Structure based SAT dataset for analysing GNN generalisation

Yi Fu **Anthony Tompkins** **Yang Song** **Maurice Pagnucco**
School of Computer Science and Engineering, University of New South Wales, Australia

Abstract

Satisfiability (SAT) solvers based on techniques such as conflict driven clause learning (CDCL) have produced excellent performance on both synthetic and real world industrial problems. While these CDCL solvers only operate on a per-problem basis, graph neural network (GNN) based solvers bring new benefits to the field by allowing practitioners to exploit knowledge gained from solved problems to expedite solving of new SAT problems. However, one specific area that is often studied in the context of CDCL solvers, but largely overlooked in GNN solvers, is the relationship between graph theoretic measure of *structure* in SAT problems and the *generalisation* ability of GNN solvers. To bridge the gap between structural graph properties (e.g., modularity, self-similarity) and the generalisability (or lack thereof) of GNN based SAT solvers, we present **StructureSAT**: a curated dataset, along with code to further generate novel examples, containing a diverse set of SAT problems from well known problem domains. Furthermore, we utilise a novel splitting method that focuses on deconstructing the families into more detailed hierarchies based on their structural properties. With the new dataset, we aim to help explain problematic generalisation in existing GNN SAT solvers by exploiting knowledge of structural graph properties. We conclude with multiple future directions that can help researchers in GNN based SAT solving develop more effective and generalisable SAT solvers.

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

1 INTRODUCTION

The satisfiability (SAT) [Biere et al., 2009b] problem is a hallmark of computer science research with remarkable real-world utility, especially in solving combinatorial optimisation problems. SAT forms the basis of the study of computational complexity especially around the complexity class of NP problems. Moreover, as a theoretical tool, it facilitate research into the nature of computation and solving difficult computational problems. On the practical side, SAT has been applied to many interesting real world problems such as logistics planning [Kautz and Selman, 1999], product configuration [Sinz et al., 2003], and software verification [Ivančić et al., 2008], retaining its high relevance today.

Advances over the last decades in SAT solving have appeared to converge on conflict driven clause learning (CDCL) methods [Marques-Silva et al., 2021] for the best general problem solving performance. While it is widely believed that CDCL solvers are performant towards various aspects of problems [Alyahya et al., 2023], such solvers almost exclusively operate on a per-problem basis. That is to say they do not explicitly reuse knowledge from different problems. Alternatively, graph neural networks (GNNs) have emerged as a complementary approach to representing and solving SAT problems by incorporating the benefits of deep learning [Guo et al., 2023]. Using an optimisation based approach, as opposed to a pure search like algorithm in CDCL, GNN-based solvers have the potential to adapt useful information from training problems to accelerate solving unseen novel problems.

However, SAT problems largely reside in NP-hard problems. With machine learning methods such as GNNs, prior works have demonstrated provably negative results on challenges that are NP-hard [Yehuda et al., 2020]. Although deep neural networks have the ability to ingest large datasets [Krizhevsky et al., 2017, Simonyan and Zisserman, 2015], there lacks a dataset that facilitates sufficiently large and unbiased training

for SAT problems, and existing SAT datasets with higher difficulties generally have a limited number of problems. Indeed, currently the largest benchmarks SATLIB [Hoos and Stützle, 2023] and SATCOMP [International SAT Competition,] contain less than 10k industrial problems. While we can generate synthetic datasets to train GNNs [Selsam et al., 2019], the resulting models would struggle to generalise to more diverse and challenging, or real-world problems [Li et al., 2024b], limiting the application of GNN-based solvers.

Moreover, currently, the generalisability of GNN solvers has been significantly overlooked, especially in terms of the relationship between generalisation and graph structures of SAT problems. More specifically, SAT problems in prior work are typically generated in a random manner without delving into what makes instances difficult or useful for training and testing, potentially limiting the models from generalising to more challenging datasets, especially to industrial instances. For example, the largest SAT dataset on GNN – G4SATBench [Li et al., 2024b], which is constructed with 7 generators, only considers generalisation regarding the numbers of variables as a measure of performance vs. complexity. The impact of the *structural properties* of a dataset is however ignored and remains a less explored area. In particular, problem difficulty can be strongly influenced by the intrinsic *graph structure* of each problems, as shown experimentally using traditional solver [Alyahya et al., 2023]. It is thus intuitive to hypothesise that such structural properties studied in SAT would influence GNN's performance, especially their ability to generalise. However, for most GNN solvers, only the ability to generalise to larger, in-distribution problems from the same domain is discussed, whereas the structural properties across domains are not investigated.

To bridge the gap between structural measures and generalisability of GNNs on SAT, we propose StructureSAT, a large-scale dataset containing diverse problem domains and structural measures. In this work, we are not interested in the set of all possible SAT problems, but rather existing, well studied, SAT problem domains which we aim to analyse through the lens of graph theoretic structure. Thus, we focus on easy-to-generate synthetic training datasets for investigating GNN's generalisability. StructureSAT contains 11 SAT domains from 4 high level categories: random, crafted, pseudo-industrial and industrial, within which we study 9 structural properties that have proven to be influential to traditional SAT solvers, including both conjunctive normal form (CNF) based and graph based properties [Alyahya et al., 2023]. With traditional SAT solvers, typically certain structure values are controlled for each domain, and CDCL solving time are recorded as a metric of solver performance [Giráldez-Cru and Levy, 2016], so that the effects of different structural properties can be analysed using different domains. However, it is difficult and expensive to follow this approach in GNN-SAT solving given the vast amount of combination of choices in training and testing domains, structure values, models used and evaluation metrics. To make this task manageable, in this work, we generate different groups of training and testing sets based on the structure values, and evaluate the GNN generalisability for both in-domain and out-domain distributions. Specifically, we carefully deconstruct each problem domains based on their structural properties, and split each domain into multiple subsets based on the range of values for each of the properties. Based on these subsets, we analyse the relationship between the SAT problem structures and generalisation ability of GNN solvers with three GNN models. This analysis is conducted on a diverse set of in-domain (training/validation and testing sets from the same domain but with different structural ranges) and out-domain problems (training and testing sets from different domains) to comprehensively evaluate different scenarios for generalisation. Our results demonstrate that GNN solvers provide higher generalisability for certain structural properties but significantly poor performance for others, thus highlighting potential future directions on improving the generalisation performance of current GNN-based solvers. The main contributions of this paper are as follows:

- We present StructureSAT, a large-scale dataset for investigating generalisation capability of GNN-based SAT solvers.
- We provide a comprehensive analysis on the properties of StructureSAT and problem difficulty, demonstrating that different graph structures have different effects on the generalisability of GNN solvers.
- We conduct extensive experiments with GNNs, focusing on how structural properties of each problem domain affect different generalisation tasks with both in-domain and out-of-domain SAT problems. Our results indicate that significant future work is required to enhance the generalisation performance of GNN-based SAT solvers.

2 RELATED WORK

SAT Dataset. SAT problems can be classified into different categories, mainly as random, crafted, and industrial [Alyahya et al., 2022], while most come from either synthetic generators or existing datasets. Most generators such as CNFgen [Lauria et al., 2017], focus on random problems like random k-SAT or combinatorial problems.

The largest established datasets are SATLIB [Hoos and Stützle, 2023] and SAT Competitions (SATCOMP) [International SAT Competition,], while only SATCOMP contains industrial problems and is limited in size. To overcome this constraint, new generators have been proposed to generate pseudo-industrial instances, including hand-crafted generators Community Attachment [Giráldez-Crú and Levy, 2015] and Popularity-Similarity [Giráldez-Cru and Levy, 2017], or graph-generative models generators [You et al., 2019, Li et al., 2024a, Li et al., 2023, Chen et al., 2023, Garzón et al., 2022]. For GNN based SAT solving, [Selsam et al., 2019] proposed random generator SR(n) and [Cameron et al., 2020] generated uniform-random 3-SAT instances as datasets. G4SATBench [Li et al., 2024b] produced a dataset including 7 types of synthetic generators with varying variable sizes.

