{S \subseteq v_{G/g} \\ |S| = d-t}} \prod_{i \in [k+1, \dots, n]} p_X(v_i \in S, \frac{h_{v_o} h_i}{w^2}) \right\rangle_{[h_1, h_2, \dots, h_n]} \\
 &= \left\langle \langle p_g(c, H) \rangle_{H|h_{v_o}=h} \binom{n-k}{d-t} \left( \frac{h \langle h \rangle}{w^2} \right)^{(d-t)} \left( 1 - \frac{h \langle h \rangle}{w^2} \right)^{(n-k)-(d-t)} \right\rangle_h \\
 &= \sum_{j=1}^k \beta_{j,\theta} \langle u^j P(x = d-t | x \sim \text{Binomial}(n-k, u \langle u \rangle)) \rangle_u
 \end{aligned} \tag{3.42}$$

Where in [Equation 3.42](#) we use the random variable  $u := h/w$ , where  $h \sim D_h$ , and  $\langle p_g(c, H) \rangle_{H|h_1=h}$  using [Equation 3.35](#) is written as the polynomial  $\sum_{j=1}^k \beta_{j,\theta} u^j$ .

To give an example, we set  $\theta = (c, o)$  so that  $c$  is the v-like graph of 3 vertices and  $o$  contains the vertex with degree 2. In [Figure 3.8](#) we plot the integrand inside the average over  $u$ ,  $\rho$  versus  $u$  for a different values of  $d$ , and in [3.9a](#) we plot  $N_{\theta,d}$ . In practice, this figure helped us verifying our numerical integration and understanding how exactly the distribution of the degree drifted beyond  $w$ .

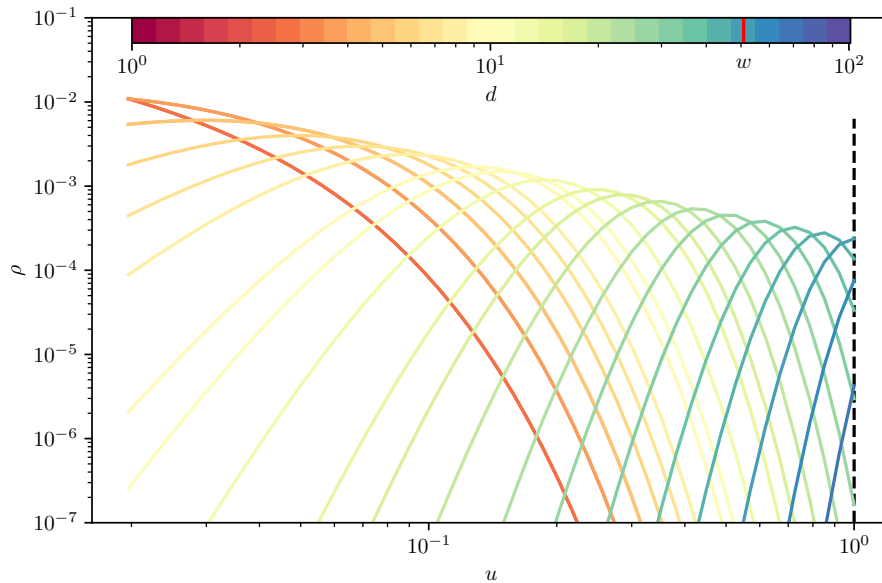
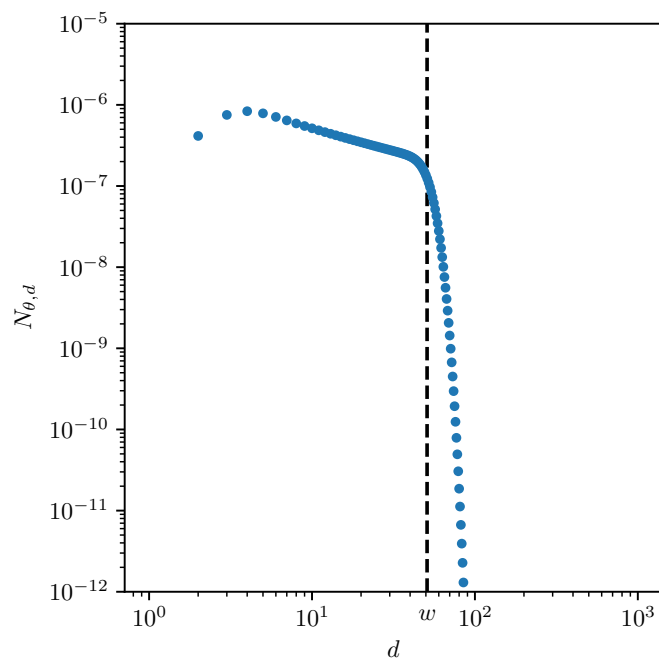


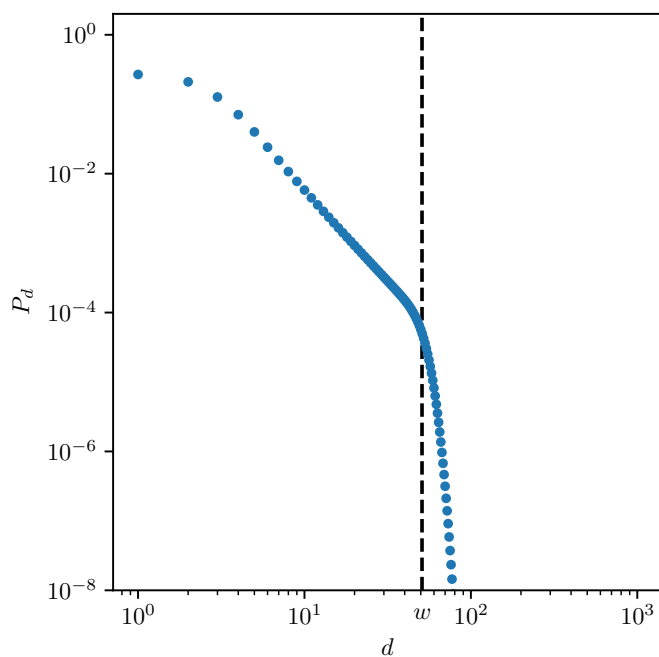
Figure 3.8.: Probability density  $\rho$  corresponding to the integrand of the expression for  $N_{\theta,d}$ , vs.  $u$  in its full domain, for a representative discrete range of  $d$ , and a  $y$ -axis minimum of  $10^{-7}$ .

Since the binomials in  $N_{\theta,d}$  and  $P_d$  are very similar and have an exponential peak at  $u = d/w$ , for large  $n$  and away from  $w$  they will practically have the same value. This means that  $f_{\theta,d}(\theta, d)/P_{\exists,d}(d)$  will be  $\propto \langle p_g(c, H) \rangle_{H|h_1=d} = \sum_{j=1}^k \beta_{j,\theta} d^j$  (see Figure 3.10 and Figure 3.9 for the numerator and denominator of  $f_{\theta,d}$ ). In the result plots, the formula in Equation 3.41 is referred to by “using  $N_{\theta,d}$ ”. Comparing this approximation to the line “using  $P_{\exists,d}$ ” confirms this second correction can account for the low degree deviation (see results in Figure 3.4), and the high range beyond  $w$ .

### 3. Analytical Approach to Orbit Counts



(a)



(b)

Figure 3.9.: Leftmost numerator and denominator for Equation 3.41.



