Graphs and networks for clustering and fitting: graph structure

(Harbin, July 2025)

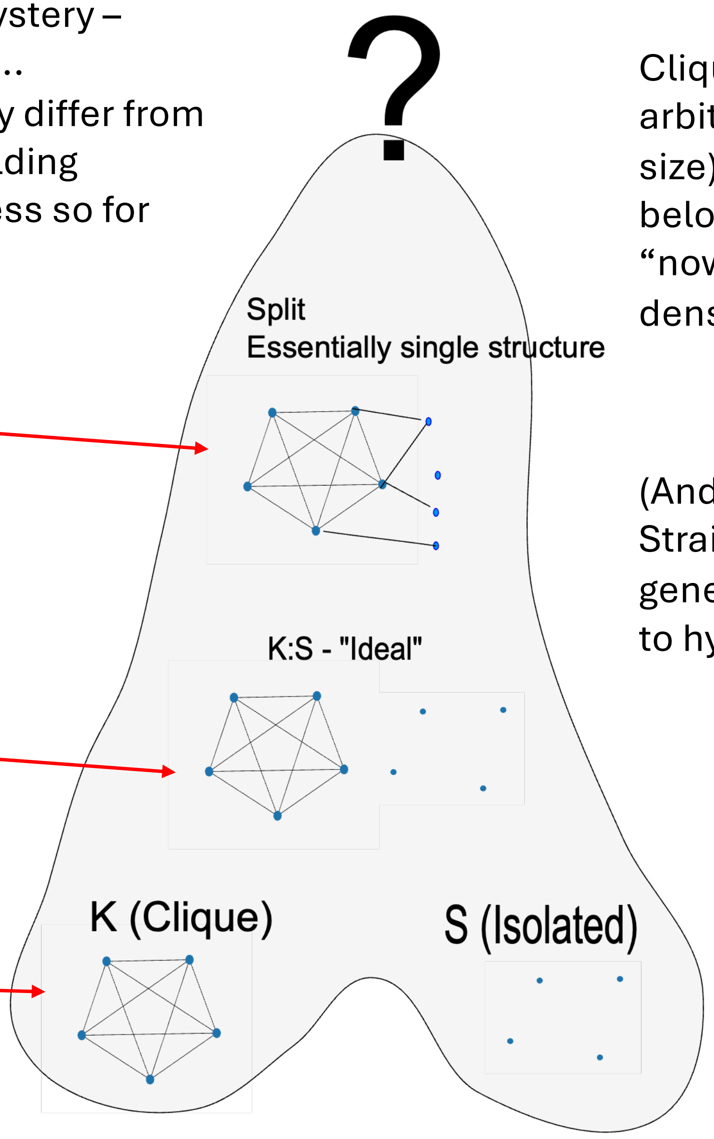
https://ai-ecu.github.io/HIT/index_SoC.html

Much above this....somewhat a mystery – especially in the hypergraph cases..
AND the relevant "actors" will likely differ from problem type to problem type. Building blocks/base common...less and less so for classes "up the tree"

"not ideal" but still relatively ideal and still single structure with some background – and only some "accidental" or minor associations between the two.

Realistic – but "ideal" data is a mix of structure and "background" and the structure is "pure" and the background "pure" and no interaction/association between structure and background

"Building Blocks"
By themselves model only the "trivial" situations:
Perfect – structure and nothing else/no structure and nothing else



Cliques (of arbitrary large size) cannot belong to "nowhere dense" classes.

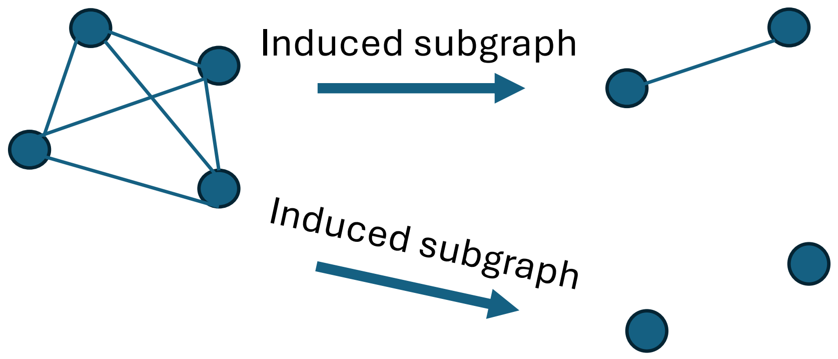
(And Straightforward generalizations to hypergraphs)

Graph classes

- Family of graphs – can be characterized not by what they do contain…but also by what they do not contain (forbidden subgraphs).
- For our purposes, there are two types of forbidden subgrapghs
  - Induced subgraphs (”what you get if you delete vertices and the edges attached to them”
  - Subgraphs (non-induced) (“what you get if you delete vertices and the edges that are attached to them AND you are permitted to delete further edges)
  - Hard to remember? Well, as we shall see, if your graph comes from data – in a very natural sense an induced subgraph is actually *exactly* what corresponds to taking a subset of the data and the graph that would be present on that subset. Whereas (non-induced) subgraphs are not so natural (taking a subset of data should not change the relationships/edges between the remaining data!)
- Not all graph classes can be defined by forbidden *induced* subgraphs (but many of the interesting ones can!). Those classes that can are called Hereditary families – we say they are “closed” under taking induced subgraphs – every induced subgraph we produce must also be a member of the family because otherwise we have produced a forbidden (induced) subgraph and therefore that wouldn’t be forbidden ☺

Graph classes – forbidden induced subgraphs

- So – if the graph class is “complete” (cliques) – what is forbidden?
  - Well every pair of vertices must have an edge (that’s the definition) so if we delete all but two vertices (remaining) that (induced) subgraph can’t be two vertices without an edge.

