UPPSALA DISSERTATIONS IN MATHEMATICS

144

Community Structure in Graphs
Algorithms and Theoretical Guarantees

Vilhelm Agdur

Department of Mathematics
Uppsala University
UPPSALA 2025

Dissertation presented at Uppsala University to be publicly examined in Room 2001, Ångströmlaboratoriet, Regementsvägen 10, Uppsala, Friday, 19 September 2025 at 09:00 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Professor Amin Coja-Oghlan (TU Dortmund).

Abstract

Agdur, V. 2025. Community Structure in Graphs. Algorithms and Theoretical Guarantees. *Uppsala Dissertations in Mathematics* 144. 43 pp. Uppsala. ISBN 978-91-506-3128-9.

This thesis consists of three papers in the area of community detection in graphs, together with an introductory section and summaries of the papers.

In Paper I, we study the problem of finding overlapping clusterings of hypergraphs, continuing the line of research started by Carlsson and Mémoli (2013) of classifying clustering schemes as functors. We extend their notion of representability to the overlapping case, showing that any representable overlapping clustering scheme is excisive and functorial, and any excisive and functorial clustering scheme is isomorphic to a representable clustering scheme.

In Paper II, we prove a new lower bound on the modularity of a graph, solely in terms of its degree sequence. We establish that, under mild assumptions on the tail of the degree sequence, a graph with average degree d has modularity of order $d^{-1/2}$, and also use this result to give lower bounds for the modularity of preferential attachment graphs.

In Paper III, we study the algorithmic problem of optimizing the modularity of *temporal* graphs, that is, graphs that may vary over time. We prove that computing a multiplicative approximation of this maximum temporal modularity is tractable parameterised by the underlying treewidth of the graph.

Keywords: community detection, graph theory, graph clustering

Vilhelm Agdur, Department of Mathematics, Box 480, Uppsala University, SE-75106 Uppsala, Sweden.

© Vilhelm Agdur 2025

ISSN 1401-2049
ISBN 978-91-506-3128-9
URN urn:nbn:se:uu:diva-564552 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-564552)

Non nobis, Domine, non nobis,
sed nomini tuo da gloriam

List of papers

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.

I V. Agdur. *A classification of overlapping clustering schemes for hypergraphs*, 2025. arXiv:2404.03332. *Submitted*.

II V. Agdur, N. Kamčev, and F. Skerman. *Universal lower bound for community structure of sparse graphs*, 2023. arXiv:2307.07271. *Submitted*.

III V. Agdur, J. Enright, L. Larios-Jones, K. Meeks, F. Skerman, and E. Yates. *Approximating temporal modularity on graphs of small underlying treewidth*, 2025. arXiv:2507.17541. Accepted in *ALGOWIN 2025*, won Best Student Paper.

Reprints were made with permission from the publishers.

Contents

1 Introduction 9

2 Background 11
2.1 Community detection 11
2.2 Modularity 12
2.3 Temporal graphs 14
2.4 Fixed-parameter tractability 16
2.5 Treewidth 17
2.6 Category theory 21

3 Summary of papers 25
3.1 Paper I: A classification of overlapping clustering schemes for hypergraphs 25
3.2 Paper II: Universal lower bound for community structure of sparse graphs 27
3.3 Paper III: Approximating temporal modularity on graphs of small underlying treewidth 30

4 Sammanfattning på svenska 33

5 Acknowledgements 37

References 41

1. Introduction

Why, hear him, hear him babble and drop down to his nest,
But his own nest, wild nest, no prison.
Man's spirit will be flesh-bound when found at best,
But uncumbered: meadow-down is not distressed
For a rainbow footing it nor he for his bónes rísen.

The Caged Skylark, Gerard Manley Hopkins

Suppose you are sitting in a park sometime in late September, enjoying the last days of summer watching the sparrows hunt for insects, and a squirrel carrying off some nuts. Having studied by far too much mathematics, you start turning this natural beauty into an abstract model: Instead of seeing the fluffy tail of the squirrel, you see a vertex in a graph labelled *Sciurus vulgaris*, and instead of seeing the tiny alert eyes of the bird, you see an edge from the vertex *Passer domesticus* to the vertex *Aphis pomi*.

Then, as is wont to happen, a mosquito bites you. This naturally leads to the question: Could we eradicate these mosquitoes, or would that cause a bigger problem?

In your new mathematical model, you can start to reason about this question. If we remove the vertex *Culex pipiens*, this will of course affect every other species that predates upon the mosquitoes, and every species that is predated upon by the mosquito — and that could in turn have effects even further up- or downstream from the mosquito.

However, if there is no path, or only a very weak path, between some species and the mosquito, they probably will not be significantly affected. Even if the sparrows and the goshawks are affected by the lack of mosquitoes, perhaps the crayfish will be entirely unaffected?

Mathematically, this leads you to the following question, which is the actual topic of this thesis: Given a graph, how do we find a division of its vertices into parts which are well-connected internally, but have only weak connections between each other?

Having taken you on this scenic detour through a sunny park, we will proceed with more mathematical content. We begin with a background section, giving light introductions to the topics that appear in the papers in this thesis.

These are intended neither to be exhaustive accounts of the mathematical theory, nor to give thorough literature reviews on the area — the papers themselves contain such reviews and references. Rather, we provide some examples

that will hopefully be illuminating and point to the highlights of the area, to help the reader orient themselves on a first read through of the papers.

After the background section, we provide a summary of each of the papers in the thesis, stating the main results, some motivation for why the problem might be interesting to study, and giving a sketch of the storyline of the proofs in the paper.

Finally, there is a summary in Swedish of the thesis, which focuses on explaining the contents of the papers.

2. Background

You have arranged all things by measure and number and weight.

Wisdom 11:20

2.1 Community detection

Looked at very generally, the problem of community detection has its roots already in ancient times, in the theories of forms or genera. What is the relation between humanity and the collection of all humans — are humans the way they are because of a pre-existing humanity, or is humanity merely a generalization from the collection of all humans?

Whatever one's philosophical views on this, in science, we cannot directly observe the different genera of animals, only attempt to infer this information from observing the individual animals, and noting their similarities and differences.

This is, in the broadest sense, the problem of community detection: Given a collection of objects equipped with a similarity measure, we seek to divide them into groups, or communities, in a way that captures the underlying structure of the data.

This highly nominalist idea of types invites the question of whether the same methods are truly applicable to all kinds of data: If the groups we are detecting are different species of animals, is it really the same problem as if we are studying the citation network of scientific articles and detecting fields of research? Is the notion of "community" a coherent one that truly captures a similarity between these two problems?

This is not purely a philosophical question, but also a relevant one when studying community detection mathematically: There is no single mathematical definition of a community, so there is no principled way of comparing two different community detection methods *per se*.

What we can do instead is to compare the methods on either real-world data equipped with some ground truth, or see how they perform on some theoretical model that by design has a known community structure. The former, of course, is empirical science, not mathematics, and the latter merely moves the arbitrariness one step back: Instead of arbitrariness in the choice of method, we have arbitrariness in the choice of model we want the method to work on.

In practice, there are two broad classes of method for community detection:

1. Methods that, for each possible partition of the vertex set, assigns some quality score to that partition, and then treat community detection as an optimization problem with respect to that score.
2. Methods that have a rule for when vertices belong to the same part, and directly compute the resulting partition.

Methods in the second class are the first to be studied [20], and conceptually the simplest. The most elementary example in this class is *single-linkage clustering*, where we pick some threshold of similarity, and declare that two objects are in the same part if they are at least this similar.

While it is a very simple method to describe and study, single-linkage clustering has some significant drawbacks, such as the chaining issue, wherein two very dissimilar objects can be declared as of the same type because of a long sequence of intermediate objects linking them, each of which is sufficiently similar to the previous object in the chain.

Thus, there have been many variants on single-linkage attempting to rescue it from these problems while retaining the simplicity of the method. In Paper I, we study a very general version of this in the setting of finding overlapping partitions of hypergraphs.

In the first class, we find many of the most popular methods in use today, such as Infomap [32], maximum likelihood methods fitting a random graph model [30], and many more [21].

The most common method in this class is the method of *modularity optimization*, introduced by Newman and Girvan in 2004 [28]. In Paper II we find a lower bound for the modularity of a graph, and in Paper III we study a generalization of modularity to temporal graphs.[1] Because of its centrality to the topics of this thesis, modularity gets a section all of its own in this introduction.

2.2 Modularity

As discussed in the previous section, modularity optimization is likely the most widely used method of community detection in graphs.

Definition 2.2.1 (Modularity)**.** For a graph $G = (V,E)$ on n vertices with m edges and a partition $\mathscr{A} = \{A_1, A_2, \ldots, A_k\}$ of its vertices, i.e., a collection of disjoint sets $A_1, \ldots, A_k$ whose union is the entirety of V, the *modularity* of the partition $\mathscr{A}$ is given by

$$q(G, \mathscr{A}) = \sum_{i=1}^{k} \frac{e(A_i)}{m} - \frac{\mathrm{vol}(A_i)^2}{(2m)^2},$$

[1] For a definition and discussion of temporal graphs, see Section 2.3 of this background chapter.

where $e(A_i)$ is the number of edges within the vertex set A_i and $\mathrm{vol}(A_i)$ is the *volume*, or sum of vertex degrees, of A_i, i.e., $\mathrm{vol}(A_i) = \sum_{v \in A_i} d_v$.

The modularity of the graph G is the maximum of $q(G, \mathscr{A})$ over all partitions $\mathscr{A}$, so

$$q^*(G) = \max_{\mathscr{A} \text{ partition of } V} q(G, \mathscr{A}).$$

We can see in this definition that we are rewarded for having many edges within a part, but penalized for having a large part. This aligns with our intuition for what it means to be a good partition into communities — we want many intra-community edges, but not at the cost of putting all vertices into a single part, so we also need a penalty term for making the communities needlessly large.

A little bit of basic algebra manipulating the definition will make that intuition clearer.

Definition 2.2.2 (Alternative definition of modularity)**.** We may also compute the modularity of a partition as

$$q(G, \mathscr{A}) = \frac{1}{2m} \sum_{u,v \in V} \delta_{\mathscr{A}}(u,v) \left(\mathbb{1}_{uv \in E} - \frac{d_u d_v}{2m} \right), \tag{2.1}$$

where $\delta_{\mathscr{A}}(u,v)$ is the indicator function for u and v being in the same part.

Another intuition for why modularity would be a measure of the quality of a partition into communities becomes clear from seeing (2.1). Suppose we are given a graph G and a partition of its vertices, and then randomly rewire the edges of the graph in such a way that each vertex still has the same degree.