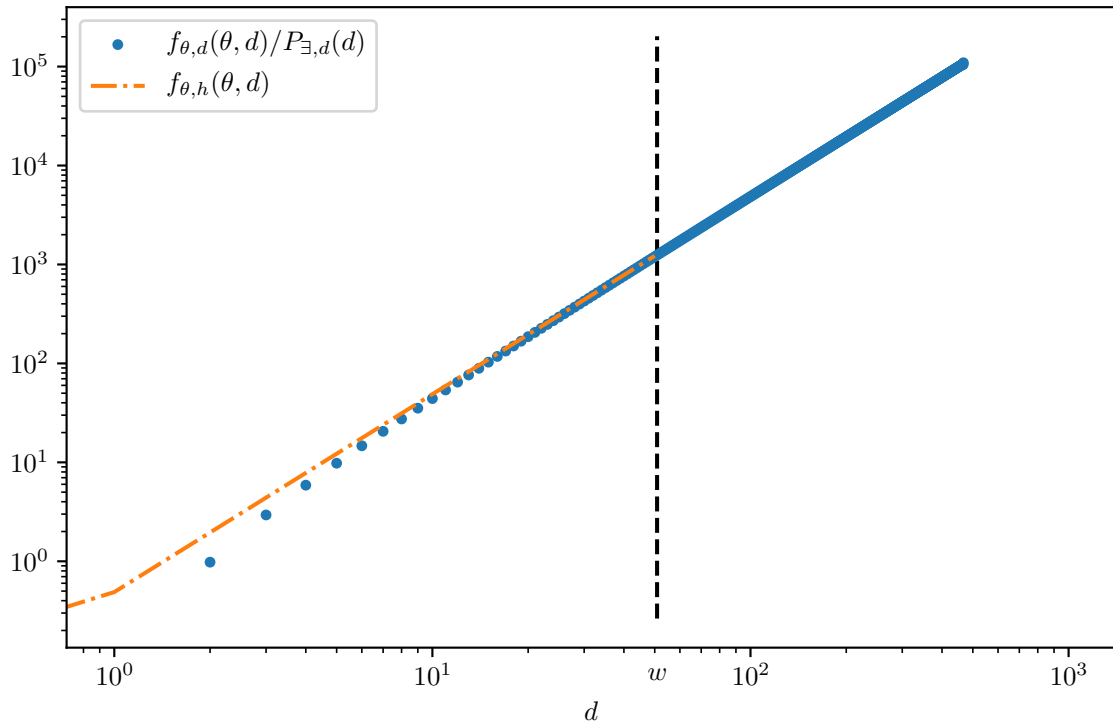


Figure 3.10.:  $f_{\theta,d}(\theta, d)/P_{\exists,d}(d)$  and  $f_{\theta,h}(\theta, d)$  versus  $d \in \{t, \dots, n - 1\}$

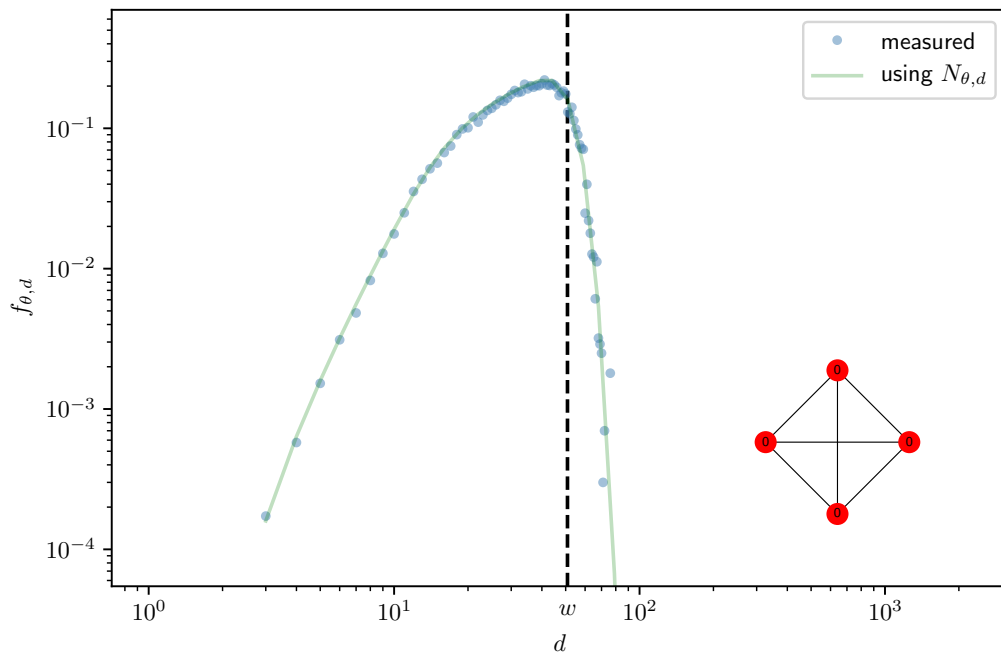
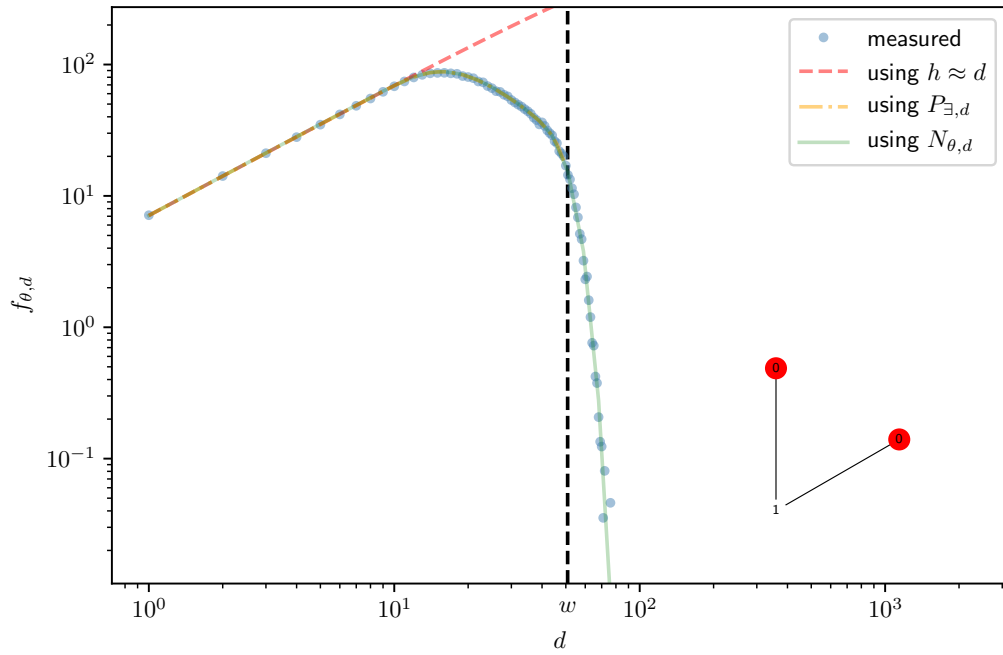


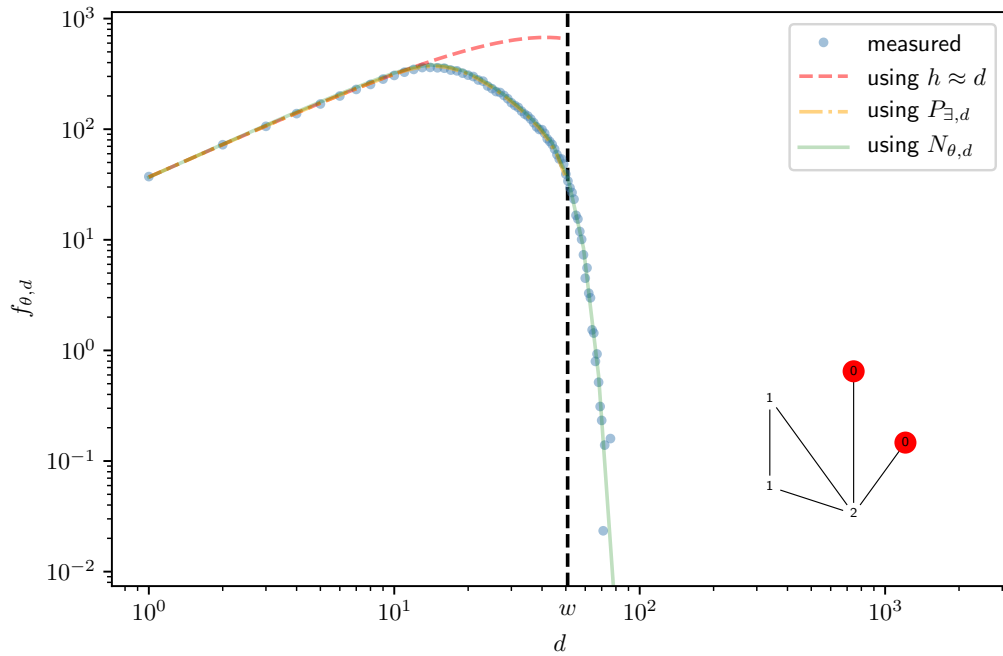
Figure 3.11.: Low degree correction example.

### 3. Analytical Approach to Orbit Counts

Illustrative examples plotting together all approximations are in Figure 3.12.