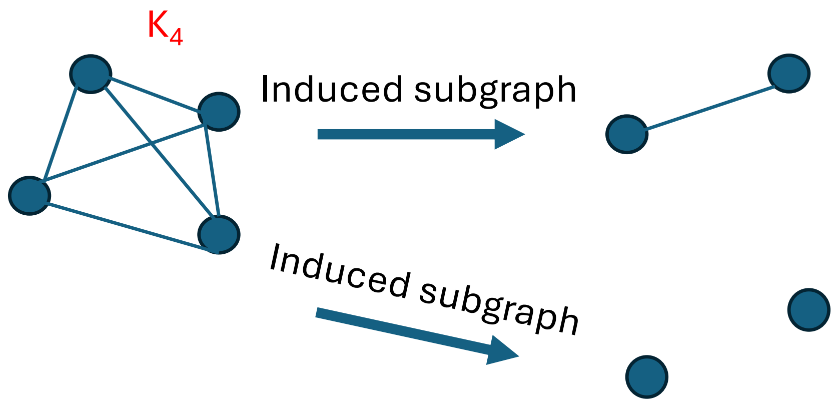


Yes! – deleting any two vertices in any order will produce this subgraph

No! No matter which two vertices you delete you will be left with two vertices that have an edge between them. Every graph you produce ”on the way” will always be a clique

Graph classes – forbidden induced subgraphs

- Some naming and Symbols to describe common “patterns” (subgraphs)

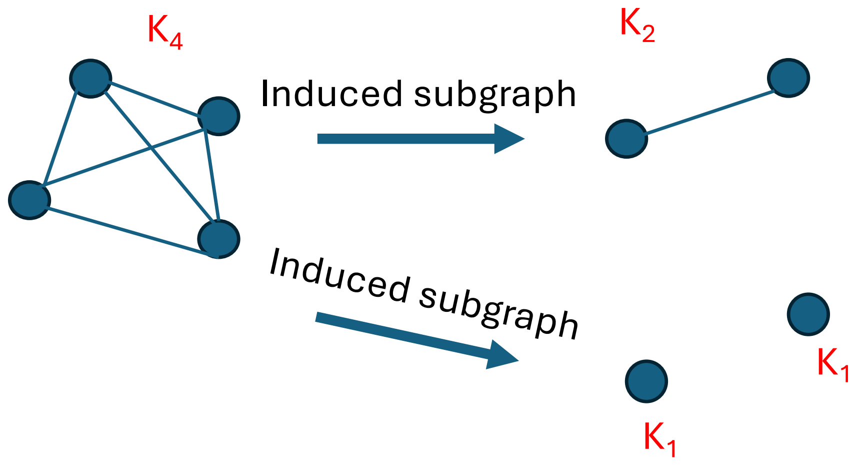


Yes! – deleting any two vertices in any order will produce this subgraph

No! No matter which two vertices you delete you will be left with two vertices that have an edge between them. Every graph you produce ”on the way” will always be a clique

Graph classes – forbidden induced subgraphs

- Some naming and Symbols to describe common “patterns” (subgraphs)

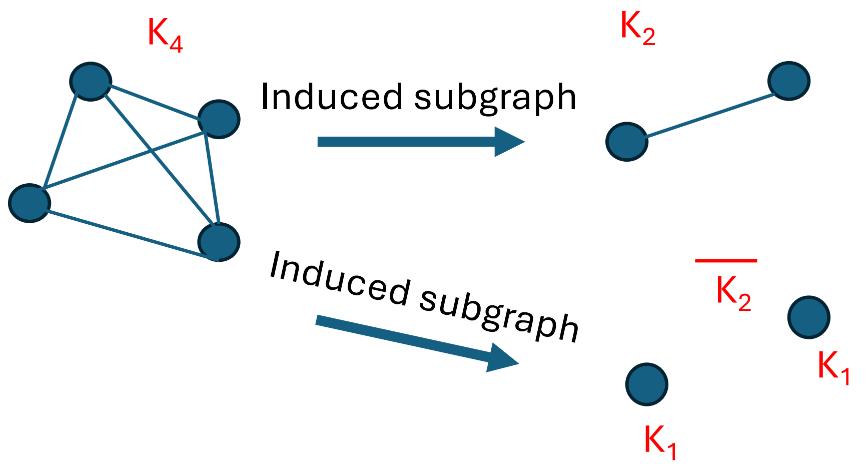


Yes! – deleting any two vertices in any order will produce this subgraph

No! No matter which two vertices you delete you will be left with two vertices that have an edge between them. Every graph you produce ”on the way” will always be a clique

Graph classes – forbidden induced subgraphs

- Some naming and Symbols to describe common “patterns” (subgraphs)



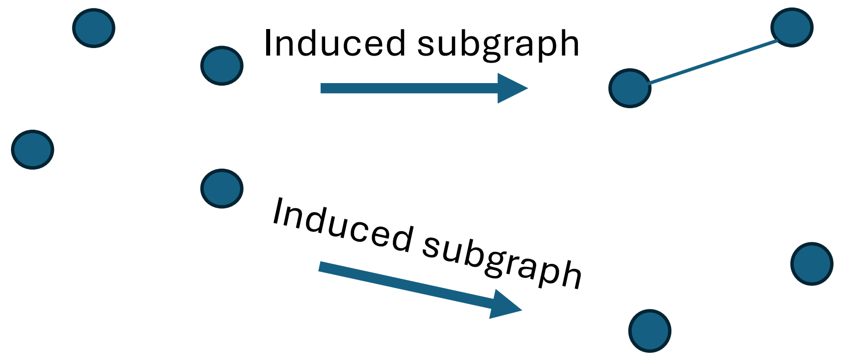
Yes! – deleting any two vertices in any order will produce this subgraph

$2K_1$
Disjoint union of $2\ K_1$

No! No matter which two vertices you delete you will be left with two vertices that have an edge between them. Every graph you produce ”on the way” will always be a clique

Graph classes – forbidden induced subgraphs

- So – if the graph class is "empty" (isolated vertices – empty subgraph) – what is forbidden?
    - Well every pair of vertices cannot have an edge (that's the definition) so if we delete all but two vertices (remaining) that (induced) subgraph must be two vertices without an edge.



No! – deleting any two vertices in any order will produce and empty subgraph

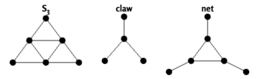
YES! No matter which two vertices you delete you will be left with two vertices that do not have an edge between them. Every graph you produce "on the way" will always be empty

Or the "easy" answer. Since this is the complement of cliques – then the forbidden subgraphs are the complements of the forbidden subgraphs of cliques!

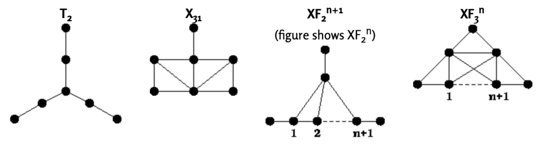
Graph classes – forbidden induced subgraphs

- Now you should easily see that there are other “larger” forbidden subgraphs (a clique cannot contain any “empty subgraph” of ANY size) but this is redundant information. We need only define the *smallest* forbidden subgraphs
- The two simple classes we have looked at only had one smallest forbidden subgraph – all other forbidden subgraphs contain these forbidden subgraphs and so are already forbidden by the smallest one.
- Most families have a SET of smallest forbidden subgraphs – it can get quite tricky to establish these – luckily smart people have spent a lot of time working these out!