SAT Structural Properties. As traditional SAT solvers perform differently over random, crafted, and industrial instances, it is believed that different SAT domains have distinct underlying properties [Alyahya et al., 2023], including problem-based properties and solver-based properties. Results in this filed have been wildy used in improving traditional solvers [Audemard and Simon, 2009] and portfolio based solvers [Sonobe, 2016]. Problem-based properties are further divided to CNF-based, including phase-transition [Cheeseman et al., 1991], backdoor [Kilby et al., 2005] and backbones [Kilby et al., 2005], and graph-based, including scale-free [Ansótegui et al., 2009], self-similar [Ansótegui et al., 2014], centrality [Katsirelos and Simon, 2012], small-worlds [Walsh et al., 1999] and community structure [Ansótegui et al., 2019]. Solver-based properties are related to SAT solvers such as: mergeability and resolvability [Zulkoski et al., 2018]. These measures are either proved mathematically, or analysed through solver-related parameters such as solving time. However, all the works are experimented with traditional SAT solvers, and most work only focus on single property measure [Li et al., 2021, Zulkoski et al., 2018]. In this work, we select several related properties, and calculate their values on each of our selected domains. We also split our dataset to different values based on each structure, and analyse the important or hard features for GNN to capture.

GNN for SAT Solving. GNN has been applied to SAT solving mainly as problem solvers, including standalone solvers, which are networks trained to classify problem satisfiability themselves and mainly encode the formula as LCG graphs [Selsam et al., 2019, Zhang et al., 2022, Chang et al., 2022, Shi et al., 2023, Cameron et al., 2020, Hartford et al., 2018, Duan et al., 2022, Ozolins et al., 2022], and hybrid solvers, which treat GNN as guidance to traditional solvers by replacing their heuristics with network predictions. These solvers focus on specific tasks and corresponding problems such as UNSAT core [Selsam and Bjørner, 2019], glue clauses [Han, 2020] or backbone variables [Wang et al., 2023] for CDCL solvers and initial assignment [Zhang et al., 2020, Li and Si, 2022] for SLS solvers. Although hybrid solvers generally achieve better results than standalone solvers, they act as modifications of traditional solvers instead of discussing solvability of GNNs directly, which are out of scope of our work.

3 DATASET DESCRIPTION

StructureSAT consists of multiple problem domains, for each of which we control the corresponding attribute values. In this section, we start by introducing the problem definition of satisfiability (SAT). Then, we present the graph structure properties measured in the dataset. Next, we introduce the types of data and generators used for raw data generation. Last but not least, we detail the construction process, structure-based splitting methods, and generalisation tasks.

3.1 SAT Preliminaries

In propositional logic, the SAT problem is the problem of finding an assignment of truth values (true or false) to propositional variables that make a Boolean formula satisfiable (i.e., true). Boolean formulae are typically expressed in conjunctive normal form (CNF). We denote the set of propositional variables by V. A literal l is either a variable $v \in V$ or its negation $\neg v$ (or $\overline{v}$). A clause c is a disjunction of literals $(l_1 \vee l_2 \vee ..l_n)$, where $\vee$ denotes propositional logic connective "or". A formula f is a conjunction of clauses $(c_1 \wedge c_2 \wedge ...c_n)$, where $\wedge$ denotes propositional logic connective "and". Generally speaking classical SAT solvers can be divided into *complete solvers* and *incomplete solvers*. A complete solver is able to prove unsatisfiability or find a satisfying assignment if they exist for a problem. Most complete solvers are based on the Davis–Putnam–Logemann–Loveland (DPLL) algorithm [Davis and Putnam, 1960, Davis et al., 1962], a backtracking search algorithm. Two popular types of solvers built from the DPLL solver are the Conflict-Driven Clause Learning (CDCL) solvers [Marques-Silva et al., 2021], which learn and add new conflict clauses to the original formula, and look-ahead

solvers [Heule et al., 2012]. In contrast, an incomplete solver cannot prove unsatisfiability such as the Moser-Tardos (MT) solver [Catarata et al., 2017] and WalkSAT solver [Selman et al., 1994] that are derived from the stochastic local search (SLS) algorithm.

SAT formulas have been expressed using various graphs [Alyahya et al., 2023]. Traditional SAT community usually encodes SAT formulas as undirected weighted graph including *variable incidence graph* (VIG) and *clause-variable incidence graph* (CVIG) [Ansótegui et al., 2012]. VIG is a graph with variables as nodes and two literals are connected with edges if they occur in the same clauses. CVIG is a bipartite graph having variables and clauses as vertices. They are connected if a variable occurs inside a clause. Another commonly used set of graphs are unweighted graphs including VIG, *variable clause graphs* (VCG), *literal incidence graphs* (LIG), and *literal clause graphs* (LCG). LIG extends VIG through extra edges between literals and their negations. VCG and LCG are bipartite graphs similar to CVIG, but without weights. LCG also has an extra edge between literals and their complements. Among all the graphs, LCG has been used most in GNN based SAT solving as others either lose important information of clause (LIG and VIG), or information of polarity (VCG). As prior research [Li et al., 2024b] has shown that the particular bipartite graph construction (LCG or VCG) does not have much different effect on model accuracy while non-bipartite graphs (LIG or VIG) could not represent SAT well for GNN training, in this work we will be focusing on LCG only.

3.2 SAT Structure Properties

In this work, we choose several structural properties from prior studies based on two criteria: public availability of their code and algorithms, and their proven impact on traditional CDCL solvers. We then apply these properties as a splitting method for training and testing in the dataset. StructureSAT classifies structural properties of SAT problems into two classes based on the embedding methods: CNF based properties and graph based properties [Alyahya et al., 2022].

CNF based Properties. Given a problem P with set of variables V, the CNF based properties are related to the CNF encoding of a problem.

- **Phase transition** [Cheeseman et al., 1991] is a phenomenon measured by clause to variable ratio (n_c/n_v), where n_c and n_v represents number of clauses and variables respectively. It is evident that an easy-hard-easy pattern occurs for random k-SAT, where transition for random 3-SAT is at $c = 4.258{*}n{+}58.26{*}n^{-1/2}$ [Crawford and Auton, 1996]. Although the phenomenon has been observed in both random k-SAT and peudo-industrial instances, for simplicity we will only be focusing it on random 3-SAT in this dataset.
- **Backbone** [Kilby et al., 2005] refers to the set of literals of a problem which are true in all satisfying assignments. The value of each literals in the set is fixed in all assignments of the problem. StructureSAT calculates the size of the backbones(B_b) of satisfiable instances only.

Graph based Properties are structural properties from embeddings of a LCG graph G. Given a problem with set of variables V and set of clauses C, the LCG graph G with vertex X and weight w is defined as $G = (X, x|x \in C \wedge V, w)$ and the weight is either $1/|X|$ if variable is in clause, $1/|V|$ for weights between variables and their negations, or 0 if nodes are not connected. Each node x in graph has degree deg_x.