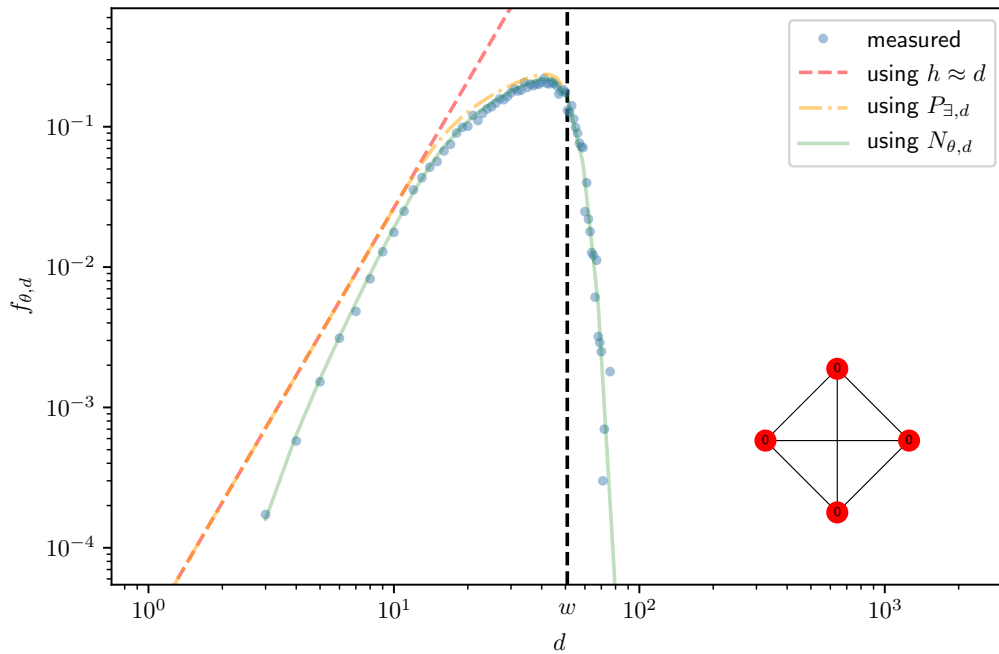


(a)



(b)

Figure 3.12.: Experimental orbit counts and theoretical approximations versus degree. In the lower right, graphlet, with nodes labeled by orbit, and in red the orbit being counted.



(c)

Figure 3.12 continued

### 3.8. Conclusions and Future Work

Results qualitatively match the theoretical values well in the log-log scale. Further verification could involve varying the power-law  $\alpha$  parameter, changing the hidden variable distribution altogether, increasing the graphlet size  $k$ , and quantifying the error over these changes via a global measure.

The formulas presented open the way for further analytical understanding of orbit counts. Some derivations are general to any random hidden variables model, and most are applicable to an expected degree model with an arbitrary distribution. The methods and steps used seem to be flexible enough to derive different measures. While we are quite certain of the initial mean value for the graphlet orbit  $h$ -frequency, the refinements made afterward lack the same rigor. Future work should determine the range of parameters where these approximations are sufficient or valid.

We did not cover the variance of the measures, which is essential for replacing simulation by analytical formulas to obtain statistics of a null-model. The same works that have derived formulas for the mean of graphlet counts also have derived the corresponding standard deviation. Having analyzed the steps taken for these variance derivations, we concluded that once means were calculated, applying the same logic to graphlet orbits should be

### *3. Analytical Approach to Orbit Counts*

possible. Therefore we prioritized understanding the mean behavior first, and left the variance calculations for future work.

In respect to variance, in general we expect high degree vertices link to many low degree vertices, and high degree vertices, since they have very low probability of appearing, will likely create high variance in orbit counts due to the many orbits that appear between one high degree vertex and many low degree vertices. Measurements on input data will be comparable to the null-model in a degree interval of low uncertainty. Determining this usable interval could be an interesting endeavor.

# Bibliography

- [1] Réka Albert and Albert-László Barabási. “Statistical Mechanics of Complex Networks”. In: *Reviews of Modern Physics* 74.1 (Jan. 30, 2002), pp. 47–97. DOI: [10.1103/RevModPhys.74.47](https://doi.org/10.1103/RevModPhys.74.47) (cit. on p. 1).
- [2] Luciano da Fontoura Costa et al. “Analyzing and Modeling Real-World Phenomena with Complex Networks: A Survey of Applications”. In: *Advances in Physics* 60.3 (June 2011), pp. 329–412. ISSN: 0001-8732, 1460-6976. DOI: [10.1080/00018732.2011.572452](https://doi.org/10.1080/00018732.2011.572452) (cit. on p. 1).
- [3] L. da F. Costa et al. “Characterization of Complex Networks: A Survey of Measurements”. In: *Advances in Physics* 56.1 (Jan. 2007), pp. 167–242. ISSN: 0001-8732, 1460-6976. DOI: [10.1080/00018730601170527](https://doi.org/10.1080/00018730601170527) (cit. on p. 1).
- [4] Ron Milo et al. “Network Motifs: Simple Building Blocks of Complex Networks”. In: *Science (New York, N.Y.)* 298 (Nov. 1, 2002), pp. 824–7. DOI: [10.1126/science.298.5594.824](https://doi.org/10.1126/science.298.5594.824) (cit. on pp. 1, 5).
- [5] Olaf Sporns and Rolf Kötter. “Motifs in Brain Networks”. In: *PLoS Biology* 2.11 (Oct. 26, 2004). Ed. by Karl J. Friston, e369. ISSN: 1545-7885. DOI: [10.1371/journal.pbio.0020369](https://doi.org/10.1371/journal.pbio.0020369) (cit. on p. 2).
- [6] Jeannette Janssen, Matt Hurshman, and Nauzer Kalyaniwalla. “Model Selection for Social Networks Using Graphlets”. In: *Internet Mathematics* 8.4 (Dec. 2012), pp. 338–363. ISSN: 1542-7951, 1944-9488. DOI: [10.1080/15427951.2012.671149](https://doi.org/10.1080/15427951.2012.671149) (cit. on p. 2).
- [7] Sergi Valverde and Ricard V. Solé. “Network Motifs in Computational Graphs: A Case Study in Software Architecture”. In: *Physical Review E* 72.2 (Aug. 8, 2005). ISSN: 1539-3755, 1550-2376. DOI: [10.1103/PhysRevE.72.026107](https://doi.org/10.1103/PhysRevE.72.026107) (cit. on p. 2).
- [8] Stephen A. Cook. “The Complexity of Theorem-Proving Procedures”. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing - STOC '71*. The Third Annual ACM Symposium. Shaker Heights, Ohio, United States: ACM Press, 1971, pp. 151–158. DOI: [10.1145/800157.805047](https://doi.org/10.1145/800157.805047) (cit. on p. 2).
- [9] Natasa Przulj. “Biological Network Comparison Using Graphlet Degree Distribution”. In: *Bioinformatics (Oxford, England)* 23.2 (Jan. 15, 2007), e177–183. ISSN: 1367-4811. DOI: [10.1093/bioinformatics/btl1301](https://doi.org/10.1093/bioinformatics/btl1301). pmid: [17237089](https://pubmed.ncbi.nlm.nih.gov/17237089/) (cit. on p. 2).