Empty/$\overline{K}_n$	Complete/ Clique K_n	$\overline{K}_2$ or $(2K_1)$	K_2	Constant speed (linear with n)
$\overline{\text{Complete Split}}$ (Ideal)	Complete Split	$(2K_2,P_3)$	$(C_4,\overline{P_3})$	Exponential <factorial
threshold		$(2K_2,C_4,P_4)$		factorial
split		$(2K_2,C_4,C_5)$		Super factorial
Quasi-threshold		(C_4,P_4)		factorial
Unit Interval		$\mathbf{(C_{n+4},S_3,claw,net)}$		**factorial**
Interval		$\mathbf{(C_{n+4},T_2,X_{31},XF_2^{n+1},XF_3^{n})}$		**factorial**

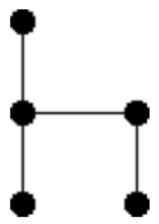


Split-interval: note T_2 is not a split graph – so automatically excluded by being split, C_{n+4} , X_{31} , fork are also not split, (but net is), XF^{n+1}_2 is not split for n>1, S_3 and rising sun are split, but XF^n_3 is not split for n>1
So…split-interval are the graphs that exclude $(2K_2,C_4,C_5,S_3$,rising sun, net)
Interval-threshold – this intersection is just threshold – as can be seen by the fact that all excluded induced subgraphs of interval are not threshold graphs – so already excluded.

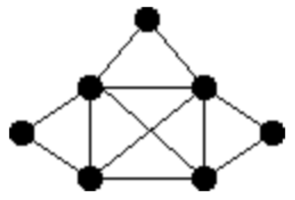


XF^0_2=fork, XF^1_2=net
$XF^0_3=S_3$, XF^1_3=rising sun

fork = chair g6: DiC



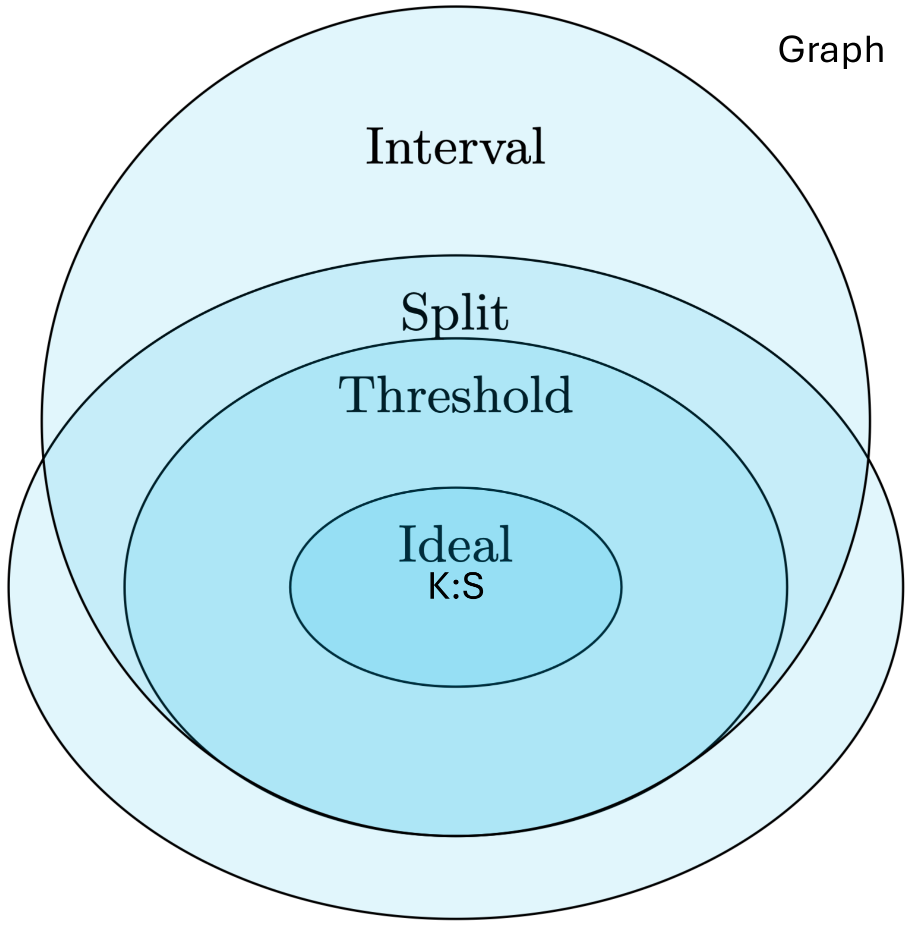
rising sun g6: F~p`_



Why does all of this matter?

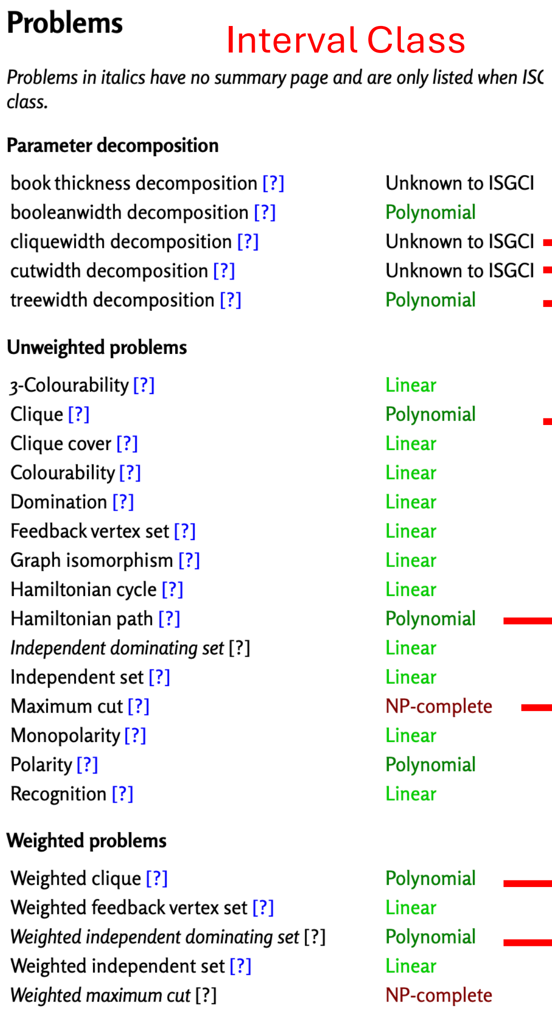
- If you know one graph class X is contained in another Y, then X is "special" compared to Y. It has more properties. It has all the properties that Y has plus some more.
- These extra properties may be exploited by algorithms – making them either simpler, faster, or both…
- Of course it depends not just on the graph class but also on the problem you want to solve on those graph classes
- But even there, a smaller (more special/simpler) graph class will often be "nice" for MANY problems.