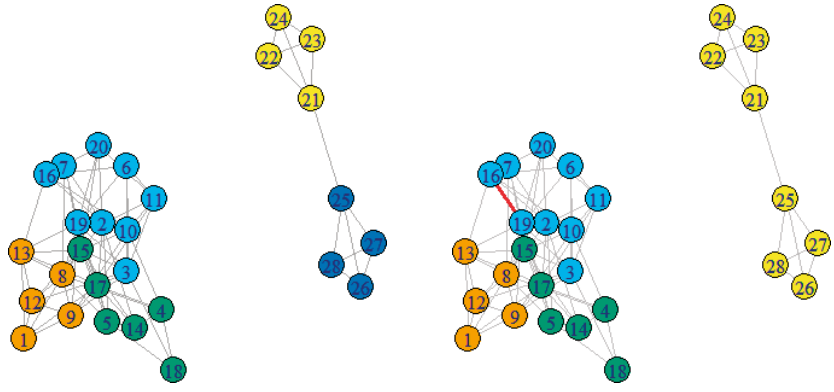
If our partition truly reflected an underlying community structure, we would expect this operation to greatly decrease the number of edges within its parts, since it should have destroyed the community structure of the graph.

A routine calculation shows that the expected change in the number of edges between two distinct vertices u and v is $\frac{d_u d_v}{2m-1} - \mathbb{1}_{uv \in E}$, and so, assuming m is very large and the contribution from loops (that is, the case of $u = v$) negligible, we get that the change in number of edges within parts is approximately

$$\sum_{u,v \in V} \delta_{\mathscr{A}}(u,v) \left(\frac{d_u d_v}{2m} - \mathbb{1}_{uv \in E} \right). \tag{2.2}$$

This similarity between (2.1) and (2.2) thus provides additional justification for regarding modularity as a measure of community structure.

As might be expected of an optimization problem in such a general form, exactly optimizing the modularity is an NP-hard problem [7]. In fact, even finding a multiplicative approximation to the modularity is hard [11].



(a) A graph G, with vertices coloured according to the modularity-maximising partition

(b) G with a single edge added, highlighted in red, and its new optimal partition

Figure 2.1. An illustration of the resolution limit issue with modularity.

Thus, in practice, various heuristic algorithms that run in polynomial time, such as the Louvain [4] or Leiden [33] algorithms, are commonly used, but these give no guarantees on the quality of the found solution.

Unlike rule-based methods such as single-linkage clustering [15], methods like modularity which are based on optimisation of an objective function can exhibit rather unexpected behaviours in certain cases.

Probably the most famous example of this is the *resolution limit*, first identified by Fortunato and Barthélemy [17], where modularity optimization fails to detect communities smaller than the square root of the number of edges in the graph. An example of this is given in Figure 2.1, taken from Paper I — adding a single edge to the graph pushes us over the resolution limit threshold, and makes it so that communities of size four are now merged. The fact that the new edge is in a different connected component than the merged communities makes the result even more counterintuitive.

2.3 Temporal graphs

Graphs are a widely used model for many real-world systems, such as social networks, ecological systems, or the structure of the brain. The abstraction of a graph captures the notions of people being friends, species predating each other, or regions of the brain communicating in one general framework, allowing us to use the same community detection methods to understand social groups, the structure of an ecosystem, or the functional units of the brain.

However, one aspect of reality that this model does not capture is the fact that things change over time. People talk to new people [14], birds migrate and bears hibernate [31], and the brain changes across the endocrine cycle [27].

In order to capture this, we need to define a notion of a *temporal* graph that can vary over time. In our construction, we will assume that time is discrete, not continuous, and that each vertex is present at every time. So, for example, we might have weekly aggregated data on which species has been observed to predate which species during the past week, not the exact timestamps of each predation event.

Definition 2.3.1 (Temporal graph)**.** A *temporal graph* $\mathcal{G} = (G, \lambda)$ with lifetime T consists of a graph $G = (V, E)$ and a function $\lambda : E \to 2^{[T]}$, assigning each edge a set of times at which it is active. For each $t \in [T]$, the *snapshot* of $\mathcal{G}$ at time t is the graph $G_t = (V, E_t)$, where $E_t = \{e \in E \mid t \in \lambda(e)\}$.

Notice that, formally, a temporal graph is a static graph — that is, a graph in the usual sense — equipped with the extra information of at what set of times each edge is present. However, it is often more convenient to think of it as consisting of the sequence of snapshots $G_1, G_2, \ldots, G_T$ — the two perspectives are entirely equivalent.

When studying temporal graphs as a graph theorist, it is often useful to construct an auxiliary graph encoding the entire structure of the temporal graph. We do this by, essentially, stacking all of the snapshots on top of each other, and adding edges between copies of the same vertex in adjacent snapshots. This gives us a single graph with two types of edges: the edges at each time, and the timelike edges between vertices at different times.

Definition 2.3.2 (Time-expanded graph)**.** The *time-expanded graph* of a temporal graph $\mathcal{G} = (G, \lambda)$ is the graph $G^* = (V^*, E^s \cup E^t)$, where $V^* = V \times [T]$ and

$$E^s = \{((u,t),(v,t)) \mid (u,v) \in E, t \in [T]\},$$
$$E^t = \{((u,t),(u,t+1)) \mid u \in V, t \in [T-1]\}.$$

The first set of edges corresponds to the edges in the snapshots, while the second set corresponds to the edges between copies of the same vertex in time-adjacent snapshots.

While this construction does not appear explicitly in Paper III, it underlies the proofs of the approximation results that go into the algorithm of that paper. We turn the problem of optimizing temporal modularity on a temporal graph into a cut problem on the time-expanded graph, with edge weights given by the definition of temporal modularity, and then prove approximation results for that cut problem.

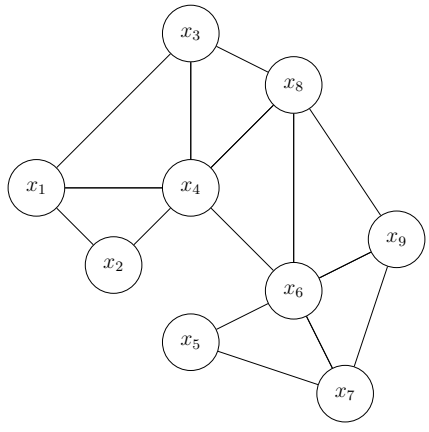


Figure 2.2. The primal graph of the instance in (2.3).

2.4 Fixed-parameter tractability

A canonical hard problem in computer science is 3-SAT: Given a formula such as

$$(x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (x_3 \vee x_4 \vee \neg x_8) \wedge (x_4 \vee \neg x_6 \vee x_8) \\ \wedge (\neg x_5 \vee x_6 \vee x_7) \wedge (x_6 \vee x_8 \vee \neg x_9) \wedge (\neg x_6 \vee \neg x_7 \vee x_9), \tag{2.3}$$

we ask whether it has a satisfying assignment, i.e., an assignment of truth values to the variables $x_1, x_2, \ldots, x_9$ such that the formula evaluates to true.

However, despite being hard in general, it is often solvable efficiently in practice, even for large instances. Explaining this phenomenon is an important question in computer science.

It is known that a *random* instance of 3-SAT with sufficiently few clauses per variable will be satisfiable with high probability [19], and a satisfying assignment for such instances can be found efficiently [9].

To understand this phenomenon, we can consider what kind of structure of the problem might make it easy to solve. Let us begin by considering the *primal graph* of the instance, which is a graph where each variable corresponds to a vertex, and there is an edge between two vertices if they appear together in a clause. For the instance in (2.3), the primal graph is shown in Figure 2.2.

From the figure, it is clear that removing vertices x_4 and x_6 leaves a graph that is a path. On the side of the formula, this corresponds to fixing values for x_4 and x_6. So, if we set both to false, the remaining formula would be

$$(x_1 \vee \neg x_2) \wedge (x_3 \vee \neg x_8) \wedge (\neg x_5 \vee x_7) \wedge (x_8 \vee \neg x_9),$$

which is an instance of 2-SAT, a problem solvable in polynomial time [3].

In this particular instance, we can find a satisfying assignment by setting each of x_1, x_3, x_7, and x_8 to true, and assigning arbitrary values to the remaining

variables. If the formula were unsatisfiable, we would have had to try every combination of truth values for x_4 and x_6, and seen that each resulting instance of 2-SAT was unsatisfiable.

What we have found is called a *strong backdoor set* for this instance — a set of variables such that for any assignment of truth values to the variables in the set, the primal graph of the remaining instance has low treewidth. In our example, the treewidth is 1, since the remaining graph is a path, and all trees have treewidth 1. See Section 2.5 for a definition and discussion of the concept of treewidth.

It is known that if an instance ϕ of 3-SAT has a strong backdoor set of size k, then it can be solved in time $O(2^{O(k)}|\phi|)$ [16]. Thus, for any fixed backdoor set size, the problem can be solved in *polynomial* time.

This is just one instance of a very general phenomenon, which is studied under the name of *fixed-parameter tractability*.

Definition 2.4.1. Suppose that, for each instance of a problem, there is an associated parameter $k \in \mathbb{N}$. If there exists an algorithm that takes an instance of the problem with parameter k and solves it in time $O(f(k)n^c)$, where f is an arbitrary function and c is a constant independent of k, then we say that the problem is *fixed-parameter tractable* with respect to the parameter k.

In Paper III, we study the problem of optimising the temporal modularity of a graph, and show that the problem of *approximating* this optimum is fixed-parameter tractable with respect to the treewidth of the underlying graph.

2.5 Treewidth

Many problems that are hard on general graphs can be solved in a very simple way on trees, by exploiting the tree's natural recursive structure, which allows us to solve the problem separately on each branch and then glue together the partial solutions.

Consider, for example, the problem of *vertex colouring*, where we want to colour the vertices of a graph with the minimum number of colours such that no two adjacent vertices have the same colour. On trees, this problem is trivial: we can colour the root with one colour, its children with a second, their children with the first again, and so on, alternating colours at each level. This gives us a valid colouring with at most two colours.

Graph colouring is NP-hard in general [23], so this leads us to the natural question of finding a parameter which captures the notion of "being like a tree", and which allows us to solve problems like vertex colouring efficiently. The most common such parameter is *treewidth* — intuitively, a graph with low treewidth "looks like a tree if you squint at it".

For vertex colouring, it is easy to see how to find a colouring of a so-called *k-tree*, which is essentially a fattened version of a tree.

Definition 2.5.1 (k-tree)**.** A *k-tree* is a graph which can be constructed in the following way:

- Start with a $k+1$-clique.
- Add new vertices one at a time, each time connecting the new vertex to k vertices of the existing graph, in such a way that the new vertex and the k vertices it connects to form a $k+1$ clique.
- Repeat until all vertices have been added.