## Bibliography

- [10] Pedro Paredes and Pedro Ribeiro. “Towards a Faster Network-Centric Subgraph Census”. In: *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM '13*. The 2013 IEEE/ACM International Conference. Niagara, Ontario, Canada: ACM Press, 2013, pp. 264–271. ISBN: 978-1-4503-2240-9. DOI: [10/gf27p2](https://doi.org/10/gf27p2) (cit. on pp. 3, 5, 6, 11, 17).
- [11] Sebastian Wernicke. “Efficient Detection of Network Motifs”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 3.4 (Oct. 2006), pp. 347–359. ISSN: 1545-5963. DOI: [10.1109/TCBB.2006.51](https://doi.org/10.1109/TCBB.2006.51) (cit. on p. 5).
- [12] Brendan D. McKay and Adolfo Piperno. “Practical Graph Isomorphism, II”. In: *Journal of Symbolic Computation* 60 (Jan. 2014), pp. 94–112. ISSN: 07477171. DOI: [10.1016/j.jsc.2013.09.003](https://doi.org/10.1016/j.jsc.2013.09.003) (cit. on pp. 5, 19).
- [13] Pedro Ribeiro and Fernando Silva. “G-Tries: A Data Structure for Storing and Finding Subgraphs”. In: *Data Mining and Knowledge Discovery* 28.2 (Mar. 2014), pp. 337–377. ISSN: 1384-5810, 1573-756X. DOI: [10.1007/s10618-013-0303-4](https://doi.org/10.1007/s10618-013-0303-4) (cit. on pp. 5, 17).
- [14] Pedro Ribeiro et al. “A Survey on Subgraph Counting: Concepts, Algorithms and Applications to Network Motifs and Graphlets”. In: (Oct. 28, 2019). arXiv: [1910.13011](https://arxiv.org/abs/1910.13011) [cs] (cit. on p. 5).
- [15] Miguel E. P. Silva, Pedro Paredes, and Pedro Ribeiro. “Network Motifs Detection Using Random Networks with Prescribed Subgraph Frequencies”. In: *Complex Networks VIII*. Ed. by Bruno Gonçalves et al. Cham: Springer International Publishing, 2017, pp. 17–29. ISBN: 978-3-319-54240-9 978-3-319-54241-6. DOI: [10.1007/978-3-319-54241-6\\_2](https://doi.org/10.1007/978-3-319-54241-6_2) (cit. on p. 7).
- [16] Tijana Milenković and Nataša Pržulj. “Uncovering Biological Network Function via Graphlet Degree Signatures”. In: *Cancer Informatics* 6 (Jan. 2008), CIN.S680. ISSN: 1176-9351, 1176-9351. DOI: [10.4137/CIN.S680](https://doi.org/10.4137/CIN.S680) (cit. on p. 16).
- [17] Alessandro Garavaglia and Clara Stegehuis. “Subgraphs in Preferential Attachment Models”. In: *Advances in Applied Probability* 51.03 (Sept. 2019), pp. 898–926. ISSN: 0001-8678, 1475-6064. DOI: [10.1017/apr.2019.36](https://doi.org/10.1017/apr.2019.36) (cit. on p. 17).
- [18] Fan Chung and Linyuan Lu. “Connected Components in Random Graphs with Given Expected Degree Sequences”. In: *Annals of Combinatorics* 6.2 (Nov. 2002), pp. 125–145. ISSN: 0218-0006. DOI: [10/ck4vpj](https://doi.org/10/ck4vpj) (cit. on pp. 17, 23).
- [19] Catherine Matias et al. “Networks Motifs: Mean and Variance for the Count”. In: *REVSTAT* 4.1 (Mar. 2006), pp. 31–51 (cit. on p. 17).
- [20] Giovanni Micale et al. “Fast Analytical Methods for Finding Significant Labeled Graph Motifs”. In: *Data Mining and Knowledge Discovery* 32.2 (Mar. 2018), pp. 504–531. ISSN: 1384-5810, 1573-756X. DOI: [10.1007/s10618-017-0544-8](https://doi.org/10.1007/s10618-017-0544-8) (cit. on p. 17).

- [21] Clara Stegehuis, Remco van der Hofstad, and Johan S. H. van Leeuwen. “Variational Principle for Scale-Free Network Motifs”. In: *Scientific Reports* 9.1 (Dec. 2019). ISSN: 2045-2322. DOI: [10.1038/s41598-019-43050-8](https://doi.org/10.1038/s41598-019-43050-8) (cit. on pp. 17, 22).
- [22] F. Picard et al. “Assessing the Exceptionality of Network Motifs”. In: *Journal of Computational Biology* 15.1 (Jan. 2008), pp. 1–20. ISSN: 1066-5277, 1557-8666. DOI: [10.1089/cmb.2007.0137](https://doi.org/10.1089/cmb.2007.0137) (cit. on p. 17).
- [23] Marián Boguñá and Romualdo Pastor-Satorras. “Class of Correlated Random Networks with Hidden Variables”. In: *Physical Review E* 68.3 (Sept. 15, 2003). ISSN: 1063-651X, 1095-3787. DOI: [10.1103/PhysRevE.68.036112](https://doi.org/10.1103/PhysRevE.68.036112) (cit. on p. 18).
- [24] Michael Artin. *Algebra*. Englewood Cliffs, N.J: Prentice Hall, 1991. 618 pp. ISBN: 978-0-13-004763-2 (cit. on p. 19).
- [25] Albert-László Barabási. “Scale-Free Networks: A Decade and Beyond”. In: *Science* 325.5939 (July 24, 2009), pp. 412–413. ISSN: 0036-8075, 1095-9203. DOI: [10.1126/science.1173299](https://doi.org/10.1126/science.1173299) (cit. on p. 25).





# A. Results Extended to All Graphlet-Orbits With $k \in \{3, 4, 5\}$

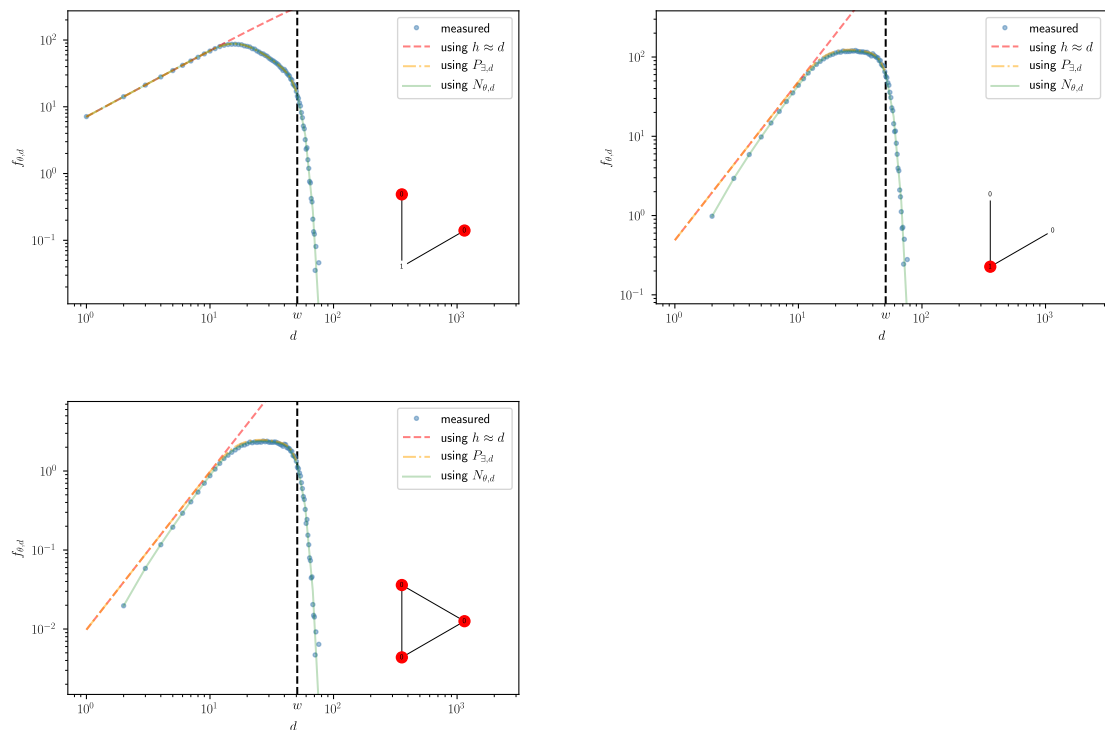


Figure A.1.: Results and theoretical approximations for all 72 graphlet-orbits with  $k \in \{3, 4, 5\}$ .