Complexity characterization



Complexity characterization – why does this matter?

Interval Class



Problems

Problems in italics have no summary page and are only listed when ISC class.

Parameter decomposition

book thickness decomposition [?]	Unknown to ISGCI
booleanwidth decomposition [?]	Polynomial
cliquewidth decomposition [?]	Unknown to ISGCI
cutwidth decomposition [?]	Unknown to ISGCI
treewidth decomposition [?]	Polynomial

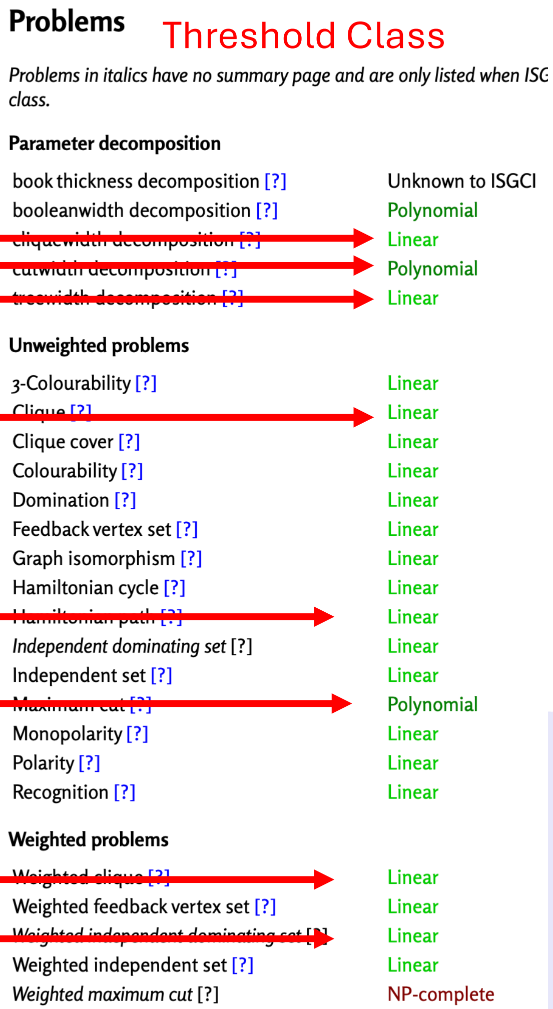
Unweighted problems

3-Colourability [?]	Linear
Clique [?]	Polynomial
Clique cover [?]	Linear
Colourability [?]	Linear
Domination [?]	Linear
Feedback vertex set [?]	Linear
Graph isomorphism [?]	Linear
Hamiltonian cycle [?]	Linear
Hamiltonian path [?]	Polynomial
Independent dominating set [?]	Linear
Independent set [?]	Linear
Maximum cut [?]	NP-complete
Monopolarity [?]	Linear
Polarity [?]	Polynomial
Recognition [?]	Linear

Weighted problems

Weighted clique [?]	Polynomial
Weighted feedback vertex set [?]	Linear
Weighted independent dominating set [?]	Polynomial
Weighted independent set [?]	Linear
Weighted maximum cut [?]	NP-complete

Threshold Class



Problems

Problems in italics have no summary page and are only listed when ISG class.

Parameter decomposition

book thickness decomposition [?]	Unknown to ISGCI
booleanwidth decomposition [?]	Polynomial
cliquewidth decomposition [?]	Linear
cutwidth decomposition [?]	Polynomial
treewidth decomposition [?]	Linear

Unweighted problems

3-Colourability [?]	Linear
Clique [?]	Linear
Clique cover [?]	Linear
Colourability [?]	Linear
Domination [?]	Linear
Feedback vertex set [?]	Linear
Graph isomorphism [?]	Linear
Hamiltonian cycle [?]	Linear
Hamiltonian path [?]	Linear
Independent dominating set [?]	Linear
Independent set [?]	Linear
Maximum cut [?]	Polynomial
Monopolarity [?]	Linear
Polarity [?]	Linear
Recognition [?]	Linear

Weighted problems

Weighted clique [?]	Linear
Weighted feedback vertex set [?]	Linear
Weighted independent dominating set [?]	Linear
Weighted independent set [?]	Linear
Weighted maximum cut [?]	NP-complete

Ideal Class (K:S) (Complement of Split)

Many problems trivial – any greedy algorithm finds max clique, max independent set…

Graph classes – Other ways to characterise

- We just briefly looked at defining by forbidden induced subgraph
- We also looked at more “text” – descriptive ways
- There are also ways by defining what are called “graph” parameters
  - Technically they are an assignment to any graph as “number” and the graph classes are described by either the value of this number (for whole graph class) or by the “growth” of that number as the size of the graph grows (ie. As a factor of N – the number of vertices in the graph
  - Often of particular interest is whether the graph parameter is bounded for the given family/class
  - Just as there is a ”zoo” of graph classes – there are a zoo of “graph parameters”