For a k-tree, there is a similarly simple algorithm for colouring it using $k+1$ colours:

- Colour the initial $k+1$ vertices with $k+1$ different colours.
- For each subsequent vertex, since it only has k neighbours, there must be one colour which is not used by any of its neighbours. Colour the new vertex with that colour.

It is not too hard to see that if a graph is k-colourable, then any subgraph of it is also k-colourable, simply by using the same colouring. Thus, we have seen that any subgraph of a k-tree is $k+1$-colourable, and have a vague idea of an algorithm for colouring it, by finding a supergraph of it which is a k-tree.

This leads us to the following definition of treewidth:

Definition 2.5.2 (Treewidth [5])**.** The *treewidth* of a graph G is the minimum k such that G is a subgraph of a k-tree.

It turns out that, in practice, this is not the best structure to use for algorithms on graphs of bounded treewidth, since it does not give us a tree structure for the graph. Instead, we use a *tree decomposition* of the graph:

Definition 2.5.3 (Tree decomposition)**.** A *tree decomposition* of a graph $G = (V,E)$ is a tree $T = (I,F)$ together with a set of *bags* X_i, for $i \in I$, such that:

- For each vertex v of G, the set
$$\{i \in I \mid v \in X_i\}$$
of nodes of T whose bag contains v forms a non-empty connected subtree.
- For each edge uv of G, there exists an $i \in I$ such that $u, v \in X_i$.

Definition 2.5.4 (Treewidth via tree decomposition [5])**.** The *treewidth* of a graph G is the minimum across all tree decompositions $T, \{X_i\}_{i \in I}$ of the maximum size of a bag X_i minus one, i.e.

$$\mathrm{tw}(G) = \min_{T=(I,F) \text{ decomposition of } G} \max_{i \in I} |X_i| - 1.$$

The idea of this definition is that, since each bag has bounded size, we can afford to use a brute force method within each bag — for example by considering every possible colouring of the vertices in the bag — and then use the tree structure to combine the results from each bag into a colouring of the entire graph.

This type of algorithm is called a *dynamic programming* algorithm. To illustrate how this works, let us consider the vertex colouring problem again, now as the problem of determining whether a given graph G is j-colourable, for some fixed j.

Before we begin, we need to define a more restricted kind of tree decomposition, called a *nice tree decomposition*, because it simplifies the description of the dynamic programming algorithm. Any tree decomposition can be transformed into a nice tree decomposition in linear time, with only a polynomial increase in the number of bags and no increase in width, so the asymptotic complexity of the algorithm remains unchanged [6].

Definition 2.5.5 (Nice tree decomposition)**.** A *nice tree decomposition* is a tree decomposition $T = (I, F)$, where T is a rooted tree, such that there are four types of nodes:

- *Leaf nodes*: Leaf nodes (including the root) have empty bags, i.e., $X_i = \emptyset$.
- *Introduce nodes*: A node i with a single child j, such that $X_i = X_j \cup \{v\}$ for some vertex $v \notin X_j$.
- *Forget nodes*: A node i with a single child j, such that $X_i = X_j \setminus \{v\}$ for some vertex $v \in X_j$.
- *Join nodes*: A node i with two children j and k, such that $X_i = X_j = X_k$.

This construction allows us to define the dynamic programming algorithm by specifying its behavior at each type of node.

For each $i \in I$, let V_i be the set of vertices in the bag i or bags below it, that is, let

$$V_i = \bigcup_{\substack{i' \in I \\ i' \text{ is a descendant of } i}} X_{i'}.$$

The state of the algorithm is encoded by a function $c[i, S]$, where i is a node in the tree decomposition and $S : X_i \to [j]$ is a j-colouring of the vertices in the bag X_i. We define $c[i, S]$ by

$$c[i,S] = \begin{cases} 1 & \text{if } \exists S' : V_i \to [j] : S'|_{X_i} = S,\ S' \text{ is proper,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Thus, if r is the root, V_r is the entire vertex set, and therefore $c[r, \emptyset]$ is 1 if and only if the entire graph is j-colourable. Meanwhile, if i is a leaf, $V_i = \emptyset$, and trivially $c[i, \emptyset] = 1$. The problem is thus how to recurse over the tree structure to compute the interesting value $c[r, \emptyset]$ at the root using the trivial base cases $c[i, \emptyset]$ at the leaves.

To do this, we need to define what we do on introduce, forget and join nodes:

- *Introduce node*: If i is an introduce node with child ℓ, then we have $X_i = X_\ell \cup \{v\}$ for some vertex $v \notin X_\ell$. We can then compute $c[i,S]$ as

$$c[i,S] = \begin{cases} 0 & \text{if } S \text{ is not proper,} \\ c[j, S|_{X_\ell}] & \text{otherwise.} \end{cases}$$

  This works because if S is a proper colouring of X_i, then it can be extended to a proper colouring of V_i if and only if $S|_{X_\ell}$ can be extended to a proper colouring of V_ℓ, since there is no edge between v and any vertex in $V_\ell \setminus X_\ell$. That no such edge exists follows from the properties of the tree decomposition: If such an edge existed, it would have to be covered by some bag, but any vertex in $V_\ell \setminus X_\ell$ has already been forgotten in X_ℓ, so it cannot be in the bag X_i or any other bag in $T \setminus V_\ell$.
- *Forget node*: If i is a forget node with child j, then we have $X_i = X_\ell \setminus \{v\}$ for some vertex $v \in X_\ell$. Now, for each $\iota \in [j]$, define S_ι as the function which is equal to S on X_i, and which assigns ι to v. Then we can compute $c[i,S]$ as

$$c[i,S] = \begin{cases} 0 & \text{if } S \text{ is not proper,} \\ \max_{\iota \in [j]} c[\ell, S_\iota] & \text{otherwise.} \end{cases}$$

  This works because S can be extended to a proper colouring of V_i precisely when it is proper and can be extended to some S_ι which in turn extends to all V_ℓ.
- *Join node*: If i is a join node with children ℓ_1 and ℓ_2, then we have $X_i = X_{\ell_1} = X_{\ell_2}$. Here, we can simply define $c[i,S] = \min\{c[\ell_1,S], c[\ell_2,S]\}$, since S will extend to all of V_i precisely when it extends to both V_{ℓ_1} and V_{ℓ_2}.

Notice that this algorithm will check if a colouring on a bag is proper many times, but since the size of the bags is bounded, this just gets us a factor of j^k each time. The only dependency on the number of vertices is in the number of bags, which is linear in n, and so we get a total running time of $O\left(n \cdot j^{k+O(1)}\right)$.

In order to be fully rigorous, we would of course need to actually prove that this correctly computes c, not merely provide an intuition for why it will work. This is usually the most involved and technical part of any proof using a dynamic programming algorithm — thus, we omit that in this introduction, since the example is only intended to illustrate the general idea.

In Paper III, we use a more involved dynamic programming algorithm to find a partition of a temporal graph which maximises the temporal modularity. The algorithm is more complicated, needing to keep track of a more complex state, but the basic idea is the same dynamic programming paradigm.

2.6 Category theory

Perhaps the odd topic out in this otherwise combinatorial thesis, category theory is typically viewed as a branch of mathematics closer to algebra than to combinatorics. It studies mathematical structures and the relationships between them in an abstract way, moving from particular collections of mathematical objects to the general notion of a *category* of objects and morphisms.

In Paper I, we use the language of category theory primarily as a generalization of monotonicity, and to provide a natural notion of isomorphism between clustering schemes. To see what we mean by this, let us introduce the basic concepts of category theory and some examples.

Definition 2.6.1. A *category* $\mathscr{C}$ consists of the following:

- A collection of *objects* $\mathrm{ob}\,C$.
- For each pair of objects $A, B \in \mathrm{ob}\,C$, a collection of *morphisms* $\hom(A,B)$ from A to B.
- For each triple of objects $A, B, C \in \mathrm{ob}\,C$, a composition map

  $$\circ_{A,B,C} : \hom(B,C) \times \hom(A,B) \to \hom(A,C)$$

  that is associative in the sense that for any morphisms $f : A \to B$, $g : B \to C$, and $h : C \to D$, we have

  $$h \circ (g \circ f) = (h \circ g) \circ f.$$

- For each object A, an identity morphism $\mathrm{id}_A : A \to A$ such that for any morphism $f : A \to B$, we have $\mathrm{id}_B \circ f = f$ and $f = f \circ \mathrm{id}_A$.

The definition is, of course, very abstract, but in every case we are interested in, the objects will be sets and the morphisms will be functions between them — in the language of the category theorist, they are *concrete* categories. In this case, the composition of morphisms is just the composition of functions, and the identity morphism is the identity function on a set.

The simplest example of a category is the *category of sets*, denoted Set. The objects are sets, and the morphisms are functions between them. The composition of morphisms is just the composition of functions, and the identity morphism on a set A is the identity function $\mathrm{id}_A : A \to A$.

For a more combinatorial example, consider the category Graph, where the objects are graphs and the morphisms are graph homomorphisms. It should be clear that this forms a category, as we can compose graph homomorphisms and the identity morphism on a graph is just the identity function on its vertex set.

The real power of the categorical language does, however, not lie in defining categories themselves, but in the notion of a *functor* between categories, which allows us to translate between different categories while preserving the structure of the categories.

Definition 2.6.2. A *functor* $F : \mathscr{C} \to \mathscr{D}$ between categories $\mathscr{C}$ and $\mathscr{D}$ consists of the following:

- For each object $A \in \operatorname{ob}\mathscr{C}$, an object $F(A) \in \operatorname{ob}\mathscr{D}$.
- For each morphism $f : A \to B$ in $\mathscr{C}$, a morphism $F(f) : F(A) \to F(B)$ in $\mathscr{D}$.

The functor must satisfy the following properties:

- For any object A, we have $F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$.
- For any morphisms $f : A \to B$ and $g : B \to C$, we have $F(g \circ f) = F(g) \circ F(f)$.

Again, although the definition is abstract, it is straightforward to apply in the examples we consider.

For instance, consider the functor $F : \mathrm{Graph} \to \mathrm{Set}$ that maps each graph to its vertex set, and each graph homomorphism to the corresponding function between vertex sets. This is known as the *forgetful functor* from graphs to sets, as it forgets the graph structure and retains only the vertex set. It is straightforward to verify that this is indeed a functor.

To illustrate what we mean by a generalisation of monotonicity, consider the example of the line graph of a graph. Intuitively, it is obvious that the line graph of a graph is *monotone* under addition of edges.