A. Results Extended to All Graphlet-Orbits With  $k \in \{3, 4, 5\}$

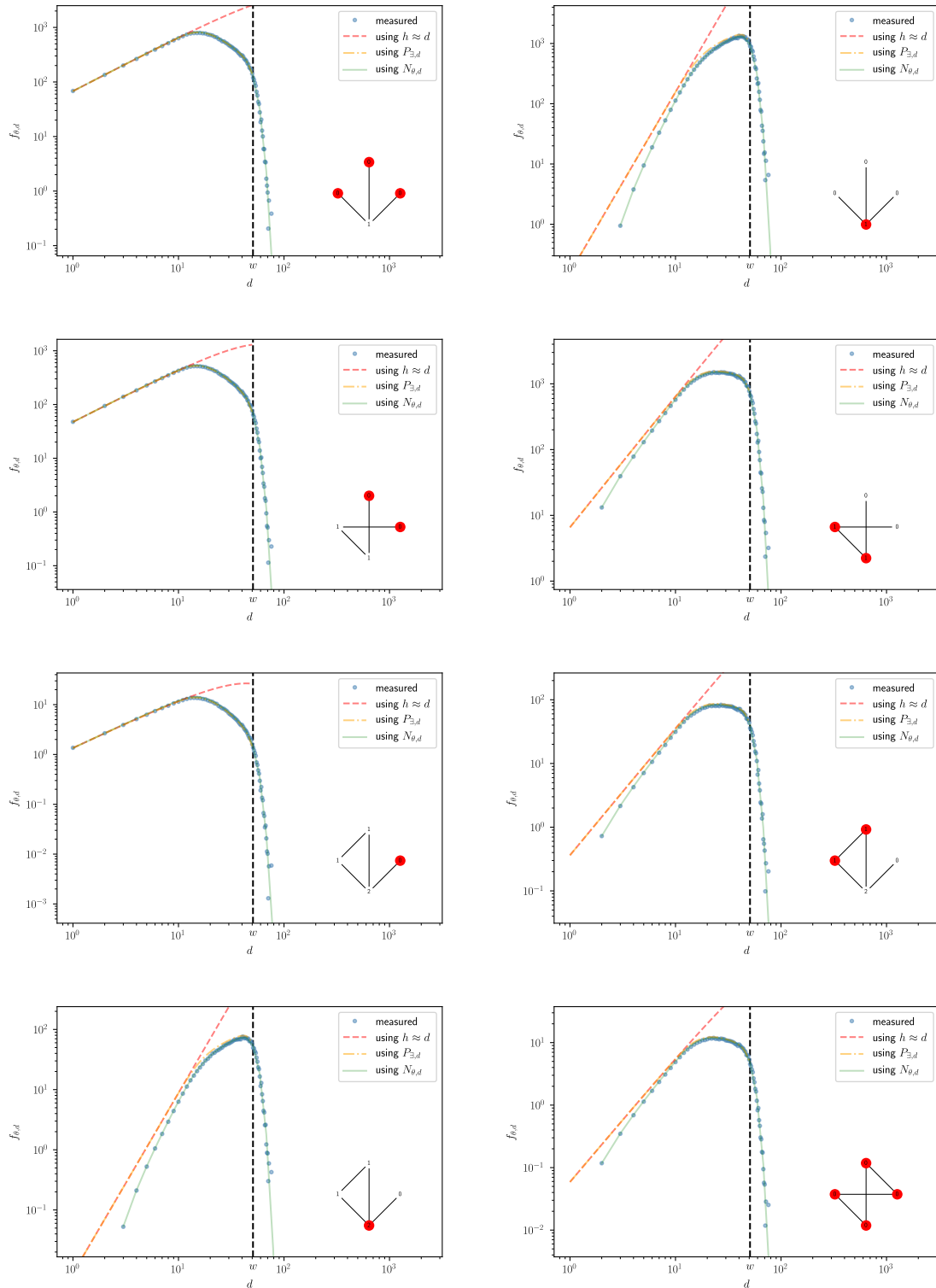


Figure A.1 continued

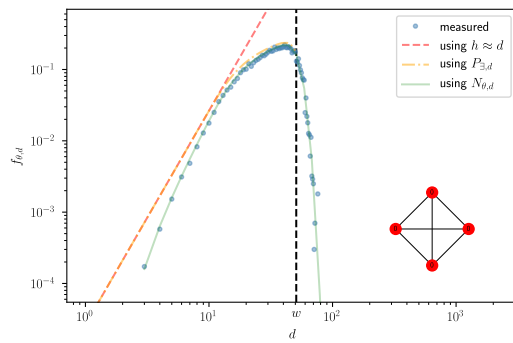
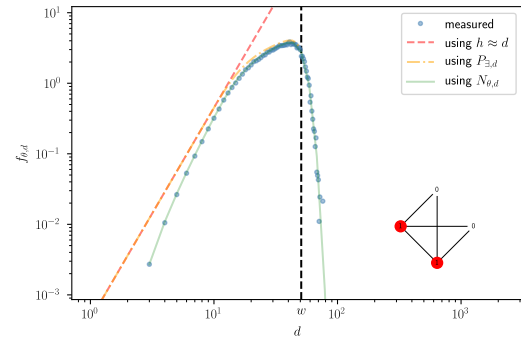
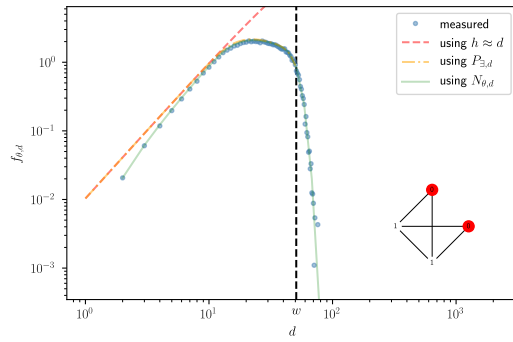