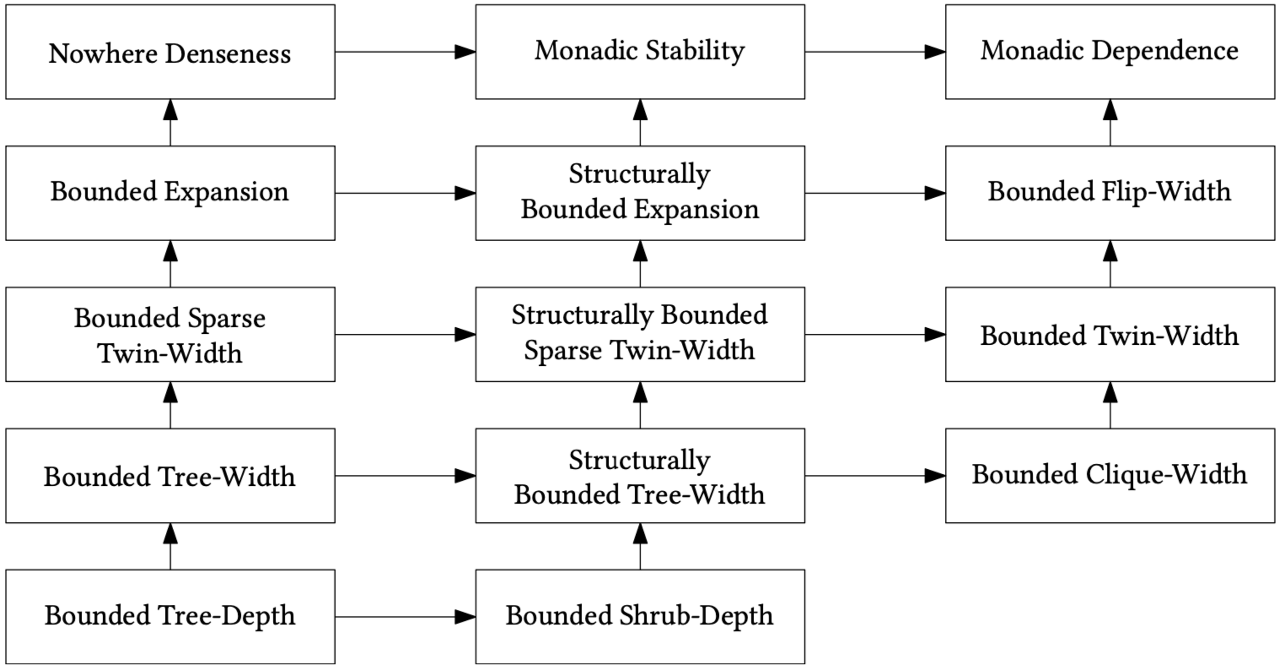


Figure 17.1: A hierarchy of the class properties discussed in this thesis.

Monadically Stable and Monadically Dependent Graph Classes

Characterizations and Algorithmic Meta-Theorems

Nikolas Mählmann

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)

1. Gutachter: Prof. Dr. Sebastian Siebertz
2. Gutachter: Dr. habil. Patrice Ossona de Mendez

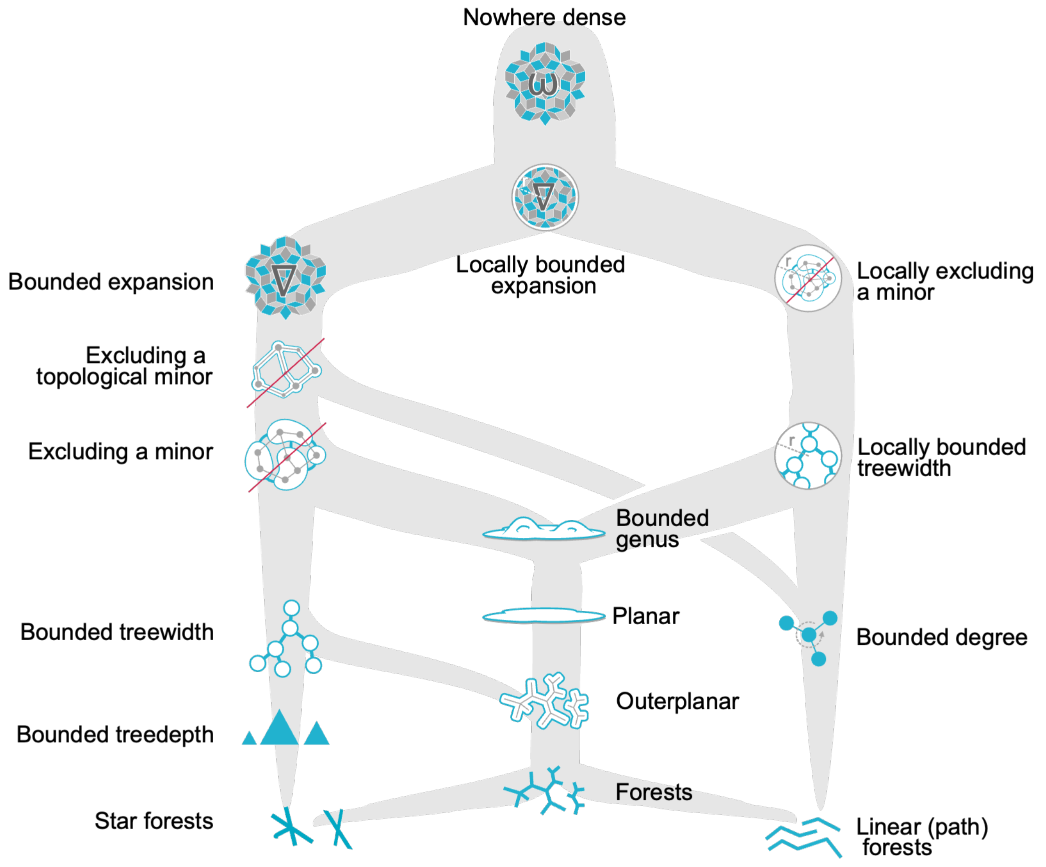
Datum der Verteidigung: 5. September 2024

Universität Bremen
Fachbereich 3 - Mathematik und Informatik

How do these views of graph classes relate? (And relate to broad classes of problems and broad classes of algorithms?)

- Complexity of solving a problem (on some data – some instance of a graph class) depends on
  - How complex you graph class is – how complex your data is – how non-special it is….
  - How complex your problem is (which in turn is related in some quantitiave ways to complex it is to describe your problem – like is it describable in First Order Logic.