One way to make this precise might be to fix a vertex set V, let $G(V)$ be the set of all graphs on V, and likewise let $G\left(\binom{V}{2}\right)$ be the set of all graphs on the vertex set $\binom{V}{2}$. This then makes the line graph a function $L : G(V) \to G\left(\binom{V}{2}\right)$, which is monotone with respect to the subgraph relation on the two sets of graphs.

However, this is in a certain sense not a natural definition: It requires us to specify the vertex set V in advance, and so would require us to think of all graphs as living in the same universe. For instance, if we have a graph G' that is isomorphic to some $G \in G(V)$, the function L would not be defined on G'.

To a category theorist, this kind of definition that cares about the precise set which makes up an object, instead of the isomorphism class of the object, is called *evil*.

A more natural way of defining this would be to consider the category $\mathrm{Graph}_{\mathrm{inj}}$ of graphs with *injective* graph homomorphisms.[2] We can then define the line graph as a functor $L : \mathrm{Graph}_{\mathrm{inj}} \to \mathrm{Graph}_{\mathrm{inj}}$ that maps each graph to its line graph.

Verifying that this will indeed be a functor is a fruitful exercise for the reader, since it is a simple version of a central construction in Paper I. There, we define

[2] In the categorical language, this is a *wide* subcategory of Graph, where the morphisms are restricted to injective graph homomorphisms. A graph theorist might think of it as analogous to a subgraph induced by the set of edges corresponding to injective morphisms.

a family of notions of line graph for *hypergraphs*, and show that they define functors from the category of hypergraphs to the category of simple graphs.

3. Summary of papers

> As to more than these, my son, beware.
> Of the making of many books there is no end,
> and in much study there is weariness for the flesh.
>
> Ecclesiastes 12:12

3.1 Paper I: A classification of overlapping clustering schemes for hypergraphs

Unlike the other papers in this thesis, this paper adopts a completely different community detection paradigm than the optimisation-based approach of modularity.

This line of research begins with classical single-linkage clustering, as discussed in Section 2.1, where we place two objects in the same part whenever they are sufficiently similar.

The first generalization of this is simple: What if we instead said that we put *three* objects together in a part whenever all three are sufficiently similar? If viewed as a problem on graphs, this is already somewhat non-trivial: Single-linkage clearly reduces to taking the connected components of a graph, but the generalized process can instead be seen as operating in two steps:

1. Given a graph $G = (V, E)$, construct an auxiliary hypergraph $H = (V, F)$ whose hyperedges are precisely the triangles of G.
2. Define the parts of the partition of G as the connected components of H.

With this perspective on the cluster-by-triangles method, we immediately see a natural generalization: given any graph R, we can replace "triangles" in the description above with "copies of R" to obtain a corresponding clustering method. In fact, given any set $\mathfrak{R}$ of graphs, we can look for copies of any member of $\mathfrak{R}$, and get a resulting clustering method.

Given such a set $\mathfrak{R}$, we call the auxiliary graph created in step one $\Phi_{\mathfrak{R}}(G)$, and the resulting clustering method $\Pi_{\mathfrak{R}}$ the *clustering scheme represented by* $\mathfrak{R}$. Two nice properties that any such clustering scheme will have are easy to see:

1. Adding an edge to G can only create new copies of graphs from $\mathfrak{R}$, thereby introducing additional edges in $\Phi_{\mathfrak{R}}(G)$. Since adding edges cannot split connected components, it follows that $\Pi_{\mathfrak{R}}$ is monotone under edge additions.

2. Suppose $p \subseteq V$ is a part of G under $\Pi_{\mathfrak{R}}$, and let $H = G|_p$ be the subgraph of G induced by p. What are the parts of H under $\Pi_{\mathfrak{R}}$?
   The crucial insight here is that, in fact, $\Phi_{\mathfrak{R}}(G|_p) = \Phi_{\mathfrak{R}}(G)|_p$,[1] and so, since p is, by assumption, a connected component of $\Phi_{\mathfrak{R}}(G)$, it follows that $\Phi_{\mathfrak{R}}(G|_p)$ is connected, and thus $\Pi_{\mathfrak{R}}(H)$ consists of a single part.
   What we have seen here is effectively that $\Pi_{\mathfrak{R}}$ is idempotent.

The study of this was initiated by Carlsson and Mémoli [8], who showed that *any* clustering scheme satisfying 1. and 2. must in fact be a representable clustering scheme, in the sense that there exists an $\mathfrak{R}$ such that the clustering scheme is exactly $\Pi_{\mathfrak{R}}$.

In our paper, we extend their result in two directions: First, we observe that everything in the construction applies equally well to hypergraphs, and so by slightly rephrasing the definitions, we extend the framework to the hypergraph setting.

Second, and more significantly, we generalize to partitions with *overlapping* parts, by observing that there are many natural ways to define connected components in a hypergraph. For ordinary graphs, we could give the following somewhat convoluted and circular method for computing the connected components:

1. Given a graph G, compute its line graph $\Lambda(G)$.[2]
2. Compute the connected components of $\Lambda(G)$.
3. For each component of $\Lambda(G)$, find the set of vertices incident to an edge in that component.

That this correctly computes the connected components of G is easy to see. The exact same method applies to hypergraphs, provided we define the line graph as the graph whose vertices are the hyperedges, with an edge between two hyperedges whenever they overlap in at least one vertex.

The trick here is that this definition makes the line graph of a hypergraph an ordinary graph, not a hypergraph, and so we have "lifted" the definition of connected components from the simple graphs to the hypergraphs.

Thus, by shifting the definition of the line graph of a hypergraph, we obtain new notions of connected components. The modification is very simple: Instead of connecting two hyperedges whenever they overlap in at least one vertex, we require that they overlap in at least k vertices. As shown by Figure 3.1, this can result in partitions that overlap, or even have one part strictly contained in another.

With this generalization, we very nearly recover the full classification of the simple graph case from Carlsson and Mémoli. However, the possible presence of parts that are strict subsets of other parts means we obtain only that

[1] At this point, considering a concrete example and seeing why this will be true would be a very fruitful exercise. This is one of those insights that is hard to convey in writing or a figure on paper, but becomes very clear upon *drawing* the figure for yourself.

[2] That is, the graph whose vertices are the edges of G, and where two vertices $e, e' \in V(\Lambda(G)) = E(G)$ are adjacent if and only if there is a vertex $v \in G$ such that e and e' are both incident to v.

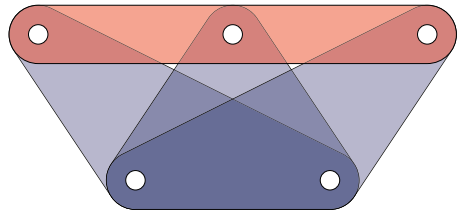


Figure 3.1. An example of a graph G with overlapping connected components for $k = 2$.

any clustering scheme satisfying the two properties must be *isomorphic* to a representable clustering scheme, not necessarily strictly equal.

We even provide an example of a clustering scheme that satisfies the two properties without being equal to a representable scheme, using a slight tweak to the line graph construction. It seems plausible that an even sharper classification could be achieved by finding the right class of line graph constructions for hypergraphs.

In addition to these generalizations, we observe that known results on the algorithmic problem of finding copies of a fixed graph in a sparse graph imply that, for a broad class of graphs, the clustering induced by any representable clustering scheme can be computed in polynomial time. Thus, the full statement of our results is:

Theorem 3.1.1 (Main results)**.** *Any representable clustering scheme is excisive and functorial. Conversely, any excisive and functorial clustering scheme is isomorphic to a representable clustering scheme.*

Further, for any class $\mathscr{C}$ of simple graphs of bounded expansion and any finitely representable clustering scheme $\Pi_{\mathfrak{R},k}$, there exists an algorithm that computes $\Pi_{\mathfrak{R},k}(G)$ in time $O\left(|V(G)|^{\alpha(\mathfrak{R})}\right)$ for any $G \in \mathscr{C}$, where $\alpha(\mathfrak{R})$ is the maximum independence number of any graph in the representing set $\mathfrak{R}$.

3.2 Paper II: Universal lower bound for community structure of sparse graphs

In this paper, we investigate the lowest possible modularity a graph may attain, under mild assumptions on its degree sequence. At first glance, this may seem like the opposite of the question we should be asking — surely in community detection we are interested in graphs with *high* modularity, since those are supposed to be the ones with a community structure?

However, modularity does not come with a natural threshold for determining whether a community structure is present. If we compute the modularity of some graph to be 0.23, does that mean there is or is not an underlying com-

munity structure? We can compare it with other graphs and observe that it has more community structure than a graph with modularity 0.15, but less than one with 0.48, but this alone does not answer the question.

If, however, we knew that *any* graph with the same degree sequence must have modularity at least 0.19, that would provide good reason to say we have found no evidence of a community structure, since there exist graphs with that degree sequence and no community structure, and those would have roughly the same modularity.

Asymptotically, our result can be stated as follows:

Theorem 3.2.1. *Let G be an n-vertex graph with average degree $\bar{d} \geq 1$, $L = \left\{v \in G \;\middle|\; d_v < C\bar{d}\right\}$ for some $C > 1$, and assume that $\mathrm{vol}(L) \geq (1+\gamma)m$ for some $\gamma > 0$.*

If $\Delta(G) = o(n)$ and $\bar{d}^{10} = o(n)$, then

$$q^*(G) \geq \frac{0.26\gamma}{\sqrt{C\bar{d}}}(1+o(1)).$$

The assumption is that there exists some cutoff C such that a majority of the volume of the graph is on vertices whose degree is less than C times the average degree.

The only way to violate this assumption is for the degree sequence to have an extremely heavy tail. Essentially, as soon as the degree sequence has a bounded first moment, our theorem applies.[3] This idea also allows us to prove the following consequence of our theorem:

Theorem 3.2.2. *Let G be a graph with degree sequence $(d_i)_{i\in[n]}$, with average degree $\bar{d}$, satisfying*

$$\frac{1}{n}\left|\{i : d_i \geq k\}\right| \leq A\bar{d}^{\tau-1}k^{1-\tau}$$

for all i, with constants $\tau > 2$ and $A > 0$. For $b = 0.1\left(\frac{\tau-2}{8A}\right)^{\frac{1}{2(\tau-2)}}$ and sufficiently large n,

$$q^*(G) \geq b\bar{d}^{-1/2}.$$

Finally, the paper applies this result to show that preferential attachment graphs have modularity at least $\Omega\left(\bar{d}^{-1/2}\right)$.

The first trick that opens the way to proving our result is the following simple observation about the modularity of two-part partitions:

[3] Of course, we are being imprecise in this statement, since the graphs are not assumed to be random. Heuristically, however, this is true — think about Markov's inequality.