- **Self-similary** [Ansótegui et al., 2014] is measured by fractal dimension(D_f). G is self-similar if the minimum number of boxes of size s required to cover G decreases polynomially for some D_f.
- **Scale-free** [Ansótegui et al., 2009] can be measured by frequency of variables (α_v) or clauses size (α_c). A graph is scale-free if the arity of nodes is characterised by a random variable N that follows a power-law distribution.
- **Treewidth** (T_w) [Mateescu, 2011] measures the tree-likeness of graphs. It is the minimum width over all possible tree decomposition of G. In this work we calculate treewidth using $treewidth - min - degree$ function from NetworkX [Hagberg and Conway, 2020], which calculates using the Minimum Degree heuristic.
- **Centrality** specifies how important a node is within a graph. Following [Freeman, 1977], we calculate the betweeness centrality BE using NetworkX [Hagberg and Conway, 2020].
- **Community structure** [Ansótegui et al., 2019] is measured by modularity(Q). The modularity Q of G is computed with respect to a given partition C of the same graph and is a measure of the fraction of within-community edges in relation to *another* random graph that has an equal number of vertices and degree. A graph's modularity score corresponds to the maximal modularity of any possible partition in C: $Q(G) = \max\{Q(G, C)|C\}$. While computing the exact value of Q is NP-hard, we would approximate lower-bounds to Q.

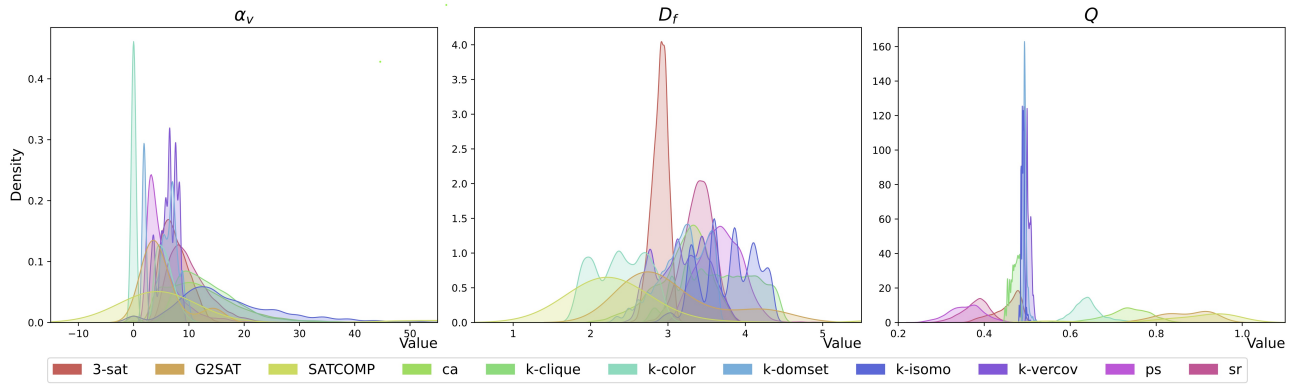

Figure 1: Distributions of various structural graph properties from different problem domains. Left: scale-free measure by variable α_v, Middle: fractal dimension D_f, Right: modularity Q.

- **Small-world** [Walsh et al., 1999] measures the extent of graph topology. It is approximated using Proximity Ratio (P_r). G is considered small-world if $P_r \geq 1$. In this work we use the sigma function from NetworkX [Hagberg and Conway, 2020] for calculation, which is only an approximation of the P_r value due to the need for random graph generation.
- **Entropy** [Zhang et al., 2021] measures the uncertainty of random systems. Given volume of the graph vol_G, we measure the One-dimensional Entropy (H) following [Zhang et al., 2021].

3.3 StructureSAT Composition

Dataset Generation We use various existing SAT generators and datasets to produce original SAT formulas in 11 domains. To show the variety of structural differences, we show the distributions of various structural properties in Fig. 1 from calculating the statistical values on raw data pairs with a 60 seconds timeout. The distributions are plotted on 10k problem pairs for each selected domains, with each pair containing 50% satisfiable and 50% unsatisfiable problems. For industrial SAT problems and graph generative model generators, less than 100 pairs are selected within the time limit. Next,we detail the provided domains in StructureSAT, including important parameters used for generation. Note that we create 5 domains from combinational problems, and other approaches produce one domain each.

Random-3-SAT: Uniform random 3-SAT (r-3SAT) is a special case for Random k-SAT where each clause contains exactly 3 literals. The problems are generated using CNFgen [Lauria et al., 2017] with $10 - 40$ number of variables at the phase transition point.

SR(n): SR(n) [Selsam et al., 2019] is a special expression of random k-SAT that consists of a balanced dataset with pairs, with k as the maximum number of variables in the problem. Each pair contains one satisfiable problem and one unsatisfiable problem, differing by 1 single literal in one clause. In this work we use problem with $10 - 40$ number of variables as main training data, represented as SR(10-40).

Combinatorial problems (5 domains): For most combinatorial problems, the goal is to find some combination of elements in a solution space while respecting defined constraints. A valid solution would correspond to a satisfying assignment in the SAT encoding of the problem [Biere et al., 2009a]. In StructureSAT, we generate 5 combinatorial problems using CNFgen [Lauria et al., 2017]. Each problem is related to solving constraints over a graph. Thus, we generate random graphs using the Erdős–Rényi model [Newman, 2018] with edge probability $\binom{v}{k}^{-1/\binom{v}{2}}$, where v representing number of vertex. The selected parameters and problems include k-coloring, $3 \leq k \leq 4$, k-dominating-set, $2 \leq k \leq 3$, k-clique-detection, $3 \leq k \leq 4$, k-vertex-cover, $3 \leq k \leq 5$, and automorphism in graph G.

CA: Community Attachment (CA) is a seminal work generated from [Giráldez-Crú and Levy, 2015] to mimic the community structure of industrial problems, i.e., value of *modularity* Q on a SAT instance's VIG graph. In StuctureSAT, we select 0.7-0.9 as the value of Q. The generator uniformly and randomly selects 4 to 5 literals inside the same community with probability $P = Q + 1/co$, where co is the number of communities, and 4 to 5 literals in distinct community with probability $1 - P$. These process are done iteratively to form a formula.

PS: Popularity-Similarity (PS) [Giráldez-Cru and Levy, 2017] is a generation algorithm developed on the idea of *locality*, which is a

measure of both community structure and scale-free structure on a SAT instance's VCG graph. PS randomly samples variable v in clause c with the probability $P = 1/(1 + n_v^\beta * n_c^{\beta_1} * \theta_{n_v * n_c}/R)^T$, where β is power-law distribution, θ is random angle assigned to v and c, T is temperature between 0.75 and 1.5, and R is an approximate normalisation constant.

SATCOMP: StructureSAT uses industrial instances from SAT competition (SATCOMP) 2007.

G2SAT: To overcome the limiting number of instances in industrial problem, graph generative models have been used as problem generator. In this work we mainly use **G2SAT** [You et al., 2019] as generator. Following previous work, we select problems from the industrial dataset, standadise with SatElite preprocessor[Eén and Biere, 2005], and generate similar instances using GraphSAGE [Hamilton et al., 2017] and a two-phase generation process. Both industrial and G2SAT datasets are used mainly as testing set in the dataset.

3.4 Augmented Problems

This section describes augmented dataset setup that might affect the generalisation ability of GNN. CDCL-based SAT solvers produce conflict clauses during searching and add learned clauses, which are reverse of conflict clauses, to the original problems. Prior work [Li et al., 2024b] has shown that training on augmented SAT problems with the learned clauses added could lead to better accuracy on augmented SAT domains than training on raw problems. Since this might be caused by the destroying of structure during the addition of learned clauses [Ansótegui et al., 2019], we are interested in an empirical analysis on the structural difference and their effect on out-of-distribution generalisation. Thus, we gather learned clauses from DRAT-trim proofs [Wetzler et al., 2014] after running problems with CadiCal solver [Fleury and Heisinger, 2020], then augment our dataset with the collected learned clauses.