Hierarchy of “complexity”/niceness – for `Sparse’ graphs



Slightly modified from diagram of Felix Reidl – circa 2020

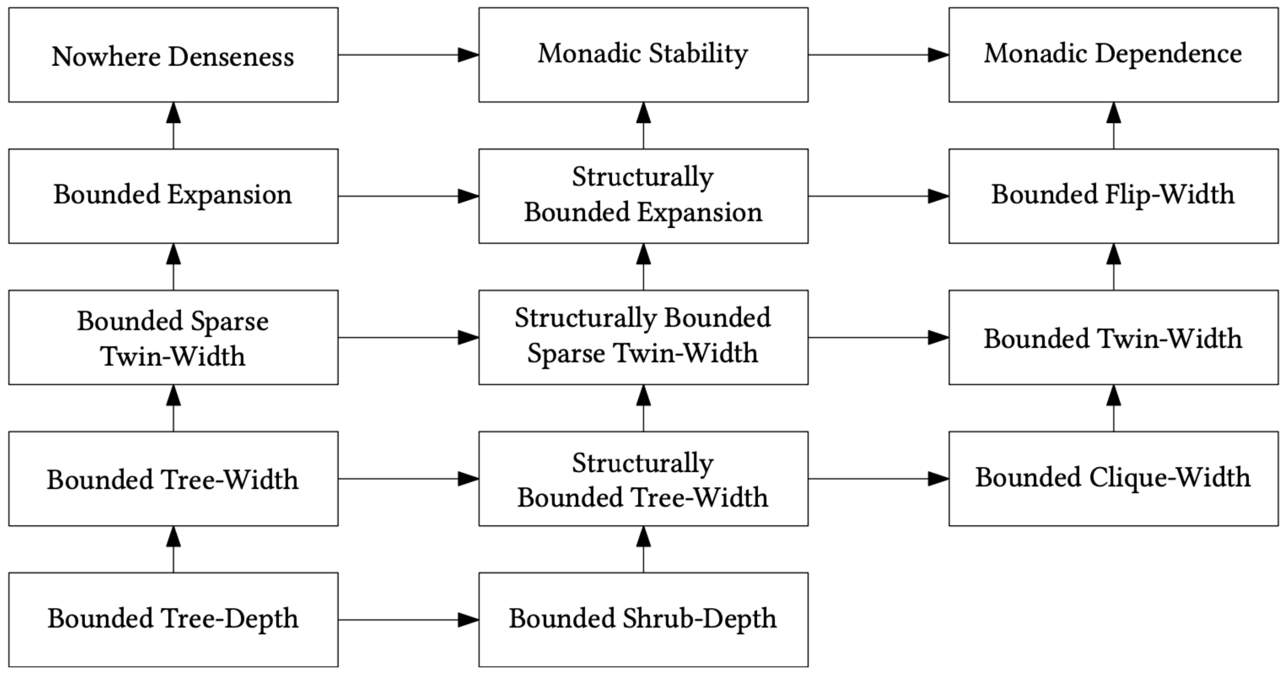
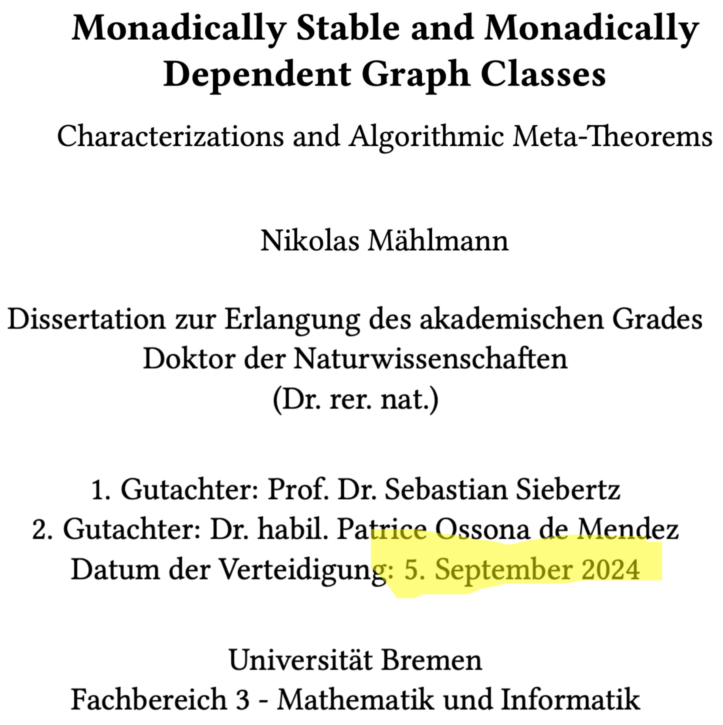


Figure 17.1: A hierarchy of the class properties discussed in this thesis.



Monadically Stable and Monadically Dependent Graph Classes

Characterizations and Algorithmic Meta-Theorems

Nikolas Mählmann

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)

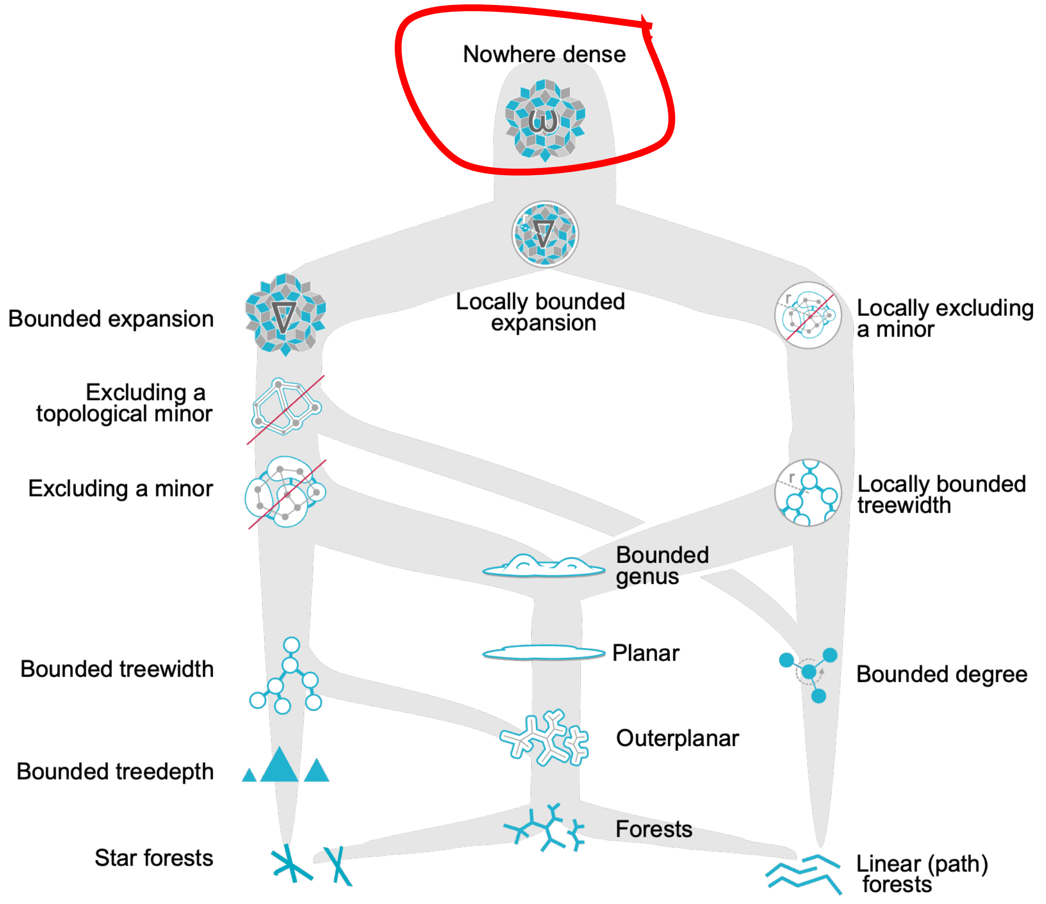
1. Gutachter: Prof. Dr. Sebastian Siebertz
2. Gutachter: Dr. habil. Patrice Ossona de Mendez