Lemma 3.2.3. *Suppose $G = (V,E)$ is a graph on n vertices with m edges, and $V = A \cup B$ is a two-part partition of G. Then*

$$q(G,\{A,B\}) = \frac{1}{2} - \frac{e(A,B)}{e(G)} - \frac{(\mathrm{vol}(A) - \mathrm{vol}(B))^2}{2\,\mathrm{vol}(G)^2},$$

where $e(A,B)$ is the number of edges between A and B.

This reflects the intuition behind what modularity measures: If there are no cross-edges between parts and the two parts have equal volume, the modularity is 0.5, and any deviation from this is penalised.

If we had the assumption that G were d-regular, then Lemma 3.2.3 would immediately translate any result on bisections of G into a result on its modularity.

So, as a first step towards our result, we want to find a result on bisections of graphs that also gives a bound on the difference in volume between the two sides. For this, we can turn to a result by Alon, which, if one looks at the proof, can be phrased as follows:

Theorem 3.2.4 ([2, 1])**.** *Let $G = ([n],E)$ be a graph, and M be a perfect matching on $[n]$ disjoint from E such that there exist no short loops[4] of G and M. Then there exists a $U \subset [n]$ such that $M \subset U \times U^c$, and*

$$e_G(U,U^c) \leq \frac{1}{2}\sum_{i=1}^{n}\left(\frac{d_i}{2} - \frac{3}{8\sqrt{2}}\sqrt{d_i}\right). \tag{3.1}$$

In the original proof, Alon constructs a matching in a fairly straightforward way, since he only needs the absence of short loops. We, however, can use our construction of the matching M to ensure additional properties of the partition U, U^c.

Our construction gives a matching in which nearly all vertices are matched with others of nearly equal degree. This ensures that the degree sum on each side — that is, the volume — is approximately equal, thus, as observed, giving us a bound on the modularity.

However, our construction of this matching requires the maximum degree of the graph to be at most $n^{1/9}$, so this does not yet yield the full generality of our result.

[4] A technical requirement for their probabilistic method-based proof to go through, since it gives control over the independence or correlation of certain random variables:

Definition 3.2.5. Given a graph $G = ([n],E(G))$ and a matching M disjoint from $E(G)$, a *short loop* of G and M is a loop of length at most twelve containing between one and three edges from M and never more than three consecutive edges of G.

The final step is to observe that the modularity function is not sensitive to small changes in the underlying graph, so removing a few high-degree vertices does not significantly affect its value.

This is where our assumption about a cutoff C comes into play — the set of vertices with degree at most $C\bar{d}$ satisfies the maximum degree assumption, and so we can find an appropriate matching for this bulk of the graph. This gives us the main term of our bound, the $\Omega\left(\bar{d}^{-1/2}\right)$ bound. We can then handle the remaining high-degree vertices by random assignment, and show that the resulting loss is of lower order than the main contribution from the bulk.

3.3 Paper III: Approximating temporal modularity on graphs of small underlying treewidth

The problem of optimizing modularity is well known to be hard, as we discussed in the background section. Accordingly, when generalizing modularity to the setting of temporal graphs, that optimization problem is also hard.

However, there are known fixed-parameter tractability results for optimizing modularity on static graphs: it is tractable when parameterized by the max leaf number [18] and by the vertex cover number [26]. Additionally, computing an optimal partition into at most k parts is tractable when parameterized by treewidth [26].

This latter result yields an FPT algorithm for approximating the optimal modularity, thanks to the following result by Dinh and Thai:

Lemma 3.3.1 ([12])**.** *For any graph G and any positive integer k, let $q_k^*(G)$ denote the maximum modularity of a partition on G into at most k parts. Then it holds that*

$$q_k^*(G) \geq q^*(G)\left(1 - \frac{1}{k}\right).$$

In this paper, we extend the previous result on approximating the static modularity of graphs with low treewidth to the setting of temporal graphs.

We show that the Dinh–Thai bound also holds in the temporal case, and so, as in the static setting, we may restrict ourselves to partitions into at most k parts. This is a crucial step for our dynamic programming algorithm over the tree decomposition to work.

However, in the temporal setting, we encounter another snag: If the graph were allowed to have arbitrarily long lifetime, the size of the state our dynamic program would need to maintain would grow exponentially with the lifetime, as it would need to store every vertex assignment at each time.

Thus, we must also prove an approximation result showing that we can approximate the modularity of a graph with long lifetime by cutting it into

pieces of short lifetime, finding their optimal partitions, and stitching those back together.

Lemma 3.3.2. *Let $\mathscr{G}$ be a temporal graph with lifetime T, and let d be a positive integer. Then there exist integers $0 = t_0 < t_1 < \cdots < t_\ell = T$ such that $t_j - t_{j-1} \leq 2d$ for each $1 \leq j \leq \ell$, and*

$$\sum_{i=0}^{\ell-1} \widetilde{q}(\mathscr{G}_{[t_i+1,t_{i+1}]}) \geq \left(1 - \frac{1}{d}\right) \widetilde{q}^*(\mathscr{G}),$$

where $\widetilde{q}(\mathscr{G})$ denotes non-normalised modularity.[5]

With this lemma and a simple dynamic programming algorithm over time to compute the times t_i, we can turn an algorithm for computing the exact optimum k-part temporal modularity of a graph with short lifetime into an approximation algorithm for any graph.

Having these restrictions on the graphs we consider allows us to use a fairly standard dynamic programming algorithm to derive the following result:

Theorem 3.3.3. *Let $\mathscr{G}$ be a temporal graph with n vertices, lifetime T and underlying treewidth at most w. The maximum temporal modularity of a partition of $\mathscr{G}$ in at most k parts can be calculated in time $n^{O(kT)} k^{O(wT)} T^{O(c)}$.*

Applying the lemmas discussed above, this yields the main theorem of the paper:

Theorem 3.3.4. *Let $\mathscr{G}$ be a temporal graph with n vertices and lifetime T, and suppose that the underlying graph of $\mathscr{G}$ has treewidth at most w. Let c and d be positive integers. Then we can compute a $\left(1 - \left(\frac{1}{c} + \frac{2}{d}\right)\right)$-approximation to $q^*_\omega(\mathscr{G})$ in time $n^{O(cd)} c^{O(wT)} T^{O(c)}$.*

[5]This is purely for convenience in phrasing the result — had we included the normalisation, the sum would have been a weighted average of its terms. Computing the normalised and non-normalised modularity are equivalent problems.

4. Sammanfattning på svenska

Ty detta är det härliga hos mänskan
att hon kan fatta tingens inre väsen,
ej vad de *synas*, men vad de *betyda*;
och verkligheten, vart vårt öga ser,
den är symbolen endast av ett högre.

Epilog vid magisterpromotionen i Lund 1820, Esaias Tegnér

Antag att du har en lista av alla vetenskapliga artiklar som någonsin publicerats, till och med dem i denna avhandling, och informationen om vilka som citerar vilka. Kan du, baserat på enbart denna data, avgöra vilka olika forskningsfält det finns? Eller antag att du har en lista av olika områden i din hjärna, och hur mycket varje område kommunicerar med varje annat — kan du avgöra vad det finns för funktionella enheter i din hjärna?

Den gemensamma nämnaren i dessa olika frågor kallas för gruppindelning av grafer. I varje av fallen tänker vi oss att vi har en graf med en underliggande sann indelning i grupper, såsom olika forskningsområden eller enheter i hjärnan, och vi förväntar oss att vi har betydligt fler kanter inom varje grupp än mellan grupperna.

Hade vi kunnat förvänta oss att det inte fanns några kanter alls mellan grupperna vore problemet mycket enkelt: Vi hade helt enkelt beräknat de sammanhängande komponenterna till grafen, och det hade löst problemet.

I verkligheten händer det dock ibland att en sociolog citerar en biologs verk, så det krävs mer sofistikerade metoder för att särskilja de olika grupperna.

Studiet av detta problem är den gemensamma nämnaren för de olika artiklarna i denna avhandling. Den allra vanligaste metoden i praktiken för att lösa detta problem är *modularitetsmaximering*, i vilken vi definierar en kvalitetsfunktion för noduppdelningar av en graf och försöker hitta den uppdelning som maximerar denna kvalitetsfunktion.

Ifall $G = (V, E)$ är en graf med n noder och m kanter, och $\mathscr{A} = \{A_1, \ldots, A_k\}$ är en uppdelning av noderna till V, så ges *modulariteten* av $\mathscr{A}$ av

$$q(G, \mathscr{A}) = \sum_{i=1}^{k} \frac{e(A_i)}{m} - \frac{\mathrm{vol}(A_i)^2}{(2m)^2},$$

där $e(A_i)$ är antalet kanter inom A_i, och $\mathrm{vol}(A_i)$ är *volymen* av A_i, det vill säga summan av graderna av noderna i A_i. Modulariteten *av en graf* är den högsta modularitet som uppnås av någon uppdelning av dess noder.

I ARTIKEL II studerar vi problemet att ge en nedre gräns till modulariteten av en graf, enbart i termer av dess gradföljd, alltså samlingen av grader hos dess noder.

Denna fråga är intressant eftersom modulariteten inte naturligt har något mått på vad som är en signifikant modularitet — vi kan alltså avgöra att en viss graf har modularitet 0.19, men det säger oss inte nödvändigtvis något om ifall det finns en sann gruppstruktur i grafen, eller ifall uppdelningen som ger det värdet bara har passat brus i datan.

Att finna en nedre begränsing för modulariteten ger oss alltså ett sätt att åtminstone ana att ett uppmätt modularitetsvärde inte är signifikant — ifall alla grafer med samma gradföljd har en modularitet på minst 0.183 är troligen inte ett värde på 0.19 en meningsfull indikation på en sann gruppstruktur i grafen.

Vårt resultat är att varje graf, under milda antaganden om fördelningen av dess grader som säkerställer att svansen inte är extremt tung, har modularitet minst $O\left(\bar{d}^{-1/2}\right)$, där $\bar{d}$ är dess genomsnittsgrad.

I ARTIKEL III studerar vi också modulariteten, men nu i generaliserad form för *dynamiska* grafer — det vill säga grafer utrustade med en tidsstruktur, där varje kant endast är närvarande vid vissa tider. Dessa grafer modellerar alltså data som utvecklas över tiden.

Redan att exakt optimera modulariteten hos en statisk graf är NP-svårt [7], så att optimera den dynamiska modulariteten är *a fortiori* också svårt. Således är vi intresserade av att finna parametrar hos detta problem som gör att vi kan lösa problemet effektivt, alltså i polynomiell tid, ifall parametern är liten. Detta kallas att problemet är FPT (från den engelska termen *fixed-parameter tractability*) med avseende på den givna parametern.