3.5 Structure-aware Splitting and Generalisation Selection

We propose a novel splitting method for StructureSAT and focus on 3 types of generalisation tasks. To do this, after generating the raw datasets from each domain, instead of randomly splitting them to train/valid/test sets, we divide each raw dataset based on specific structural values. Specifically, for each domain, we select the average value Z from every calculated graph-based properties, and split each domain into a low-value subset and a high-value subset, producing a total of 16 subsets (8 structures $*$ 2 ranges) per domain. Each subsets contains 80k training data and 10k validation data.[1]

The three types of generalisation tasks are represented using 3 aspects of test problems: in-domain larger problems, in-domain problems with different properties, and out-of-distribution problems. Specifically, for in-domain larger problems, we follow the generation process described in Section 3.3 and randomly generate larger problem sets D_{t1} using all the synthetic generators. For example, for training with SR problems, the training and validation sets contain problems with 10-40 number of variables, while the testing sets contain problems with 40-100 and 100-200 number of variables, each with 10k pairs.

To test generalisation on in-domain problems with different structures, we are interested in CNF-based properties. Thus we focus on random 3-SAT problems with different c/v ratios and different backbone values on all domains. For analysing the c/v ratio on random 3-SAT, the training and validation sets are problems with 10-40 number of variables, and a c/v ratio of roughly 4.7. Our testing set contain random 3-SAT problems with same number of variables, but different number of clauses. Specifically the test set D_{t2} has a c/v ratio of 3.5, 4, 5.5 and 6, with each ratio having 10k pairs of problems. For backbone on all domains, the test sets comprise solely satisfiable problems, categorised according to the average backbone size following the implementation of [Biere et al., 2023]. Additionally, we also include existing dataset - random 3-SAT problems with controlled number of variables (100) and clauses (403) but different backbone sizes sourced from SATLIB [Hoos and Stützle, 2023].

For out-of-distribution problems, we randomly generate 10k problem testing data from each synthetic generators without property split and form D_{t3}, thus holding the structural range as in Fig. 1. We combine problems from every domain, excluding the training domain, to form the out-of-domain testing sets D_{t4}. Small-size industrial and application problems from SATCOMP and generated G2SAT problems are also selected as larger industrial testing set in D_{t3} .

With the novel splitting and generation methods introduced in StructureSAT, we produce different training, validation and testing sets. These sets comprise: (1) problems with certain attribute fixed at a value, (2) problems with certain attribute fixed within a range, and (3) randomly generated problems without attribute-based splitting. These sets are then used to analyse different generalisation tasks.

[1] Our dataset and codebase are available here.

Table 1: NeuroSAT testing results. For each metric, we train NeuroSAT on the small and big split of the corresponding SR(10-40) training sets. We then test the trained model on the testing data of 10-40, 40-100, and 100-200 variables for each metric. Better training split performance in bold.

Train Split	Test Set	Metric					
		D_f	α_v	α_c	T_w	Q	H
small	10-40	94.5	**94.4**	**95.4**	95.0	93.8	94.0
big		**95.4**	93.1	95.3	**95.3**	**94.8**	**95.8**
small	40-100	68.5	**69.7**	70.8	65.5	70.8	64.8
big		**72.4**	67.1	**73.2**	**72.1**	**73.3**	**73.8**
small	100-200	**57.0**	**56.8**	57.7	54.7	**57.9**	53.8
big		57.0	54.5	**59.0**	**56.5**	57.8	**58.0**

4 EXPERIMENT

In this section, we experimentally evaluate the generalisation ability of GNN-based SAT solvers with StructureSAT. We are interested in how is the generalisability of GNN affected by the structures of both training and testing instances. Specifically, we investigate the following questions: Q1: How does the structure influence generalisation to in-domain larger size problems; Q2: Could GNNs generalise to the same domain but for problems with different structure; Q3: How much does structure influence out of distribution generalisation; Q4: Does training on augmented problems influence out of distribution generalisation?

4.1 Evaluation Setup

With the newly developed dataset StructureSAT, we conduct several experiments on the task of predicting satisfiablity of SAT problems, which is considered a supervised binary graph-classification task. We consider different baseline GNN models NeuroSAT [Selsam et al., 2019], and GCN [Kipf and Welling, 2016] and GIN [Xu et al., 2018] in our experiments and follow the implementation of G4SATBench [Li et al., 2024b] for LCG message passing. We refer the reader to Appendix C for implementation details. As prior work has shown different GNN models perform similarly in general [Li et al., 2024b], we want to emphasise that the main focus of this paper is not to find the best performing baseline model. Instead, we aim to find the general trend on generalisation of GNN-based SAT solvers with regard to various structural measures.

We experiment on each of the generalisation tasks mentioned in Section 3.5. Due to limited space, for Q1 and Q3 we only present part of results using SR(10-40) in this section, which is the dataset with the best cross-domain generalisation ability [Li et al., 2024b]. All experimental results on other domains and models can be found in Appendix D. In this section, for simplicity, we make SR stand for SR(10-40), R3 stand for Random-3-SAT, KCL stand for k-clique, KD stand for k-dominating-set, KV stand for k-vertex-cover, KCO stand for k-coloring, AM stand for automorphism, G2 stand for G2SAT, and IN stand for industrial problems. Note that for simple computation, we select 24 small problems from G2SAT. 12 industrial problems used to train G2SAT are used as IN domain.

4.2 Generalisation to In-domain Problems

Large Size Generalisation. To answer Q1, we train a NeuroSAT model on a small SAT problem set and test its performance on varying number of variables from testing set D_{t1}. Table 1 shows the testing result of NeuroSAT trained on SR(10-40), which is SR(n) problems with 10-40 number of variables. In general, larger structural valued problems results in better results on large problems. NeuroSAT are influenced more on certain properties when testing on larger test sets. Specifically, when testing on SR(40-100), H and T_w gets a 9.0% and a 6.7% increase when trained on larger splits. Additionally, when comparing between number of variables of test sets, medium sized test set, SR(40-100), has the most difference between large and small split. We hypothesise this may due to the increase in relevant metric values in larger problems, that makes GNN find more similarity in medium sized properties. Thus, generating problems with property values in the upper range, especially larger H, T_w and α_c values as seen in Table 1 has the potential to enable GNNs solve larger in-domain problems.

Table 2: R3 results with different ratios.

Metric	Accuracy
3.5	97.6
4	95.9
4.7	95.5
5.5	98.7
6	99.4

Different Ratio Generalisation. Table 2 shows the result of NeuroSAT trained on random 3-SAT instances and tested on problems with varying c/v ratios from D_{t2}, through which we try to answer Q2. Surprisingly, although models are trained with c/v ratio around 4.7, which is the "hardest" point for random 3-SAT as discussed in Section 3.2, testing at this point has the lowest accuracy. Furthermore, the further away c/v is from the phase transition point, the better their testing accuracy. This signals that GNNs are not always better at solving in-domain problems with similar properties to the training data. Specifically, for random 3-SAT problems, GNN would capture the structure of "simpler-ratioed" R3 problems more easily than the "harder" ones.

Different Backbone Generalisation. To explore how backbone size affect the generalizability of GNNs, Table 3 displays the performance of NeuroSAT and GCN models trained on R3 without splits and tested on satisfiable problems with varying backbone sizes. These problems, sourced from SATLIB [Hoos and Stützle, 2023], maintain a controlled number of variables (100) and clauses (403). The results indicate that, although the problem sizes are consistent, variations in backbone sizes significantly affect the accuracy of both models. As the backbone sizes increase, the accuracy of both models decreases. Additionally, the tested dataset contains 100 variables, which is larger than the training set variables (10-40 variables). However, the actual accuracies exhibit considerable variation, ranging from a high of 97.1% with 10 backbones to a low of 35.7% with 90 backbones. This significant variance highlights how the presence of larger backbones within a problem can increase complexity and consequently the difficulty for GNNs to solve it. Conversely, problems with fewer backbones have fewer constraints and, therefore, are much easier for GNNs to resolve.