Datum der Verteidigung: 5. September 2024

Universität Bremen
Fachbereich 3 - Mathematik und Informatik

Hierarchy essentially in terms of graph parameters – though the top left two are in terms of graph properties that relate FOL descriptions of graph problems and graph classes for which FPT algorithms exist for problems definable in FOL

Hierarchy of “complexity”/niceness – for `Sparse’ graphs



Slightly modified from diagram of Felix Reidl – circa 2020

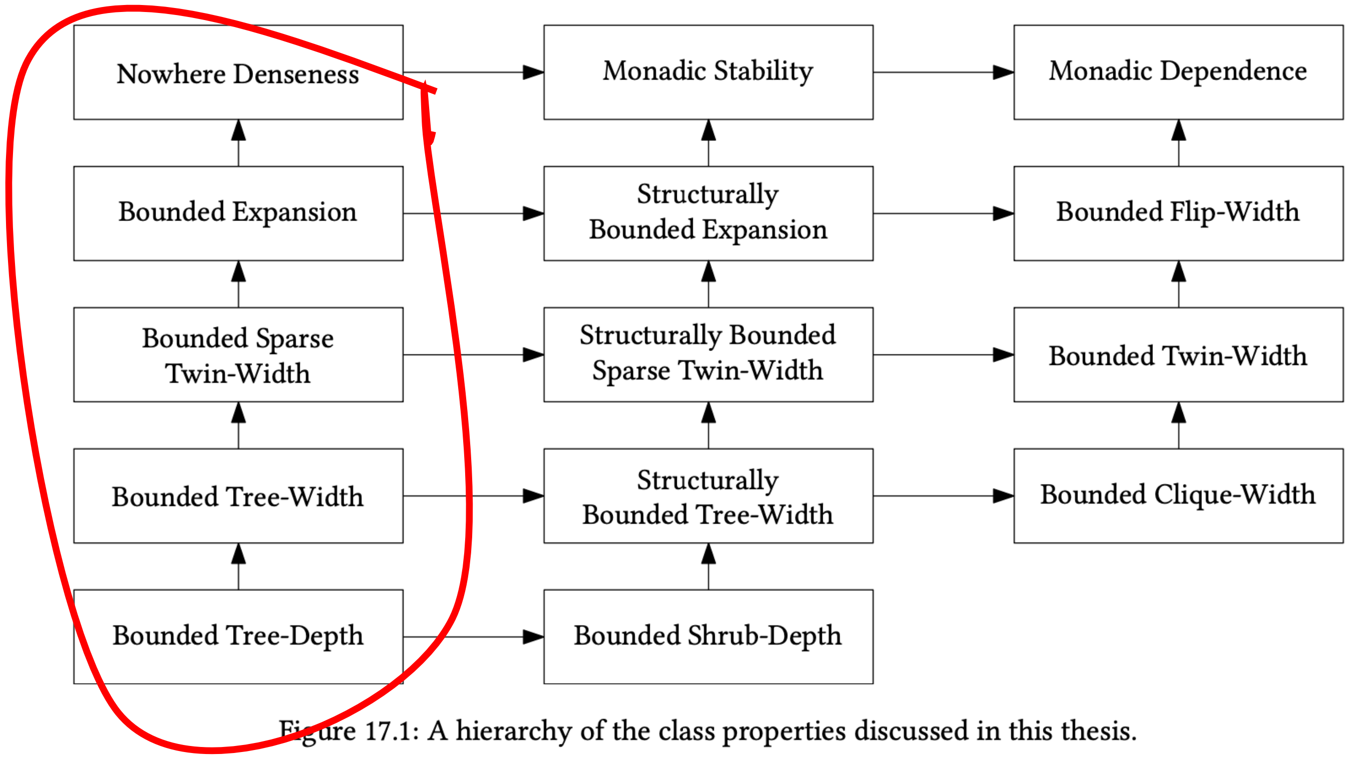
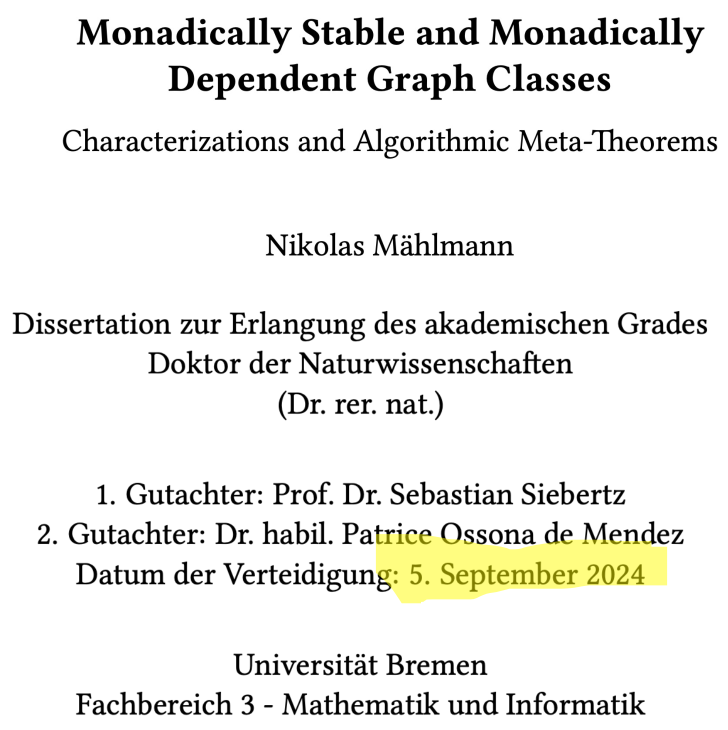


Figure 17.1: A hierarchy of the class properties discussed in this thesis.