Det är känt att problemet att approximera modulariteten hos en graf inom en viss fast multiplikativ faktor är FPT med avseende på *trädvidden* för statiska grafer [26], vilket ledde oss till att bevisa ett motsvarande resultat även för dynamiska grafer.

Om man enbart optimerar över uppdelningar i högst k delar, och antar att den dynamiska grafens livslängd är högst T, kan den dynamiska modulariteten beräknas exakt för grafer av låg trädvidd i polynomiell tid — detta kan visas relativt enkelt med en dynamisk programmerings-algoritm.[1]

Från den mer kombinatoriska sidan behöver vi alltså finna approximationsresultat som möjliggör approximation av den dynamiska modulariteten hos en godtycklig graf genom modulariteten över högst k delar hos en samling grafer med begränsad livslängd.

[1] Ordet 'dynamisk' i 'dynamisk programmering' har ingen koppling till 'dynamisk' i 'dynamisk graf', utan är enbart ett oturligt språkligt sammanträffande. På engelska heter de 'dynamic programming' respektive 'temporal graph', men om vi inte önskar börja tala om 'tidsliga grafer' får vi acceptera detta sammanträffande.

Vi visar dessa approximationsresultat och finner att den dynamiska modulariteten kan approximeras inom en faktor $1 - \frac{1}{k} - \frac{2}{d}$ genom att beräkna modulariteten i högst k delar hos grafer med livslängd högst d. Detta innebär i sin tur att vi kan beräkna en approximation av den dynamiska modulariteten inom samma faktor för en dynamisk graf med underliggande trädvidd högst w på tid $n^{\mathcal{O}(cd)}c^{\mathcal{O}(wT)}T^{\mathcal{O}(c)}$, så att approximationen är FPT med avseende på den underliggande grafens trädvidd.

I ARTIKEL I studerar vi slutligen en helt annan metod för gruppindelning av grafer, som inte bygger på optimering av en kvalitetsfunktion utan i stället använder en enkel regel för när två noder ska tillhöra samma grupp.

Denna familj av metoder bygger på vad som sannolikt är den allra första gruppindelningsmetoden för data för vilken man har information om likheten mellan varje par av element, nämligen metoden med enkel länkning. I denna metod bestämmer vi helt enkelt att två objekt tillhör samma grupp om deras likhet överstiger ett visst tröskelvärde.

Metoden har därefter generaliserats brett genom att välja mer sofistikerade regler för när två objekt ska tillhöra samma grupp. Den enklaste generaliseringen är att säga att *tre* objekt ska tillhöra samma grupp om de bildar en triangel — det vill säga om alla tre är tillräckligt lika varandra.

Detta kan ses som att vi delar in i grupper baserat på följande process:

1. Givet en graf $G = (V, E)$ bildar vi en hjälp-hypergraf $H = (V, F)$, vars kanter är trianglarna i G.
2. Grupperna i G är de sammanhängande komponenterna i H.

Genom att ersätta trianglarna i denna definition med andra grafer, eller med samlingar av grafer, får vi en hel familj av gruppindelningsmetoder.

Denna familj studerades av Carlsson och Mémoli [8], som klassificerade den genom att visa att den utgör precis de gruppindelningsmetoder som har följande två naturliga egenskaper:

1. *Funktorialitet:* Metoderna är monotona under tillägg av kanter och noder, i betydelsen att tillägg av en kant i en graf enbart kan slå ihop två grupper — aldrig splittra en grupp eller flytta en nod mellan grupper.
2. *Excisivitet:* Metoderna är idempotenta: Om man tillämpar metoden på G och identifierar en grupp H, och man därefter tillämpar metoden på $G[H]$, delgrafen inducerad av H, så kommer den att identifiera hela $G[H]$ som en enda grupp.

I vår artikel generaliserar vi deras resultat på två sätt:

1. Vi studerar gruppindelningsmetoder för hypergrafer, i stället för enbart för ordinära grafer.
2. Vi studerar indelningar i grupper som tillåts överlappa varandra, i stället för att kräva att alla grupper är disjunkta.

Vi finner väsentligen samma klassificering som Carlsson och Mémoli, förutom att möjligheten att ha en grupp som är en strikt delgrupp till en annan resul-

terar i att vi enbart får att alla funktoriella och excisiva gruppindelningsmetoder är *isomorfa* med en metod i vår familj, istället för att vara lika med någon av dem.

Vi visar också att kända algoritmer för att finna kopior av en fix graf i en gles graf möjliggör beräkning av dessa gruppindelningar i polynomiell tid för en bred klass av tillräckligt glesa grafer.

5. Acknowledgements

O God, protector of those who hope in you,
without whom nothing has firm foundation, nothing is holy,
bestow in abundance your mercy upon us
and grant that, with you as our ruler and guide,
we may use the good things that pass
in such a way as to hold fast even now to those that ever endure.

Collect for the Seventeenth Week in Ordinary Time

Through Christ, and with Christ, and in Christ, all honour and glory belongs to the Lord, for ever and ever.

Yet, of course, it often pleases Him to use created things for His Glory, and so it behooves us to thank not only Him, but also all the people He was gracious enough to work through, whether they were deeply aware of His working or not.

Even just in the natural order of created causes and effects, no thesis is ever the product of just one man or woman's intelligence and effort — and my effort and intelligence have certainly never seemed or felt sufficient for the task now completed.

A large part of the credit for my getting to this point must, of course, go to my supervisor, Fiona Skerman, for keeping me on track — or perhaps more accurately for keeping me on the right tracks. Left to my own devices, I would have been on twenty tracks, having progressed on each only a few meters before the engine gave out.

Likewise, mention must be made of my fellow PhD students at the department, who turned a solitary pursuit of research into, if not a less lonely task, at least a task that could be complained about to understanding ears. People say you ought not complain about things, and that it is always a vice — those people have clearly never managed to write a three-page proof that ended with the line $1 = 3.45$, and been unable to find the error after weeks of effort.

When research is frustrating, another consolation can be in teaching,[1] and so an acknowledgement is due to the department for entrusting me with lecturing two different courses, and even more to the students whose enthusiasm for the subject inspired me when my own was lacking. A particular mention should be made of the student whose bachelor's thesis I supervised, Tim: If nothing

[1]The opposite is also true: When you have a pile of 250 exams to grade, suddenly research looks much more appealing.

else, if I had not had the opportunity to talk about mathematics in Swedish for so many hours, writing the Swedish summary of this thesis would have been much harder.

Even if mathematical research can make one feel as if they were a brain on a stick, concerned purely with logical questions disconnected from any external reality, that is of course not true — and the life we lead in reality is ultimately much more important than the abstract pondering we do during working hours.

The first mention when we move from life as a mathematician to life as a human must of course be where that human life began: In the family. My dad has not failed to be supportive at every step, and while mum only got to see me make it to my bachelor's degree, I am certain she would not be surprised at all to see I made it all the way here. Since I am sure a catalogue of one's parents influences on one's life would amount to not much less than an entire autobiography, I will not attempt one here.

Having started in my first home and family, we move on to a second, larger one. It is no exaggeration — in fact, almost an understatement — to say that my life outside of work has largely been lived at St. Lars. There are far too many people there to thank, and yet I must try: Without them I would have been a better mathematician and a much worse human.

Beginning in chronological order, the first mentioned must in fact be outside of the parish, and outside of Sweden: In the lonely days of Covid, the live-streamed Masses and homilies of the Dominican friars in Edinburgh rekindled a flame that had been weak and dormant for long. The torch that lit all the world on fire reached even onto the internet, and thence to me.

When the darkness of Covid receded, and I proceeded out into reality, Fr. Jan provided my welcome into the life of the Church. For that, it is fitting that he be mentioned first.

After him, the flood of names becomes overwhelming. Dominique, for organizing the student group in those early days, and later for our work together in planning and organizing. One of my earliest memories from the student group is being impressed by how you spoke of the faith in such a natural and unabashed way — you were a great witness to the possibility of faith, without any awareness or plan of being so. Your unfailing kindness and willingness to help remain inspirational.

Fr. Fredrik, for your work with the student group, but also for the guidance and exercises through the years. All the others who were in the student group at that time: Edward and Ebonie, who will likely never see this acknowledgment, especially for that time dinner at Alt Berlin turned into a Bible study.

Eric, and Max, and Sean, and many more who have been around since then: Seeing all of you through these years, with the twists and turns of your lives, is a reminder (all too rarely thought of) of the grandness of the Lord's providential plan, and the smallness of my place within it. I am grateful to have gotten to play a bit part and spectator role in your stories.

Gabriel, for being a solid and supportive friend. I must admit that when I first met you, I thought you were a bit much — why did this person seemingly like me so well for so little good reason? Well, some angel kindled a little bit of virtue in me, and made me see past my chilly Nordic distaste for enthusiasm, and that has paid off a hundred times over.

Speaking of much enthusiasm, and of speaking much, Magnus also deserves a mention, if nothing else for repeatedly belying my claim of not speaking German. The number of times I have said "please don't ask me for translations into Swedish, I don't really speak German, but anyway the Swedish translation of that is 'bosättare' " would make a wiser man than I concede his knowledge of German.

Mention should be made jointly of Fabian, Virginia, and Katharina, not only for last summer but for being such central parts of our community at that time. Yet I would be remiss were I not to mention each individually as well.

Fabian, for providing much of both serious intellectualism and of silliness — I know he would dispute the latter, but anything that involves people dressed in white tie objectively has an aspect of silliness. A true student is a false aristocrat, to paraphrase St. Augustine.

When I first invited Virginia along that summer, it was — shameful to admit — perhaps mostly as Katharina's friend. She thus joins the list of people I have initially underestimated, and although it was for entirely the opposite reason as why I did not at first appreciate Gabriel, the reason I now appreciate them is very similar: For being good and reliable friends, supportive in bad times and good company in good times.

It is no understatement to say that, of all my friends past and current, Katharina is one of those whose words and actions God has used the most in leading me to where I am today in my relationship with Him. It is likewise true that she is probably the one who will best understand the sense of this: Where others might be offended at seemingly being reduced to tools of God's providence, her spiritual sight is unusually clear, and so I have no doubt that she will be pleased to be seen to have been of use to her Lord.[2]

In more recent times, thanks is also due to Fr. Thomas, for all his work as pastor but most directly for opening up for new catechists in the parish. Getting to teach, and learn from, those children once every other Saturday has been a blessing in every sense of the word, and often the brightest spot in those weeks. Here, Ines deserves particular mention as not just a good fellow-catechist but a good fellow-Christian in times of need and of joy.