Table 3: Accuracy tested on R3 with different backbone sizes on 2 models.

Backbone size	10	30	50	70	90
NeuroSAT	**97.1**	84.4	71.8	67.9	42.6
GCN	**91.0**	77.6	65.8	54.8	35.7

Although GNNs are primarily given graph-structured data as input, their performance is also significantly influenced by the CNF encoding of the domains. Consequently, relying solely on domain size as a metric for testing generalisability and assessing problem difficulty proves to be insufficient, as there exist small-sized problems that pose significant challenges for GNNs to solve. These findings further suggest that alterations in graph structure could stem from changes in CNF-based properties, indicating needs for further testing to explore this relationship more comprehensively.

4.3 Generalisation to Other Domains

Out-of-distribution Generalisation without Augmentation To answer Q3, we train different models on the small and big splits of different domains, and test on D_{t4}, which contains 8 domains outside of the training domains. Since D_{t4} contains multiple domains, the set contains a wide range of values for each structure. Table 4 presents the mean and standard deviation of the results across 3 runs trained using SR(10-40) on 3 models. We refer the reader to Appendix D for additional experimental results. For some results in Table 4, the accuracy differences between the small and large metric splits are not significant across all model types, indicating either all models fail to capture the impact of the metrics, or the metrics in the training set have minimal influence on the model. In this work, we adopt the second assumption since we use three model types and their performance aligns with this interpretation. However, we acknowledge that the first assumption could also hold, which should be analysed in future work. Furthermore, if one model shows significant differences between the splits while another does not, we infer that the less significant model fails to effectively capture the impact of the property.

For different models, changing certain structural values in the training set would influence the testing accuracy to different extent. For instance, there is a higher difference between D_f splits on GIN(3.71%) than on GCN(1.6%), meaning that D_f is more important in the GIN learning process than GCN. This suggests different models would focus more on capturing certain structures of the dataset. Within the 3 models, Q seems to have limited influence on the testing accuracy, while H influences all models's performance to a greater extent. Moreover, for each model, the structural value influences are different. For GIN, bigger α_v groups achieve better results while smaller α_c groups seems to be better generalised.

Furthermore, we investigate how similarities in specific structural values between the training and testing datasets affect testing accuracy. We train a GIN model on different groups of SR(10-40) and evaluate its performance on D_{t3}. The average results across 3 runs are presented in Table 5. To better understand the structural differences between the training and testing domains, we plotted the average of all structural values and would refer readers to Appendix B for detailed analysis. Overall, the GIN model learns from the input SR structures and generalises more effectively to out-domain problems with similar structural characteristics. For example, the testing set R3 has a smaller average D_f compared to SR, while PS has a smaller Q than SR. Models trained on groups with smaller D_f and Q values demonstrate better generalisation to R3 and PS, respectively. This suggests that training on problems with similar structural values could enhance GNN generalisation to the target domain.

However, this pattern does not hold for certain properties and domains, such as Q in AM, where AM has a larger Q than SR, yet models trained on a smaller subset of SR generalise better to AM. One possible explanation for this discrepancy is that GNN performance is not determined by a single structural property but by a combination of structural features. Another possibil-

ity is that numerical structural values alone may not be sufficient to fully capture the problem structures learned by the GNN. For domains like AM, while altering the α_v value in the training set does affect testing outcomes, even the improved models struggle to achieve strong performance, which needs further exploration to understand this phenomenon.

Table 4: Accuracy of models trained on SR(10-40) with different structural splits.

Metric	Split	Testing Accuracies		
		NeuroSAT	GIN	GCN
D_f	small	66.4 ± 0.7	68.5 ± 1.5	58.6 ± 1.3
	big	68.1 ± 1.2	64.8 ± 2.1	57.0 ± 0.4
α_v	small	67.5 ± 1.3	63.4 ± 1.1	61.5 ± 0.6
	big	67.8 ± 2.1	67.1 ± 2.3	61.5 ± 0.5
α_c	small	68.8 ± 1.2	68.9 ± 1.2	62.1 ± 0.6
	big	68.0 ± 3.2	65.1 ± 1.8	62.0 ± 1.0
T_w	small	66.8 ± 0.9	66.7 ± 3.5	58.2 ± 0.1
	big	64.5 ± 1.4	60.9 ± 4.6	59.1 ± 1.3
Q	small	66.8 ± 2.5	64.5 ± 2.5	60.1 ± 0.1
	big	66.7 ± 1.4	64.9 ± 1.5	59.8 ± 0.5
H	small	68.5 ± 2.0	67.4 ± 0.9	63.6 ± 2.9
	big	66.2 ± 4.4	63.3 ± 3.3	56.9 ± 0.9

Augmented Problem Generalisation. To get insights to Q4, we augment SR(10-40) with learned clauses from Cadical solver. To do this, we append all the learned clauses to the "raw" problems in chronological order in which they were generated by Cadical, and label them as "augmented" set in the experiment. Then, we test the results on different domains as shown in Table 6. Both training and testing data here are not splitted with structural properties. Although models trained on augmented datasets reach high accuracy in other augmented domains, with only a few learned clauses added (on average the ratio between learned clause and original clause is roughly 15.9% in training data), they could not solve any raw data well in any domains. While most structural parameters remain the same before and after augmentation, the relatively large change in D_f could be one reason for this result.

4.4 Analysis

The results in both Section 4 and Appendix D suggest that the generalisability of GNN is influenced by the structure of the data to varying extent. GNNs tend to perform worse on larger problems that contain more variables than those in the training sets. This decline in performance is not solely attributable to scalability issues inherent to GNN models but is also likely due to changes in the structural properties and connectivity patterns of the larger problems compared to those encountered during training. Interestingly, when models are trained on smaller-sized problems that possess larger structural values, they achieve higher testing accuracy on larger-sized problems. This suggests a possible correlation where higher accuracy is obtained when the training and testing structures are similar.

Table 5: Average accuracy of GIN trained on SR(10-40) with different structural splits.

Metric	Split	Testing Domains									
		R3	KCL	KD	KV	KCO	AM	CA	PS	G2	IN
D_f	small	92.0	53.3	64.6	65.0	49.0	50.4	75.7	91.5	47.2	86.1
	big	88.3	57.2	62.5	59.2	52.2	46.8	64.6	88.1	61.1	63.9
α_v	small	93.5	51.6	55.3	55.9	47.6	47.4	65.5	93.8	50.0	80.6
	big	92.3	53.4	59.0	65.5	50.0	52.3	70.5	93.7	50.0	63.9
α_c	small	93.3	54.3	60.1	65.5	59.6	50.3	68.0	95.4	58.3	79.2
	big	93.4	51.7	54.8	59.0	56.7	49.0	65.0	92.1	52.8	57.0
Q	small	90.8	48.8	59.9	60.8	49.4	53.6	63.6	91.6	50.0	86.1
	big	89.9	53.2	63.2	62.5	50.0	52.7	65.6	86.9	38.7	68.1

Table 6: Augmented experiments. Top: NeuroSAT trained on augmented datasets. Testing dataset is split to augmented and raw. Bottom: Average graph based property values before and after augmenting.