Monadically Stable and Monadically Dependent Graph Classes

Characterizations and Algorithmic Meta-Theorems

Nikolas Mählmann

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)

1. Gutachter: Prof. Dr. Sebastian Siebertz
2. Gutachter: Dr. habil. Patrice Ossona de Mendez
Datum der Verteidigung: 5. September 2024

Universität Bremen
Fachbereich 3 - Mathematik und Informatik

All of these classes are sparse graphs

They have been recently shown to all be essentially FPT – for any problem definable in a certain class of ways

Fixed parameter tractable.
Classical Complexity Theory – growth of "cost" with size of graph (N – but sometimes other parameters from your problem) is either bounded by a polynomial (good) or not (bad).
FPT – even if not bounded by a polynomial maybe there is a parameter (p) that if fixed, the growth (typically with N) IS polynomially bounded FOR THAT FIXED p

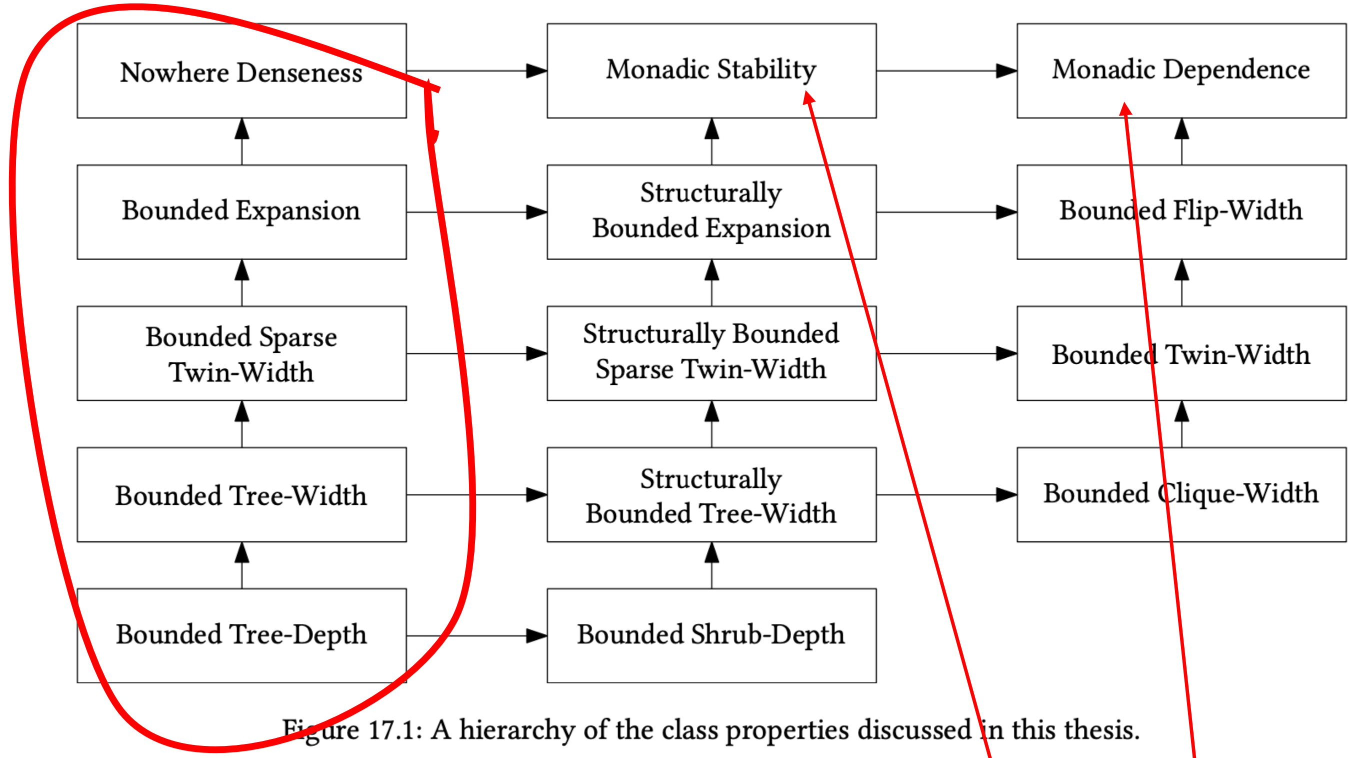
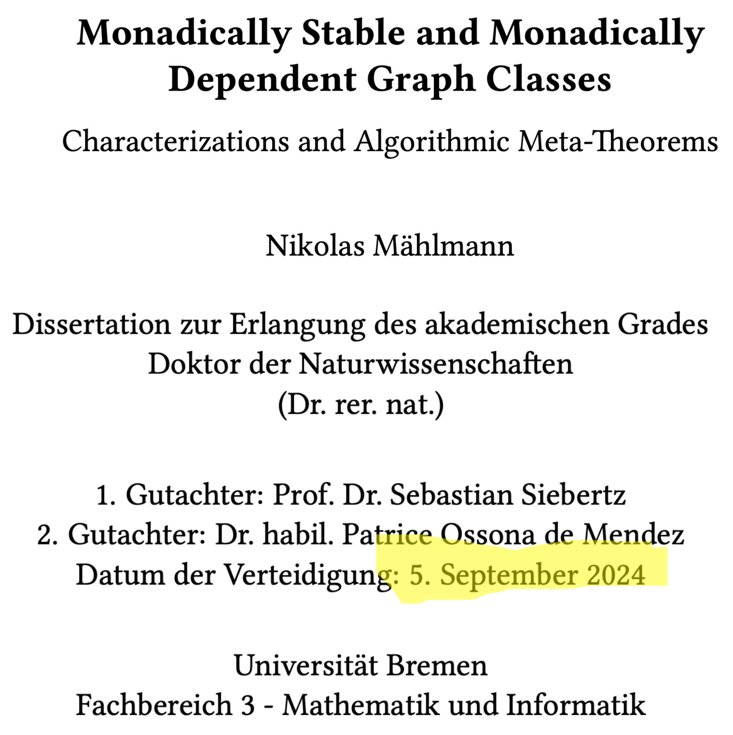


Figure 17.1: A hierarchy of the class properties discussed in this thesis.



Monadically Stable and Monadically Dependent Graph Classes

Characterizations and Algorithmic Meta-Theorems

Nikolas Mählmann

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)

1. Gutachter: Prof. Dr. Sebastian Siebertz
2. Gutachter: Dr. habil. Patrice Ossona de Mendez
Datum der Verteidigung: 5. September 2024

Universität Bremen
Fachbereich 3 - Mathematik und Informatik

All of these classes are sparse graphs

They have been recently shown to all be essentially FPT – for any problem definable in a certain class of ways

The "definable in a certain class of ways" means definable in terms of First Order Logic (FO)
If your graph classes are either the monadically stable property or have Monadic Dependence (latter implies the former) then problems definable in FO are FPT on that class
ALL the graph classes here have Monadic dependence – problems definable by FO are FPT on these graph classes.

Extending knowledge of FPT from sparse graph classes to non-sparse classes

- The previous picture –extended the knowledge of which graph classes are “nice” from an FPT/FO point of view from graph classes that are sparse – to other graph classes that also afford FPT/FO
- But this is all “high level” – the race is on to tie this to detailed picture and to work out algorithmic consequences
- My current research is in part to do just this for graph classes “natural” under a clustering/robust fitting end objective.