On a similar topic of things that give meaning to a life bogged down by abstractions, but chronologically much earlier, the people at Mikaelsgården deserve particular mention. Janne and Elina, of course, and all the staff and fellow volunteers, but even more so the guests, who for obvious reasons must

[2]It sincerely baffles me how often He shows His Providence in her life — so it is surely only fitting that she sometimes also be a tool of said Providence in the lives of others.

remain unnamed but are certainly not forgotten. When, on many days, the question of what good I did at work looms large, being able to, on some days, say I gave a plate of food to a hungry person and some water to a thirsty person is a great consolation.

Continuing on with the seemingly interminable list of fellow students,[3] Sam, for being a social nexus who brings many together, and a fellow PhD student who can appreciate the struggle.

As the evangelist wrote, if all the things Jesus did were written down, the world itself could not contain all the books that would be written. This is true not only of His works on Earth when walking among us, but also of His works in each of our lives. Thus, in the interest of leaving some space in the world, I must let many names be mentioned in passing, even if their presence in my life was not necessarily passing:

Adam the seminarian and Adam the theologian. Alex. Anna the Finnish, and Anna the French. Bernadette, Christoffer, Cicci, Dylan, Erik, Frida, Gabriel[4], Gustav, Hannah, Ines[4] and Inez, Jesaja, Josef, Judit, Katharina[4], Liina and Lucas[5], Marcus and Markus, Marta, Mickaël and Mikael, Nikki[6], Oliver, Pettar, Sam[4], Fr. Thomas[4], Virginia[4], and Yu. May this litany of friends one day be a litany of saints.

Finally, having procrastinated writing my acknowledgment of you until the very end, Miriam. If anyone I know knows the travails of working towards a doctorate, it is certainly you. It is fitting that you finished writing your thesis a bit before me, because in nearly everything (other than height), you are a bit ahead of me.

I have said it before, but you are certainly the person whose opinions and ideas I respect the most, and who has made me change my stubborn mind about the most things. Much more importantly than that intellectual level, you also keep inspiring me by your faith and example to be better. Your presence in my life has been a blessing.

So, to all the people mentioned and all the people I ought to have mentioned but forgot in my weakness: May God bless you in all that you do, and continue to glorify His Name through your lives.

Vilhelm Agdur,

on the feast of Pope Silverius,
in the year of Our Lord MMXXV.

[3] What a blessing it is to be able to complain about having had too many friends to mention and acknowledge!

[4] The same one again — you get a second mention to fill the alphabetical order.

[5] How fitting that you occur together also in alphabetical order — although you would have deserved a paragraph of your own.

[6] Is it cheating to mention you again under a nickname to fill the letter *N*?

References

[1] N. Alon, P. Hamburger, and A. Kostochka. Regular honest graphs, isoperimetric numbers, and bisection of weighted graphs. *European Journal of Combinatorics*, 20(6):469–481, 1999.

[2] Noga Alon. On the edge-expansion of graphs. *Combinatorics, Probability and Computing*, 6(2):145–152, 1997.

[3] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[4] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[5] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[6] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.

[7] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[8] Gunnar Carlsson and Facundo Mémoli. Classifying clustering schemes. *Foundations of Computational Mathematics*, 13(2):221–252, 2013.

[9] Ming-Te Chao and John Franco. Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k-satisfiability problem. *Information Sciences*, 51(3):289–314, 1986.

[10] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiway cuts. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, STOC 1992*, Proceedings of the Annual ACM Symposium on Theory of Computing, pages 241–251. Association for Computing Machinery, July 1992.

[11] Thang N Dinh, Xiang Li, and My T Thai. Network clustering via maximizing modularity: Approximation algorithms and theoretical limits. In *2015 IEEE International Conference on Data Mining*, pages 101–110. IEEE, 2015.

[12] Thang N. Dinh and My T. Thai. Finding community structure with performance guarantees in scale-free networks. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 888–891, 2011.

[13] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[14] Bojan Evkoski, Andraž Pelicon, Igor Mozetič, Nikola Ljubešić, and Petra Kralj Novak. Retweet communities reveal the main sources of hate speech. *Plos one*, 17(3), 2022.

[15] K. Florek, J. Łukaszewicz, J. Perkal, Hugo Steinhaus, and S. Zubrzycki. Sur la liaison et la division des points d'un ensemble fini. *Colloquium Mathematicum*, 2(3-4):282–285, 1951.

[16] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, M. S. Ramanujan, and Saket Saurabh. Solving d-SAT via backdoors to small treewidth. In *Proceedings of the 26th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 630–641, San Diego, CA, USA, 2015. SIAM.

[17] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, January 2007.

[18] Jaroslav Garvardt and Christian Komusiewicz. Modularity clustering parameterized by max leaf number. In *19th International Symposium on Parameterized and Exact Computation (IPEC 2024)*, pages 16–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

[19] Svante Janson, Yiannis C. Stamatiou, and Malvina Vamvakari. Bounding the unsatisfiability threshold of random 3-SAT. *Random Structures & Algorithms*, 17(2):103–116, 2000.

[20] N. Jardine and R. Sibson. *Mathematical Taxonomy*. Wiley series in probability and mathematical statistics. Wiley, 1971.

[21] Muhammad Aqib Javed, Muhammad Shahzad Younis, Siddique Latif, Junaid Qadir, and Adeel Baig. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*, 108:87–111, 2018.

[22] L. R. Ford Jr. and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[23] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[24] Eun Jung Kim, Tomáš Masařík, Marcin Pilipczuk, Roohani Sharma, and Magnus Wahlström. On weighted graph separation problems and flow augmentation. *SIAM Journal on Discrete Mathematics*, 38(1):170–189, 2024.

[25] S. Thomas McCormick, M.R. Rao, and Giovanni Rinaldi. Easy and difficult objective functions for max cut. *Mathematical Programming*, 94(2–3):459–466, Jan 2003.

[26] Kitty Meeks and Fiona Skerman. The parameterised complexity of computing the maximum modularity of a graph. *Algorithmica*, 82(8):2174–2199, 2020.

[27] Joshua M. Mueller, Laura Pritschet, Tyler Santander, Caitlin M. Taylor, Scott T. Grafton, Emily Goard Jacobs, and Jean M. Carlson. Dynamic community detection reveals transient reorganization of functional brain networks across a female menstrual cycle. *Network Neuroscience*, 5(1):125–144, 2021.

[28] Michelle G. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), February 2004.

[29] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, Dec 1991.

[30] Tiago P. Peixoto. Hierarchical block structures and high-resolution model selection in large networks. *Physical Review X*, 4(1), March 2014.

[31] Shai Pilosof, Mason A Porter, Mercedes Pascual, and Sonia Kéfi. The multilayer nature of ecological networks. *Nature Ecology & Evolution*, 1(4), 2017.

[32] Martin Rosvall, Daniel Axelsson, and Carl T. Bergstrom. The map equation. *European Physical Journal-special Topics*, 178(1):13–23, November 2009.
[33] Vincent A. Traag, Ludo Waltman, and Nees Jan van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9:5233, 2019.

UPPSALA DISSERTATIONS IN MATHEMATICS
Dissertations at the Department of Mathematics
Uppsala University

---

1. Torbjörn Lundh: *Kleinian groups and thin sets at the boundary*. 1995.
2. Jan Rudander: O*n the first occurrence of a given pattern in a semi-Markov process*. 1996.
3. Alexander Shumakovitch: *Strangeness and invariants of finite degree*. 1996
4. Stefan Halvarsson: *Duality in convexity theory applied to growth problems in complex analysis*. 1996.
5. Stefan Svanberg: *Random walk in random environment and mixing*. 1997.
6. Jerk Matero: *Nonlinear elliptic problems with boundary blow-up*. 1997.
7. Jens Blanck: *Computability on topological spaces by effective domain representations*. 1997.
8. Jonas Avelin: *Differential calculus for multifunctions and nonsmooth functions*. 1997.
9. Hans Garmo: *Random railways and cycles in random regular graphs*. 1998.
10. Vladimir Tchernov: *Arnold-type invariants of curves and wave fronts on surfaces*. 1998.
11. Warwick Tucker: *The Lorenz attractor exists*. 1998.
12. Tobias Ekholm: *Immersions and their self intersections*. 1998.
13. Håkan Ljung: *Semi Markov chain Monte Carlo*. 1999.
14. Pontus Andersson: *Random tournaments and random circuits*. 1999.
15. Anders Andersson: *Lindström quantifiers and higher-order notions on finite structures*. 1999.
16. Marko Djordjević: *Stability theory in finite variable logic*. 2000.
17. Andreas Strömbergsson: *Studies in the analytic and spectral theory of automorphic forms*. 2001.
18. Olof-Petter Östlund: *Invariants of knot diagrams and diagrammatic knot invariants*. 2001.
19. Stefan Israelsson: *Asymptotics of random matrices and matrix valued processes*. 2001.
20. Yacin Ameur: *Interpolation of Hilbert spaces*. 2001.
21. Björn Ivarsson: *Regularity and boundary behavior of solutions to complex Monge-Ampère equations*. 2002.
22. Lars Larsson-Cohn: *Gaussian structures and orthogonal polynomials*. 2002.
23. Sara Maad: *Critical point theory with applications to semilinear problems without compactness*. 2002.
24. Staffan Rodhe: *Matematikens utveckling i Sverige fram till 1731*. 2002.
25. Thomas Ernst: *A new method for q-calculus*. 2002.
26. Leo Larsson: *Carlson type inequalities and their applications*. 2003.