Domain	Split	R3	KCL	KV	KD	KCO	CA	PS	G2	IN
SR	augmented	**99.9**	50.0	**58.9**	**64.0**	**94.4**	**70.4**	**99.6**	4.2	**66.7**
	raw	50.0	50.0	56.0	54.0	57.4	50.0	51.1	4.2	41.7

Domain	Split	D_f	α_c	α_v	Q	T_w	B_e	H
SR	augmented	2.5	4.2	9.8	0.4	39.2	0.01	5.0
	raw	3.4	4.3	10.0	0.4	39.1	0.01	5.0

For in-domain problems, GNNs tend to find certain types of problems, such as random instances with a c/v ratio around the phase transition or random problems with higher backbone sizes, more challenging compared to others under the same evaluation metric. For out-of-domain problems, the influence of graph-based properties on model performance appeared less pronounced. Overall, while some structural variations are not effectively captured by GNNs, as evidenced across all three models, there is a general trend of GNNs performing better on problems with similar graph structures. However, this tendency does not hold for certain problem types, highlighting the need for further investigation into combined structural properties.

5 CONCLUSION

This paper presents StructureSAT, a structure-based SAT dataset. We aim to help analyse GNN based generalisation in SAT solving by adapting a variety of structural properties. Through our extensive cross-domain experiments using StructureSAT, we produced various insights in GNN generalisation with empirical analysis. Our experiments suggest certain properties are more influential on generalisability than others. Furthermore, the diverse GNN generalisation abilities on different domains could be the result of a combination of properties. We hope StructureSAT brings interest into the role of structural properties for future research into GNN based SAT solving.

References

[Alyahya et al., 2022] Alyahya, T. N., Menai, M. E. B., and Mathkour, H. (2022). On the structure of the Boolean satisfiability problem: A survey. *ACM Computing Surveys (CSUR)*, 55(3).

[Alyahya et al., 2023] Alyahya, T. N., Menai, M. E. B., and Mathkour, H. (2023). On the structure of the boolean satisfiability problem: A survey. *ACM Computing Surveys*.

[Ansótegui et al., 2014] Ansótegui, C., Bonet, M. L., Giráldez-Cru, J., and Levy, J. (2014). The fractal dimension of SAT formulas. In *7th International Joint Conference on Automated Reasoning (IJCAR-2014)*.

[Ansótegui et al., 2019] Ansótegui, C., Bonet, M. L., Giráldez-Cru, J., Levy, J., and Simon, L. (2019). Community structure in industrial SAT instances. *Journal of Artificial Intelligence Research*.

[Ansótegui et al., 2009] Ansótegui, C., Bonet, M. L., and Levy, J. (2009). On the structure of industrial SAT instances. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*. Springer.

[Ansótegui et al., 2012] Ansótegui, C., Giráldez-Cru, J., and Levy, J. (2012). The community structure of sat formulas. In *International Conference on Theory and Applications of Satisfiability Testing*. Springer.

[Audemard and Simon, 2009] Audemard, G. and Simon, L. (2009). Predicting learnt clauses quality in modern sat solvers. In *Twenty-first international joint conference on artificial intelligence*.

[Biere et al., 2023] Biere, A., Froleyks, N., and Wang, W. (2023). Cadiback: Extracting backbones with cadical. In *26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023)*, volume 271. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

[Biere et al., 2009a] Biere, A., Heule, M., and van Maaren, H. (2009a). *Handbook of satisfiability*, volume 185. IOS press.

[Biere et al., 2009b] Biere, A., Heule, M., van Maaren, H., and Walsh, T. (2009b). *Handbook of Satisfiability*. IOS Press.

[Cameron et al., 2020] Cameron, C., Chen, R., Hartford, J., and Leyton-Brown, K. (2020). Predicting propositional satisfiability via end-to-end learning. *The AAAI Conference on Artificial Intelligence*.

[Catarata et al., 2017] Catarata, J. D., Corbett, S., Stern, H., Szegedy, M., Vyskocil, T., and Zhang, Z. (2017). The Moser-Tardos Resample algorithm: Where is the limit? (an experimental inquiry). In *2017 Proceedings of the Ninteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*. Society for Industrial and Applied Mathematics.

[Chang et al., 2022] Chang, W., Zhang, H., and Luo, J. (2022). Predicting propositional satisfiability based on graph attention networks. *International Journal of Computational Intelligence Systems*.

[Cheeseman et al., 1991] Cheeseman, P., Kanefsky, B., and Taylor, W. M. (1991). Where the really hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, volume 91.

[Chen et al., 2023] Chen, X., Li, Y., Wang, R., and Yan, J. (2023). From matching to mixing: A graph interpolation approach for sat instance generation. In *The Twelfth International Conference on Learning Representations*.

[Crawford and Auton, 1996] Crawford, J. M. and Auton, L. D. (1996). Experimental results on the crossover point in random 3-sat. *Artificial intelligence*, 81(1-2).

[Davis et al., 1962] Davis, M., Logemann, G., and Loveland, D. (1962). A machine program for theorem-proving. *Communications of the ACM*.

[Davis and Putnam, 1960] Davis, M. and Putnam, H. (1960). A computing procedure for quantification theory. *Journal of the ACM*.

[Duan et al., 2022] Duan, H., Vaezipoor, P., Paulus, M. B., Ruan, Y., and Maddison, C. (2022). Augment with care: Contrastive learning for combinatorial problems. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR.

[Eén and Biere, 2005] Eén, N. and Biere, A. (2005). Effective preprocessing in sat through variable and clause elimination. In *International conference on theory and applications of satisfiability testing*. Springer.

[Fleury and Heisinger, 2020] Fleury, A. and Heisinger, M. (2020). Cadical, Kissat, Paracooba, Plingeling and treengeling entering the SAT competition 2020. *SAT Competition*.

[Freeman, 1977] Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*.

[Garzón et al., 2022] Garzón, I., Mesejo, P., and Giráldez-Cru, J. (2022). On the performance of deep generative models of realistic sat instances. In *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

[Giráldez-Crú and Levy, 2015] Giráldez-Crú, J. and Levy, J. (2015). A modularity-based random sat instances generator. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*. AAAI Press.

[Giráldez-Cru and Levy, 2016] Giráldez-Cru, J. and Levy, J. (2016). Generating sat instances with community structure. *Artificial Intelligence*, 238.

[Giráldez-Cru and Levy, 2017] Giráldez-Cru, J. and Levy, J. (2017). Locality in random SAT instances. In *Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*.

[Guo et al., 2023] Guo, W., Zhen, H.-L., Li, X., Luo, W., Yuan, M., Jin, Y., and Yan, J. (2023). Machine learning methods in solving the boolean satisfiability problem. *Machine Intelligence Research*, 20(5).

[Hagberg and Conway, 2020] Hagberg, A. and Conway, D. (2020). Networkx: Network analysis with python. *URL: https://networkx. github. io*.

[Hamilton et al., 2017] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, volume 30.

[Han, 2020] Han, J. M. (2020). Enhancing SAT solvers with glue variable predictions.

[Hartford et al., 2018] Hartford, J., Graham, D., Leyton-Brown, K., and Ravanbakhsh, S. (2018). Deep models of interactions across sets. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR.

[Heule et al., 2012] Heule, M. J. H., Kullmann, O., Wieringa, S., and Biere, A. (2012). Cube and Conquer: Guiding CDCL SAT solvers by lookaheads. In *Hardware and Software: Verification and Testing*. Springer Berlin Heidelberg.

[Hoos and Stützle, 2023] Hoos, H. H. and Stützle, T. (2023). SATLIB: An online resource for research on SAT. In I.P.Gent, Maaren, H., and Walsh, T., editors, *SAT 2000*. Last accessed 1/2/2025.

[International SAT Competition,] International SAT Competition. The international SAT competition web page. `http://www.satcompetition.org`. Last accessed 1/2/2025.