Figure A.1 continued

A. Results Extended to All Graphlet-Orbits With  $k \in \{3, 4, 5\}$

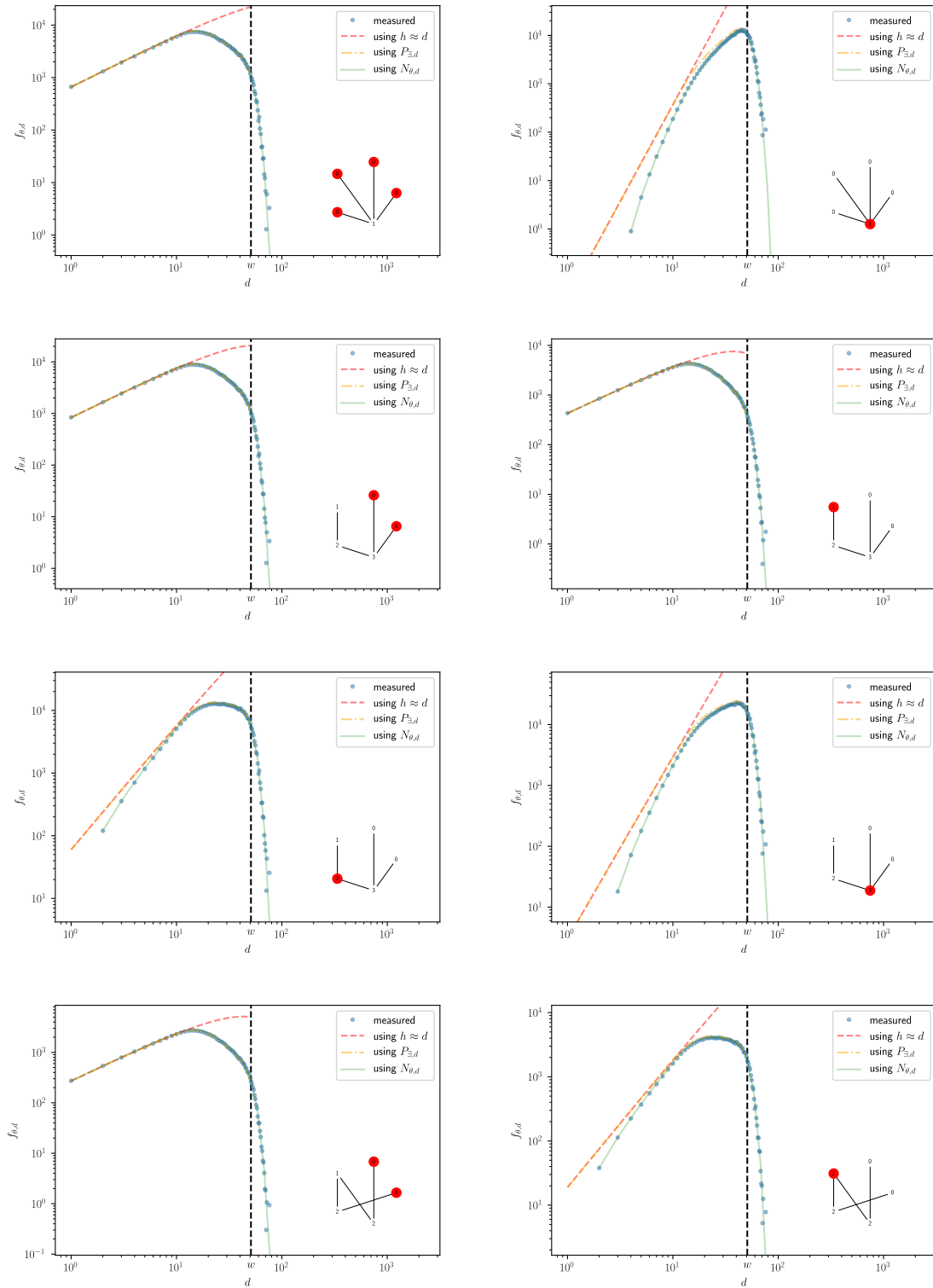


Figure A.1 continued

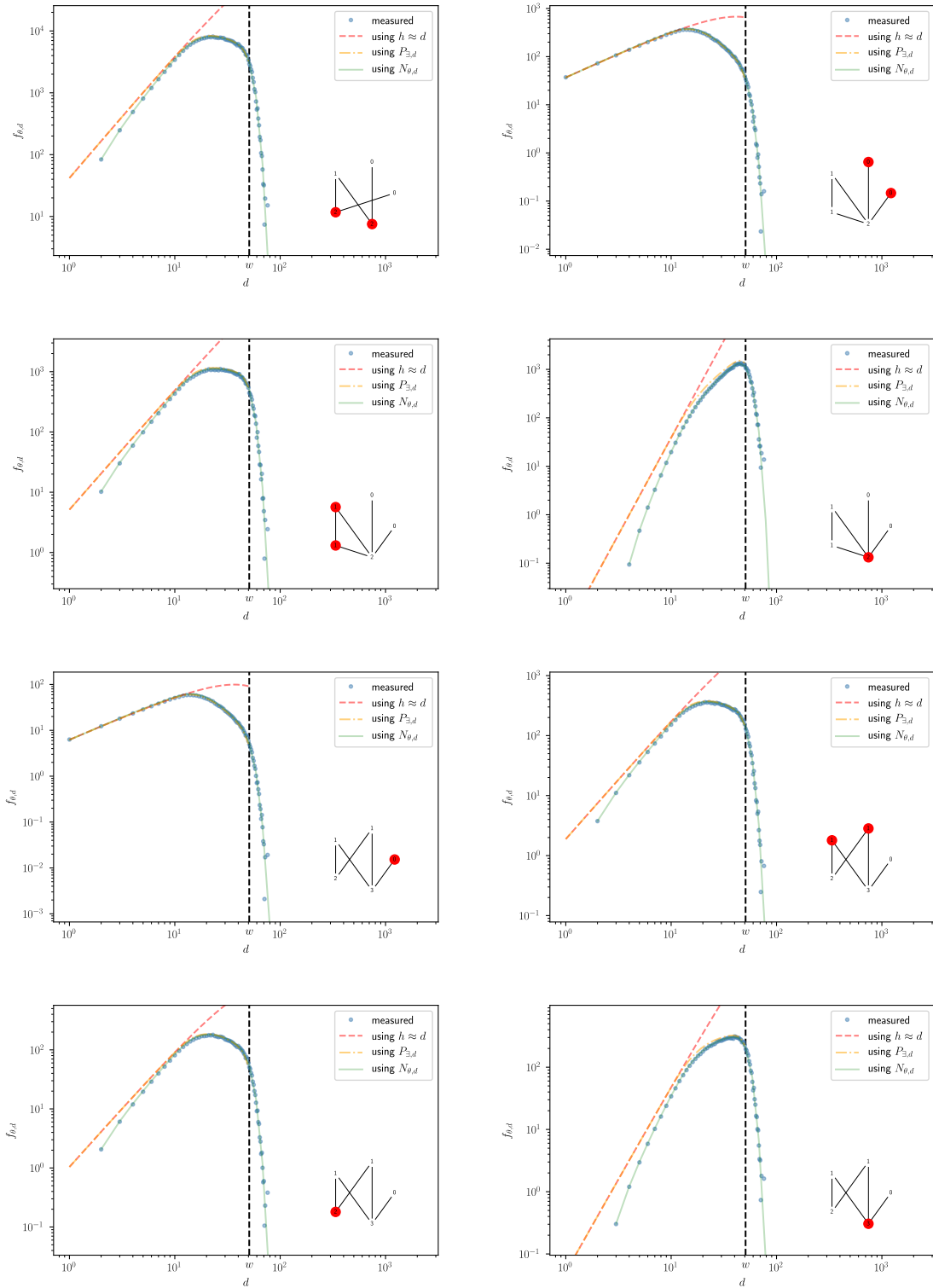


Figure A.1 continued

A. Results Extended to All Graphlet-Orbits With  $k \in \{3, 4, 5\}$