27. Tsehaye K. Araaya: *The symmetric Meixner-Pollaczek polynomials*. 2003.
28. Jörgen Olsén: *Stochastic modeling and simulation of the TCP protocol*. 2003.
29. Gustaf Strandell: *Linear and non-linear deformations of stochastic processes*. 2003.
30. Jonas Eliasson: *Ultrasheaves*. 2003.
31. Magnus Jacobsson: *Khovanov homology and link cobordisms*. 2003.
32. Peter Sunehag: *Interpolation of subcouples, new results and applications*. 2003.
33. Raimundas Gaigalas: *A non-Gaussian limit process with long-range dependence*. 2004.
34. Robert Parviainen: *Connectivity Properties of Archimedean and Laves Lattices*. 2004.
35. Qi Guo: *Minkowski Measure of Asymmetry and Minkowski Distance for Convex Bodies*. 2004.
36. Kibret Negussie Sigstam: *Optimization and Estimation of Solutions of Riccati Equations*. 2004.
37. Maciej Mroczkowski: *Projective Links and Their Invariants*. 2004.
38. Erik Ekström: *Selected Problems in Financial Mathematics*. 2004.
39. Fredrik Strömberg: *Computational Aspects of Maass Waveforms*. 2005.
40. Ingrid Lönnstedt: *Empirical Bayes Methods for DNA Microarray Data*. 2005.
41. Tomas Edlund: *Pluripolar sets and pluripolar hulls*. 2005.
42. Göran Hamrin: *Effective Domains and Admissible Domain Representations*. 2005.
43. Ola Weistrand: *Global Shape Description of Digital Objects*. 2005.
44. Kidane Asrat Ghebreamlak: *Analysis of Algorithms for Combinatorial Auctions and Related Problems*. 2005.
45. Jonatan Eriksson: *On the pricing equations of some path-dependent options*. 2006.
46. Björn Selander: *Arithmetic of three-point covers*. 2007.
47. Anders Pelander: *A Study of Smooth Functions and Differential Equations on Fractals*. 2007.
48. Anders Frisk: *On Stratified Algebras and Lie Superalgebras*. 2007.
49. Johan Prytz: *Speaking of Geometry*. 2007.
50. Fredrik Dahlgren: *Effective Distribution Theory*. 2007.
51. Helen Avelin: *Computations of automorphic functions on Fuchsian groups*. 2007.
52. Alice Lesser: *Optimal and Hereditarily Optimal Realizations of Metric Spaces*. 2007.
53. Johanna Pejlare: *On Axioms and Images in the History of Mathematics*. 2007.
54. Erik Melin: *Digital Geometry and Khalimsky Spaces*. 2008.
55. Bodil Svennblad: *On Estimating Topology and Divergence Times in Phylogenetics*. 2008.

56. Martin Herschend: *On the Clebsch-Gordan problem for quiver representations*. 2008.
57. Pierre Bäcklund: *Studies on boundary values of eigenfunctions on spaces of constant negative curvature*. 2008.
58. Kristi Kuljus: *Rank Estimation in Elliptical Models*. 2008.
59. Johan Kåhrström: *Tensor products on Category O and Kostant's problem*. 2008.
60. Johan G. Granström: *Reference and Computation in Intuitionistic Type Theory*. 2008.
61. Henrik Wanntorp: *Optimal Stopping and Model Robustness in Mathematical Finance*. 2008.
62. Erik Darpö: *Problems in the classification theory of non-associative simple algebras*. 2009.
63. Niclas Petersson: *The Maximum Displacement for Linear Probing Hashing*. 2009.
64. Kajsa Bråting: *Studies in the Conceptual Development of Mathematical Analysis*. 2009.
65. Hania Uscka-Wehlou: *Digital lines, Sturmian words, and continued fractions*. 2009.
66. Tomas Johnson: *Computer-aided computation of Abelian integrals and robust normal forms*. 2009.
67. Cecilia Holmgren: *Split Trees, Cuttings and Explosions*. 2010.
68. Anders Södergren: *Asymptotic Problems on Homogeneous Spaces*. 2010.
69. Henrik Renlund: *Recursive Methods in Urn Models and First-Passage Percolation*. 2011.
70. Mattias Enstedt: *Selected Topics in Partial Differential Equations*. 2011.
71. Anton Hedin: *Contributions to Pointfree Topology and Apartness Spaces*. 2011.
72. Johan Björklund: *Knots and Surfaces in Real Algebraic and Contact Geometry*. 2011.
73. Saeid Amiri: *On the Application of the Bootstrap*. 2011.
74. Olov Wilander: *On Constructive Sets and Partial Structures*. 2011.
75. Fredrik Jonsson: *Self-Normalized Sums and Directional Conclusions*. 2011.
76. Oswald Fogelklou: *Computer-Assisted Proofs and Other Methods for Problems Regarding Nonlinear Differential Equations*. 2012.
77. Georgios Dimitroglou Rizell: *Surgeries on Legendrian Submanifolds*. 2012.
78. Boris Granovskiy: *Modeling Collective DecisionMaking in Animal Groups*. 2012
79. Mahmoudreza Bazarganzadeh: *Free Boundary Problems of Obstacle Type, a Numerical and Theoretical Study*. 2012
80. Qi Ma: *Reinforcement in Biology*. 2012.
81. Isac Hedén: *Ga-actions on Complex Affine Threefolds*. 2013.
82. Daniel Strömbom: *Attraction Based Models of Collective Motion*. 2013.
83. Bing Lu: *Calibration: Optimality and Financial Mathematics*. 2013.

84. Jimmy Kungsman: *Resonance of Dirac Operators*. 2014.
85. Måns Thulin: *On Confidence Intervals and Two-Sided Hypothesis Testing*. 2014.
86. Maik Görgens: *Gaussian Bridges – Modeling and Inference*. 2014.
87. Marcus Olofsson: *Optimal Switching Problems and Related Equations*. 2015.
88. Seidon Alsaody: *A Categorical Study of Composition Algebras via Group Actions and Triality*. 2015.
89. Håkan Persson: *Studies of the Boundary Behaviour of Functions Related to Partial Differential Equations and Several Complex Variables*. 2015.
90. Djalal Mirmohades: *N-complexes and Categorification*. 2015.
91. Shyam Ranganathan: *Non-linear dynamic modelling for panel data in the social sciences*. 2015.
92. Cecilia Karlsson: *Orienting Moduli Spaces of Flow Trees for Symplectic Field Theory*. 2016.
93. Olow Sande: *Boundary Estimates for Solutions to Parabolic Equations*. 2016.
94. Jonathan Nilsson: *Simple Modules over Lie Algebras*. 2016.
95. Marta Leniec: *Information and Default Risk in Financial Valuation*. 2016.
96. Arianna Bottinelli: *Modelling collective movement and transport network formation in living systems*. 2016.
97. Katja Gabrysch: *On Directed Random Graphs and Greedy Walks on Point Processes*. 2016.
98. Martin Vannestål: *Optimal timing decisions in financial markets*. 2017.
99. Natalia Zabzina: *Mathematical modelling approach to collective decision-making*. 2017.
100. Hannah Dyrssen: *Valuation and Optimal Strategies in Markets Experiencing Shocks*. 2017.
101. Juozas Vaicenavicius: *Optimal Sequential Decisions in Hidden-State Models*. 2017.
102. Love Forsberg: *Semigroups, multisemigroups and representations*. 2017.
103. Anna Belova: *Computational dynamics – real and complex*. 2017.
104. Ove Ahlman: *Limit Laws, Homogenizable Structures and Their Connections*. 2018.
105. Erik Thörnblad: *Degrees in Random Graphs and Tournament Limits*. 2018.
106. Yu Liu: *Modelling Evolution. From non-life, to life, to a variety of life*. 2018.
107. Samuel Charles Edwards: *Some applications of representation theory in homogeneous dynamics and automorphic functions*. 2018.
108. Jakob Zimmermann: *Classification of simple transitive 2-representations*. 2018.
109. Azza Alghamdi: *Approximation of pluricomplex Green functions – A probabilistic approach*. 2018.
110. Brendan Frisk Dubsky: *Structure and representations of certain classes of infinite-dimensional algebras*. 2018.
111. Björn R. H. Blomqvist: *Gaussian process models of social change*. 2018.

112. Konstantinos Tsougkas: *Combinatorial and analytical problems for fractals and their graph approximations*. 2019.
113. Yevgen Ryeznik: *Optimal adaptive designs and adaptive randomization techniques for clinical trials*. 2019.
114. Andrea Pasquali: *Constructions in higher-dimensional Auslander-Reiten theory*. 2019.
115. Daniah Tahir: *Multi-trait Branching Models with Applications to Species Evolution*. 2019.
116. Andreas Rocén: *7D supersymmetric Yang-Mills theory on toric and hypertoric manifolds*. 2019.
117. Filipe Mussini: *Selected Topics in Continuum Percolation. Phase Transitions, Cover Times and Random Fractals*. 2019.
118. Laertis Vaso: *Constructions of n-cluster tilting subcategories using representation-directed algebras.* 2020.
119. Anders Israelsson: *Regularity Properties of Oscillatory Integral Operators on Function Spaces.* 2020.
120. Johan Asplund: *Chekanov-Eliashberg dg-algebras and partially wrapped Floer cohomology*. 2021.
121. Elin Persson Westin: *Stratified algebras and classification of tilting modules*. 2022.
122. Marcus Westerberg: *Prostate cancer incidence, treatment and mortality. Empirical longitudinal register-based studies and methods for handling missing data*. 2022.
123. Dan Lilja: *Properties of invariant sets in certain two-dimensional dynamical systems. Renormalization and beyond*. 2022.
124. Oleksandra Gasanova: *On properties of monomial ideals and algebras.* 2022.
125. Colin Desmarais: *Decorating trees grown in urns*. 2022
126. Malte Litsgård: *On parabolic equations of Kolmogorov-Fokker-Planck type*. 2023
127. Carmina Fjellström: *Selected Topics in Mathematical Modelling: Machine Learning and Tugs-of-War*. 2023.
128. Robert Moscrop: *Quivers, superconformal field theories and higher symmetries.* 2023.
129. Markus Thuresson: *Ext-algebras of standard modules over quasi-hereditary algebras*. 2023.
130. Aksel Bergfeldt: *Regularity of multilinear oscillatory integral operators and nonlinear dispersive partial differential equations*. 2023.
131. Fabian Burghart: *Building and Destroying Urns, Graphs, and Trees.* 2023
132. Yuqiong Wang: *Optimal stopping, incomplete information, and stochastic games.* 2023.
133. Helena Jonsson: *Structure and representations of bimodule categories*. 2023.

134. Christoffer Olsson: *Symmetry in Randomness. Additive Functionals and Symmetries of Random Trees and Tree-Like Graphs*. 2023.

135. Linnéa Gyllingberg: *The Art of Modelling Oscillations and Feedback across Biological Scales.* 2024.

136. Joel Dahne: *Computer Assisted Studies in Fluid Mechanics and Spectral Geometry.* 2024.

137. Lukas Nakamura: *Skein-valued Gromov-Witten theory and Hofer geometry in contact manifolds*. 2024

138. Axel Husin: *Generalized local systems and the Maslov class of exact Lagrangians.* 2025

139. Rebekka Müller-Widmann: *Probabilistic Models of Genetic Variability in Sequence Evolution.* 2025

140. Mingyi Hou: *Behind the training dynamics of neural networks. Analysis of Fokker-Planck equations and the path to metastability*. 2025

141. Mateusz Stroiński: *Module categories in the absence of adjunctions.* 2025

142. Elias Riedel Gårding: *Aspects of Generalised Symmetries of Four-dimensional Field Theories.* 2025

143. Darius Dramburg: *Higher representation infinite algebras from skew-group algebras. Higher preprojective gradings, Koszul gradings, and McKay quivers.* 2025

144. Vilhelm Agdur: *Community Structure in Graphs. Algorithms and Theoretical Guarantees.* 2025