[Ivančić et al., 2008] Ivančić, F., Yang, Z., Ganai, M. K., Gupta, A., and Ashar, P. (2008). Efficient SAT-based bounded model checking for software verification. *Theoretical Computer Science*.

[Katsirelos and Simon, 2012] Katsirelos, G. and Simon, L. (2012). Eigenvector centrality in industrial sat instances. In *International Conference on Principles and Practice of Constraint Programming*. Springer.

[Kautz and Selman, 1999] Kautz, H. and Selman, B. (1999). Unifying SAT-based and Graph-based Planning. *International Joint Conference on Artificial Intelligence*.

[Kilby et al., 2005] Kilby, P., Slaney, J., Thiébaux, S., Walsh, T., et al. (2005). Backbones and backdoors in satisfiability. In *Proceedings of AAAI*, volume 5.

[Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[Krizhevsky et al., 2017] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6).

[Lauria et al., 2017] Lauria, M., Elffers, J., Nordström, J., and Vinyals, M. (2017). CNFgen: A generator of crafted benchmarks. In *Theory and Applications of Satisfiability Testing – SAT 2017*. Springer International Publishing.

[Li et al., 2021] Li, C., Chung, J., Mukherjee, S., Vinyals, M., Fleming, N., Kolokolova, A., Mu, A., and Ganesh, V. (2021). On the hierarchical community structure of practical boolean formulas. In *Theory and Applications of Satisfiability Testing–SAT 2021: 24th International Conference*. Springer.

[Li et al., 2023] Li, Y., Chen, X., Guo, W., Li, X., Luo, W., Huang, J., Zhen, H.-L., Yuan, M., and Yan, J. (2023). Hardsatgen: Understanding the difficulty of hard sat formula generation and a strong structure-hardness-aware baseline. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[Li et al., 2024a] Li, Y., Guo, J., Wang, R., and Yan, J. (2024a). From distribution learning in training to gradient search in testing for combinatorial optimization. *Advances in Neural Information Processing Systems*, 36.

[Li et al., 2024b] Li, Z., Guo, J., and Si, X. (2024b). G4SATBench: Benchmarking and advancing SAT

solving with graph neural networks. *Transactions on Machine Learning Research*.

[Li and Si, 2022] Li, Z. and Si, X. (2022). NSNet: A general neural probabilistic framework for satisfiability problems. *Advances in Neural Information Processing Systems*.

[Marques-Silva et al., 2021] Marques-Silva, J., Lynce, I., and Malik, S. (2021). Chapter 4. Conflict-Driven Clause Learning SAT Solvers. In *Frontiers in Artificial Intelligence and Applications*. IOS Press.

[Mateescu, 2011] Mateescu, R. (2011). Treewidth in industrial sat benchmarks. Technical report, Technical Report MSR-TR-2011-22, Microsoft Research.

[Newman, 2018] Newman, M. (2018). *Networks*. Oxford university press.

[Ozolins et al., 2022] Ozolins, E., Freivalds, K., Draguns, A., Gaile, E., Zakovskis, R., and Kozlovics, S. (2022). Goal-aware neural SAT solver. In *International Joint Conference on Neural Networks*.

[Selman et al., 1994] Selman, B., Kautz, H., and Cohen, B. (1994). Noise strategies for improving local search. In *The AAAI Conference on Artificial Intelligence*.

[Selsam and Bjørner, 2019] Selsam, D. and Bjørner, N. (2019). Guiding high-performance sat solvers with unsat-core predictions. In *Theory and Applications of Satisfiability Testing–SAT 2019: 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9–12, 2019, Proceedings 22*. Springer.

[Selsam et al., 2019] Selsam, D., Lamm, M., Bünz, B., Liang, P., de Moura, L., and Dill, D. L. (2019). Learning a SAT solver from single-bit supervision. In *International Conference on Learning Representations*.

[Shi et al., 2023] Shi, Z., Li, M., Liu, Y., Khan, S., Huang, J., Zhen, H.-L., Yuan, M., and Xu, Q. (2023). Satformer: Transformer-based unsat core learning. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE.

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

[Sinz et al., 2003] Sinz, C., Kaiser, A., and Küchlin, W. (2003). Formal methods for the validation of automotive product configuration data. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*.

[Sonobe, 2016] Sonobe, T. (2016). Cbpenelope2016, ccspenelope2016, gulch at the sat competition 2016. *SAT Competition*.

[Walsh et al., 1999] Walsh, T. et al. (1999). Search in a small world. In *Proceedings of IJCAI*, volume 99. Citeseer.

[Wang et al., 2023] Wang, W., Hu, Y., Tiwari, M., Khurshid, S., McMillan, K., and Miikkulainen, R. (2023). NeuroBack: Improving cdcl sat solving using graph neural networks.

[Wetzler et al., 2014] Wetzler, N., Heule, M. J., and Hunt Jr, W. A. (2014). Drat-trim: Efficient checking and trimming using expressive clausal proofs. In *International Conference on Theory and Applications of Satisfiability Testing*. Springer.

[Xu et al., 2018] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

[Yehuda et al., 2020] Yehuda, G., Gabel, M., and Schuster, A. (2020). It's not what machines can learn, it's what we cannot teach. In *International conference on machine learning*. PMLR.

[You et al., 2019] You, J., Wu, H., Barrett, C., Ramanujan, R., and Leskovec, J. (2019). G2SAT: Learning to generate SAT formulas. In *Advances in Neural Information Processing Systems*.

[Zhang et al., 2022] Zhang, C., Zhang, Y., Mao, J., Chen, W., Yue, L., Bai, G., and Xu, M. (2022). Towards better generalization for neural network-based SAT solvers. In *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*.

[Zhang et al., 2020] Zhang, W., Sun, Z., Zhu, Q., Li, G., Cai, S., Xiong, Y., and Zhang, L. (2020). NLocalSAT: Boosting local search with solution prediction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*.

[Zhang et al., 2021] Zhang, Z., Xu, D., and Zhou, J. (2021). A structural entropy measurement principle of propositional formulas in conjunctive normal form. *Entropy*, 23(3).

[Zulkoski et al., 2018] Zulkoski, E., Martins, R., Wintersteiger, C. M., Liang, J. H., Czarnecki, K., and Ganesh, V. (2018). The effect of structural measures and merges on sat solver performance. In *Principles and Practice of Constraint Programming: 24th International Conference, CP 2018*. Springer.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Not Applicable]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Materials

A LIMITATIONS AND FUTURE WORK

Despite being the first dataset to more deeply analyse SAT structure in terms of GNN generalisation, there are still a few limitations and many possibilities to consider for future work. Firstly, StructureSAT is separated into subsets, where each subset has one domain and focuses on one type of feature at a time. While useful for identifying individual factors of influence at a property level, a multivariate analysis of features and domains could be considered in future work. Secondly, due to certain problems being fundamentally intractable, we do not consider some other CNF based features. The structure calculations in the distribution plots are limited to a 60 seconds timeout as most industrial and G2SAT generative processes are computationally hard to compute. Exploring effective algorithms of computing the structural properties on SAT, especially on large problems, could be a potential future direction. Thirdly, although we include SATCOMP2007 in our dataset and calculate structural values of individual problems, the models we trained could not be efficiently tested on large industrial and G2SAT problems. Thus we only test on 24 G2SAT problems and 12 small industrial problems used to generate the G2SAT problems inside the experiments. However, either better models or more efficient encoding strategies could be included in future work for GNNs to solve large industrial problems. Furthermore, to simplify the comparison process between training and testing structures, we ignored the possible structure changes and losses during the process of encoding and embedding the SAT problems, which should be considered and analysed in future work. Lastly, as we generated 200k pairs of data for each domain and divided the base dataset, there is overlap between different property split groups. Potential future work could be generating datasets with more controllable feature values as the CA generator does [Giráldez-Crú and Levy, 2015].