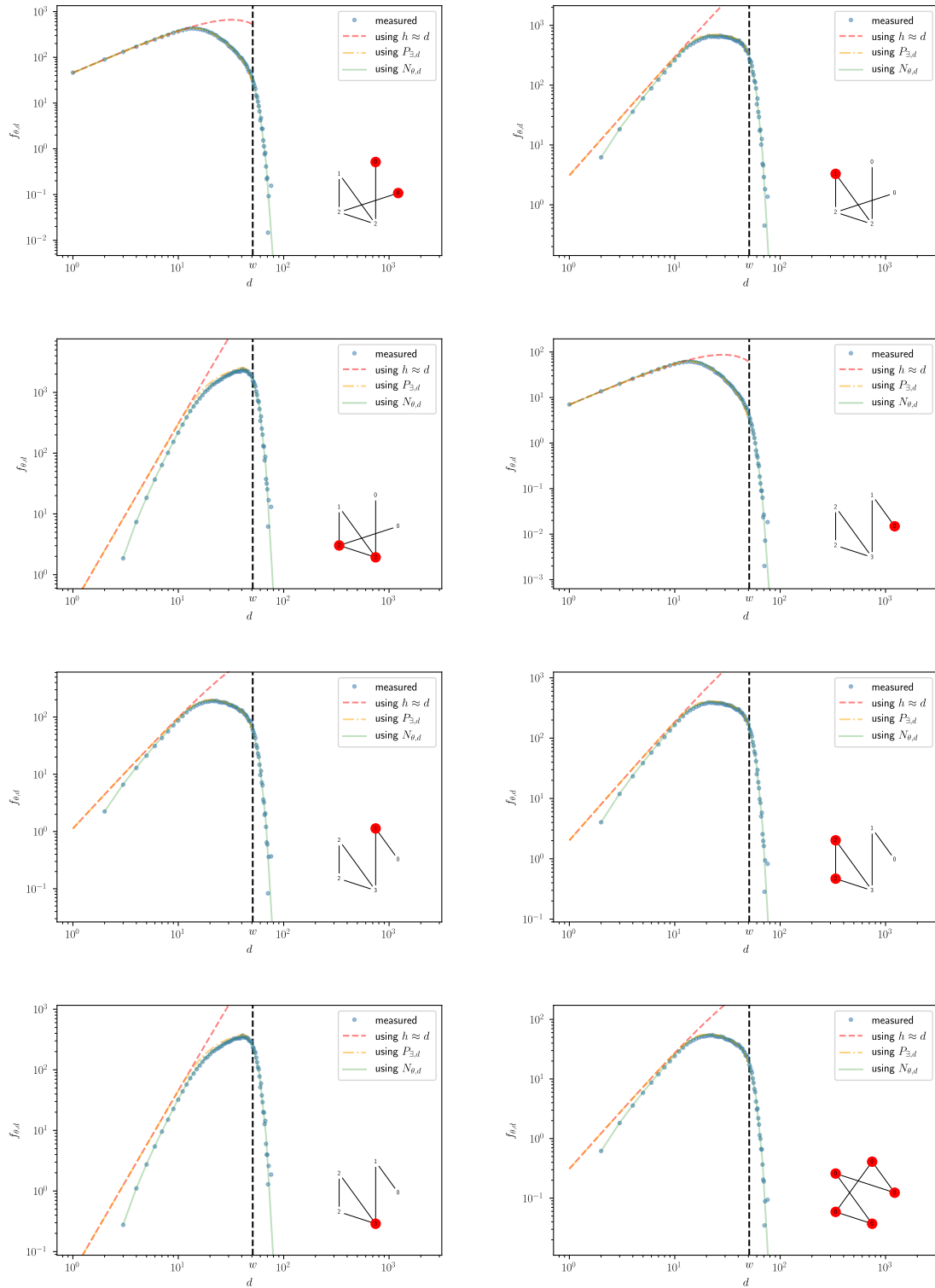


Figure A.1 continued

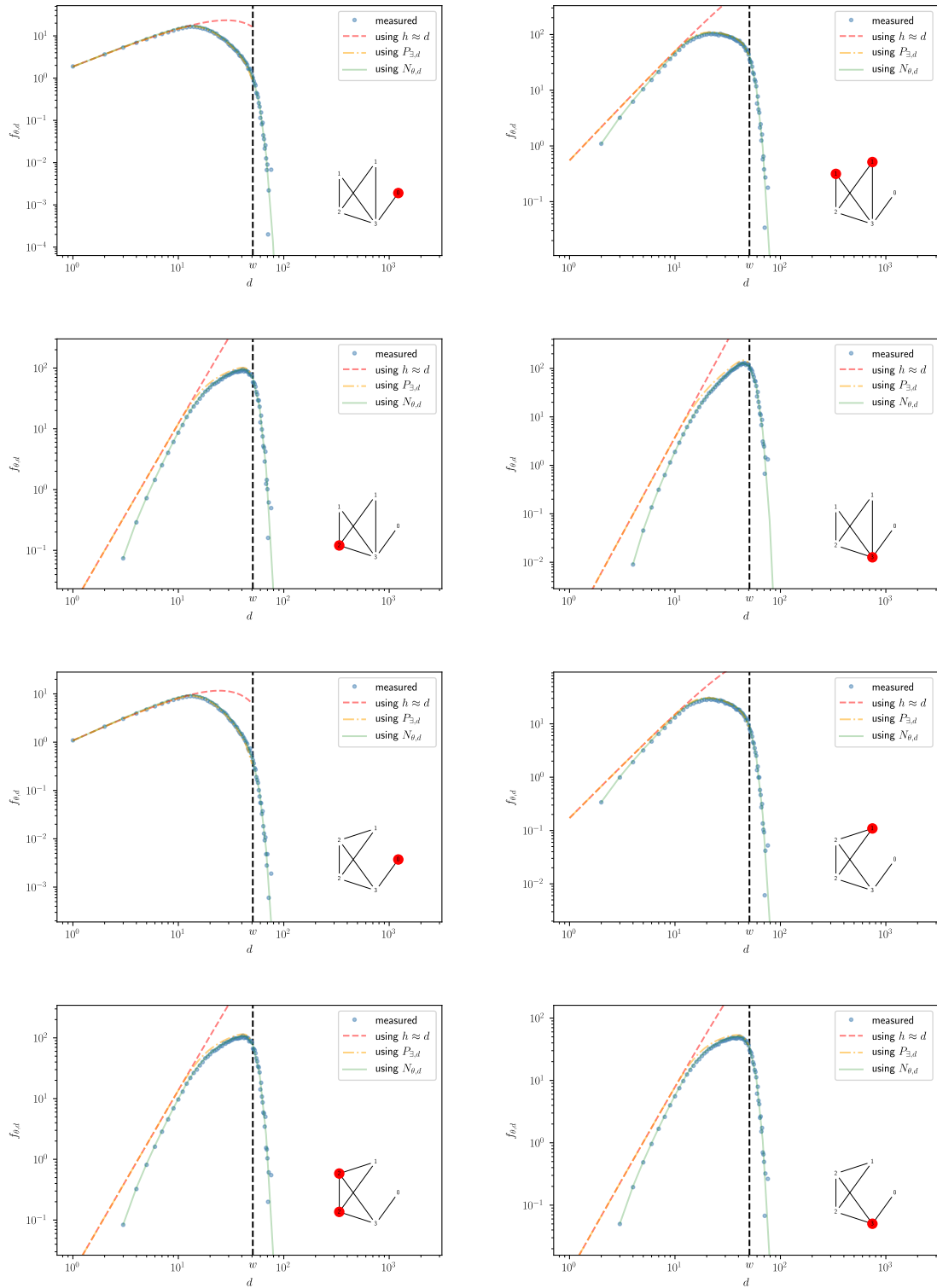


Figure A.1 continued

A. Results Extended to All Graphlet-Orbits With  $k \in \{3, 4, 5\}$

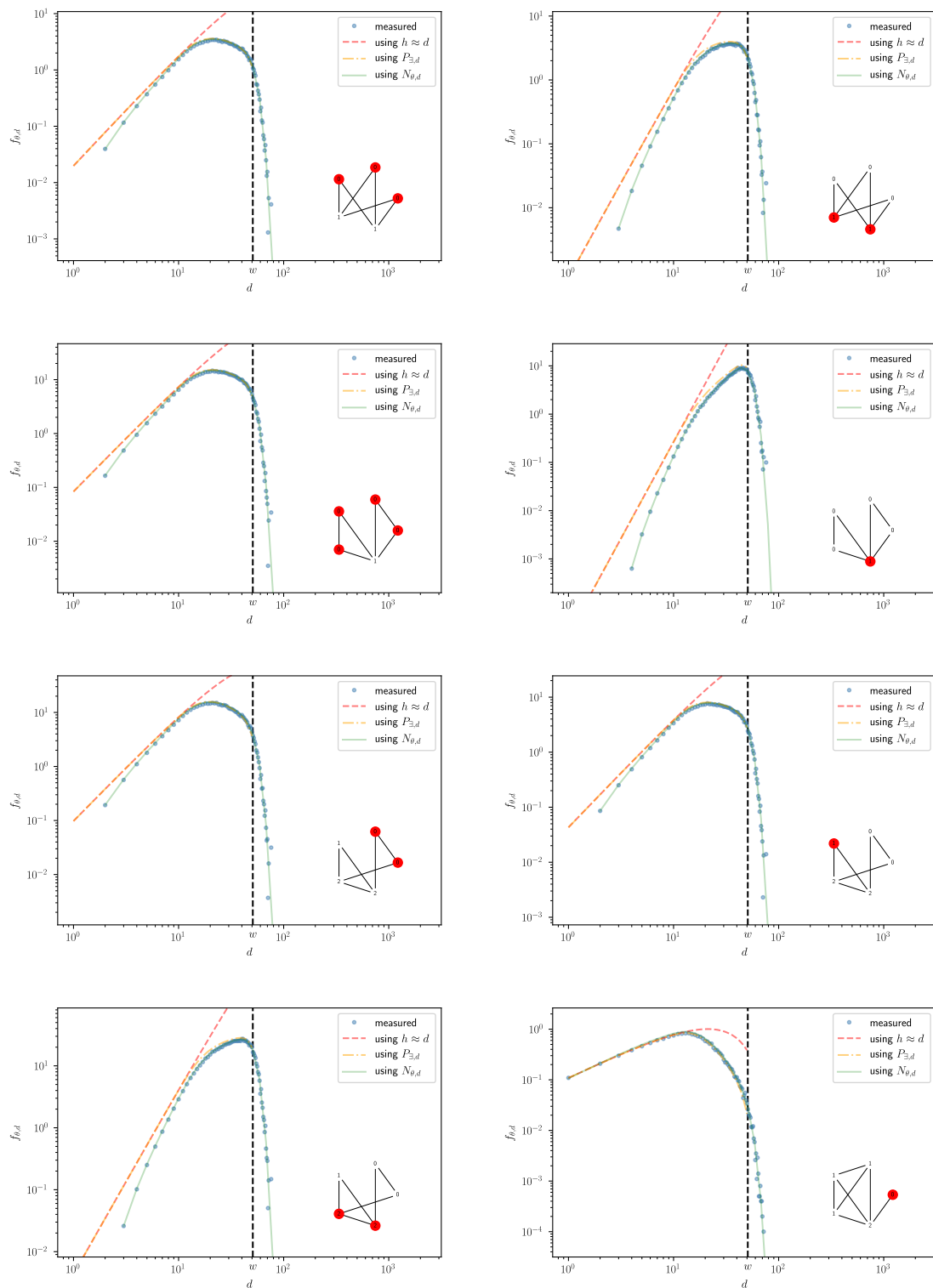


Figure A.1 continued



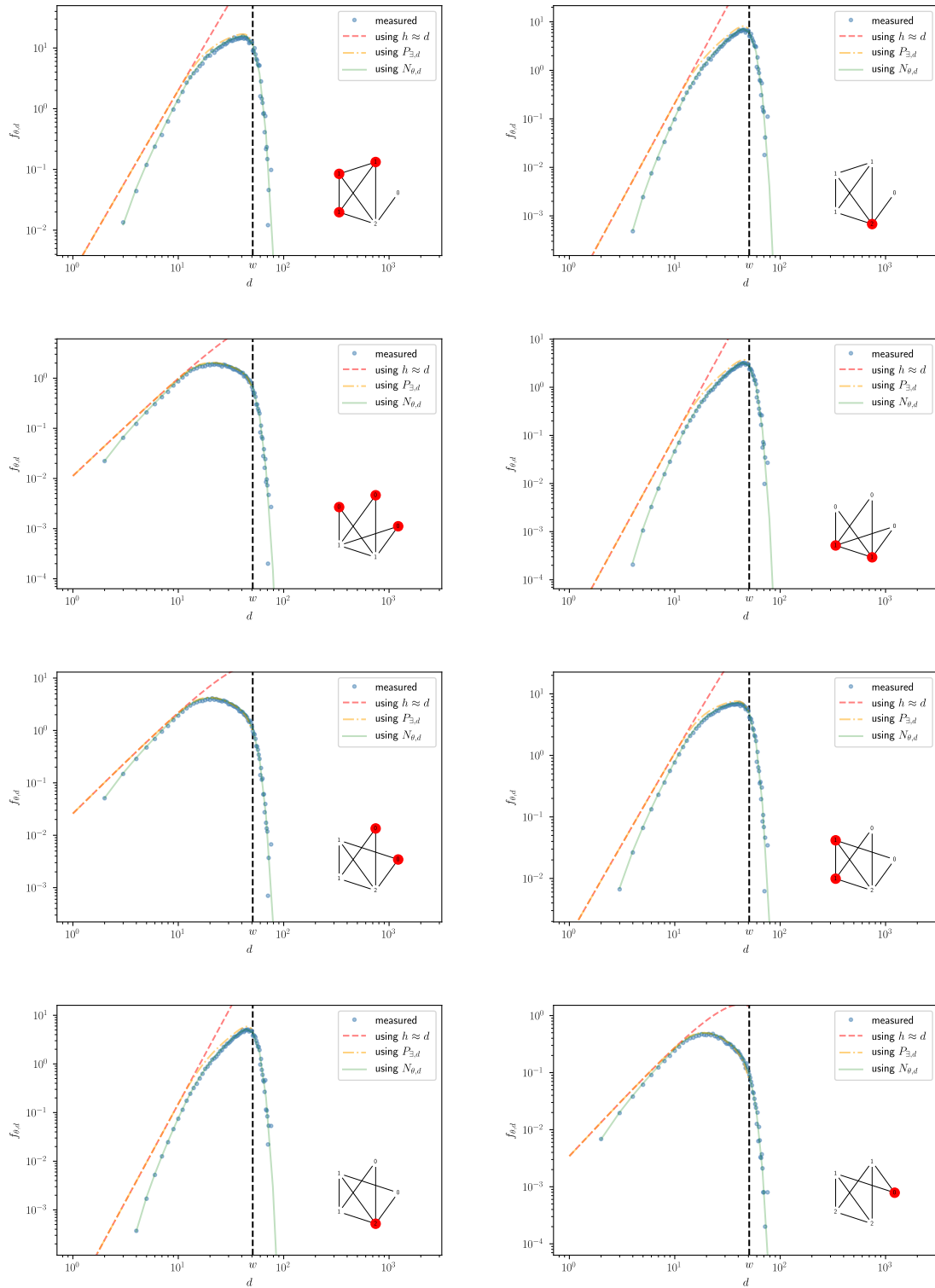


Figure A.1 continued

A. Results Extended to All Graphlet-Orbits With  $k \in \{3, 4, 5\}$

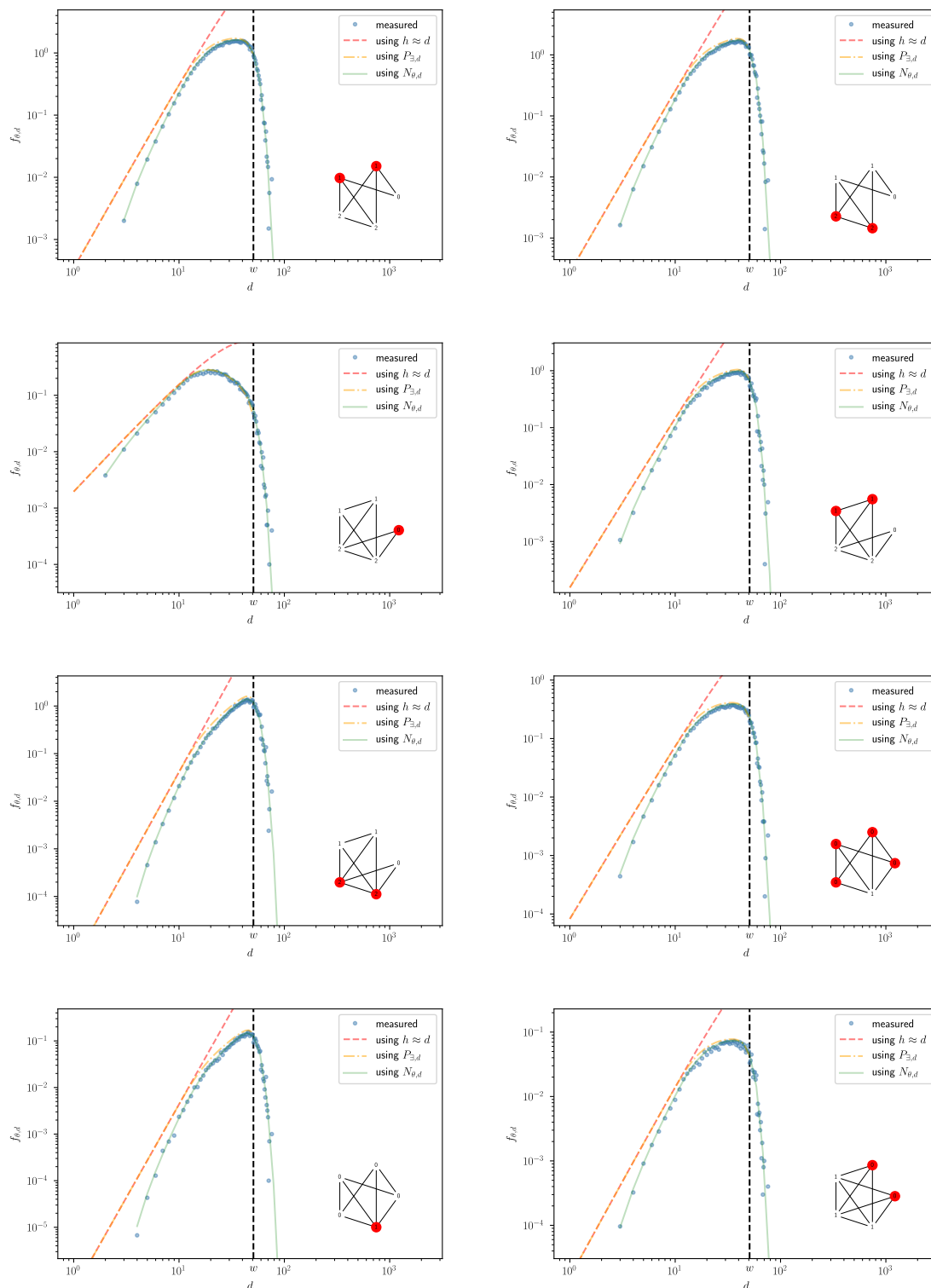


Figure A.1 continued

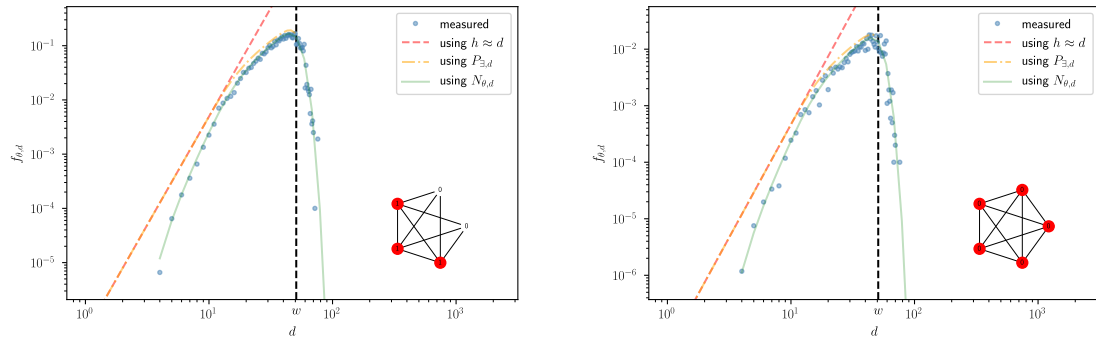


Figure A.1 continued

A. *Results Extended to All Graphlet-Orbits With  $k \in \{3, 4, 5\}$*