B FULL ANALYSIS ON DATASET

B.1 Distribution Plot

To get a comprehensive understanding of base dataset domains without splitting, we extend previous experiments and plot 6 other graph property distributions including α_c, H, BE, P_r, T_w and c/v ratio. Since LCG is a bipartite graph, some calculations have been adjusted to accommodate its bipartite characteristics. For example, the clustering coefficient in P_r has been modified to use the bipartite clustering coefficient calculation.

The figures show a wide variety of distributions over graph properties among different domains. Some distinguishable differences can be seen within Q, where random, crafted, and industrial problems have low, medium, and high values, respectively. Additionally, industrial and crafted domains have a wider range of D_f values than random domains. In Fig. 2, the value range of B_e ranks from lowest to highest in crafted, industrial, and random domains, while industrial problems have highest H values. For available P_r values in Fig. 3, the values from random domains 3-SAT and SR are close to 1, while the pseudo-industrial problems CA and PS have higher P_r. While SR has been experimented to be the domain with the best out-of-domain generalisation ability among all the mentioned domains, it does not have the widest coverage of property values. E.g., SR domain is narrower in H compared with other domains. All the distinct characteristics within the dataset highlight the potential abilities of affecting generalisation differently within GNNs.

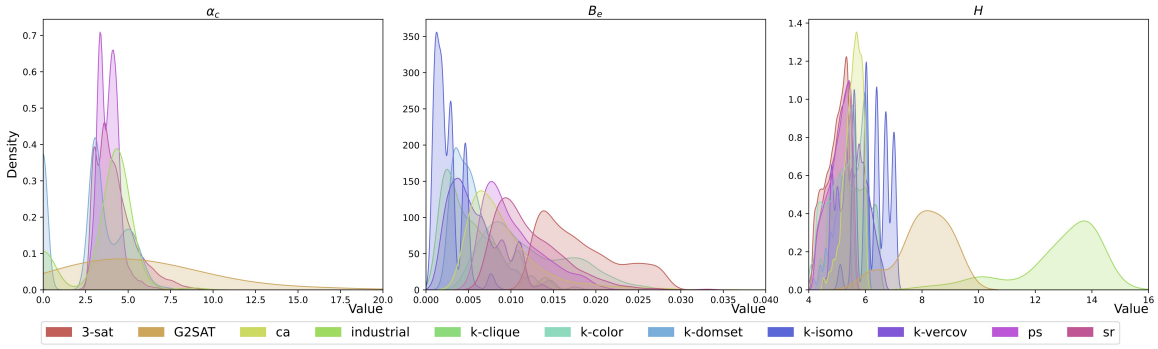

Figure 2: Distribution of various structural graph properties from different problem domains. Left: scale-free measure by clause α_c. Middle: centrality measureBE. Right: entropy measure H.

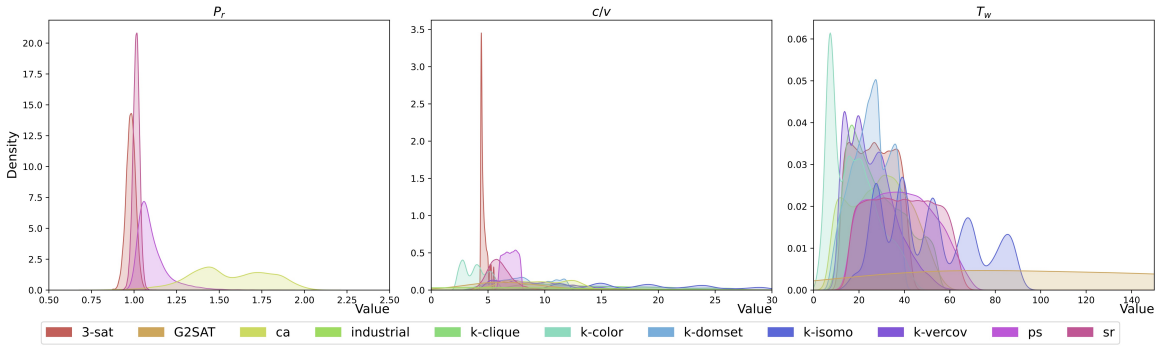

Figure 3: Distribution of various stuctural graph properties from different problem domains. Left: Proximity:P_r. Middle: ratio c/v. Right: Treediwth T_w.

B.2 Structural Plots for Each Domain

Table 8 contains average structure values of the used testing set from D_{t3}. We also show the average structure values used for splitting in the training and validation sets in Table 7. Note that this is the average value of all data in the raw datasets, which includes both Satisfiable (SAT) and Unsatisfiable (UNSAT) problems. However as there are differences in SAT and UNSAT problems of each domain due to the inherent randomness of the generation process, the average values between SAT and UNSAT are slightly different. Although the average values do not represent the overall structure and distribution for each domain, it is an estimation of the calculated structure for the problems and is used as the main metric in our experiments.

Figs. 5 to 11 present the correlation relationships between each of the calculated graph based properties, including number of variables and number of clauses, from the raw 200k datasets for each domain (datasets without splits). Each domain contains unique sturcture and different relationship patterns with the other structures, and some of those patterns are consistent across all domains. E.g., higher number of clauses always results in a lower entropy value. Some of the patterns are consistent with GNN generalisation results while others show opposite. For instance, when looking at the SR domain, higher T_w values results in higher H values, which is consistent with the pattern that both higher value of T_w and H results in better overall out-of-domain results. However, while higher T_w would also results in generally lower D_f, higher valued D_f group reaches better results on NeuroSAT.

Table 7: Mean structural properties of StructureSAT testing domains. N/A indicates not available.

	D_f	α_v	α_c	T_w	B_e	Q	H
SR (10-40)	3.39	4.25	10.03	39.12	0.012	0.38	5.01
r-3SAT	2.57	7.54	N/A	26.53	0.018	0.45	4.92
k-clique	3.67	13.44	N/A	28.19	0.005	0.47	5.51
k-dominating-set	2.88	5.44	2.95	25.03	0.006	0.49	5.46
k-vertex-cover	2.89	6.29	N/A	25.24	0.006	0.50	5.58
k-coloring	2.00	2.92	N/A	17.73	0.02	0.64	5.13
automorph	3.51	17.47	N/A	51.25	0.0026	0.49	6.28
CA	2.92	12.13	N/A	29.39	0.009	0.73	5.56
PS	3.27	5.49	3.87	38.66	0.01	0.36	5.06
G2SAT	3.06	5.30	5.56	116.87	N/A	0.87	8.09
industrial	2.68	10.09	3.50	N/A	N/A	0.87	12.49

Table 8: Mean structural properties of StructureSAT raw domains. N/A indicates not available.

	D_f	α_v	α_c	T_w	B_e	Q	H
SR (10-40)	3.00	4.25	10.03	39.23	0.01	0.38	5.01
r-3SAT	2.57	7.52	N/A	26.53	0.02	0.45	4.92
k-clique	3.26	14.23	N/A	29.93	0.005	0.47	5.61
k-dominating-set	2.89	5.44	3.01	25.33	0.01	0.49	5.48
k-vertex-cover	2.91	6.29	N/A	25.47	0.01	0.50	5.60
k-coloring	2.38	2.92	N/A	17.73	0.02	0.64	5.56
CA	2.92	12.16	N/A	29.36	0.01	0.73	5.56
PS	3.27	5.49	3.87	38.71	0.01	0.36	5.06

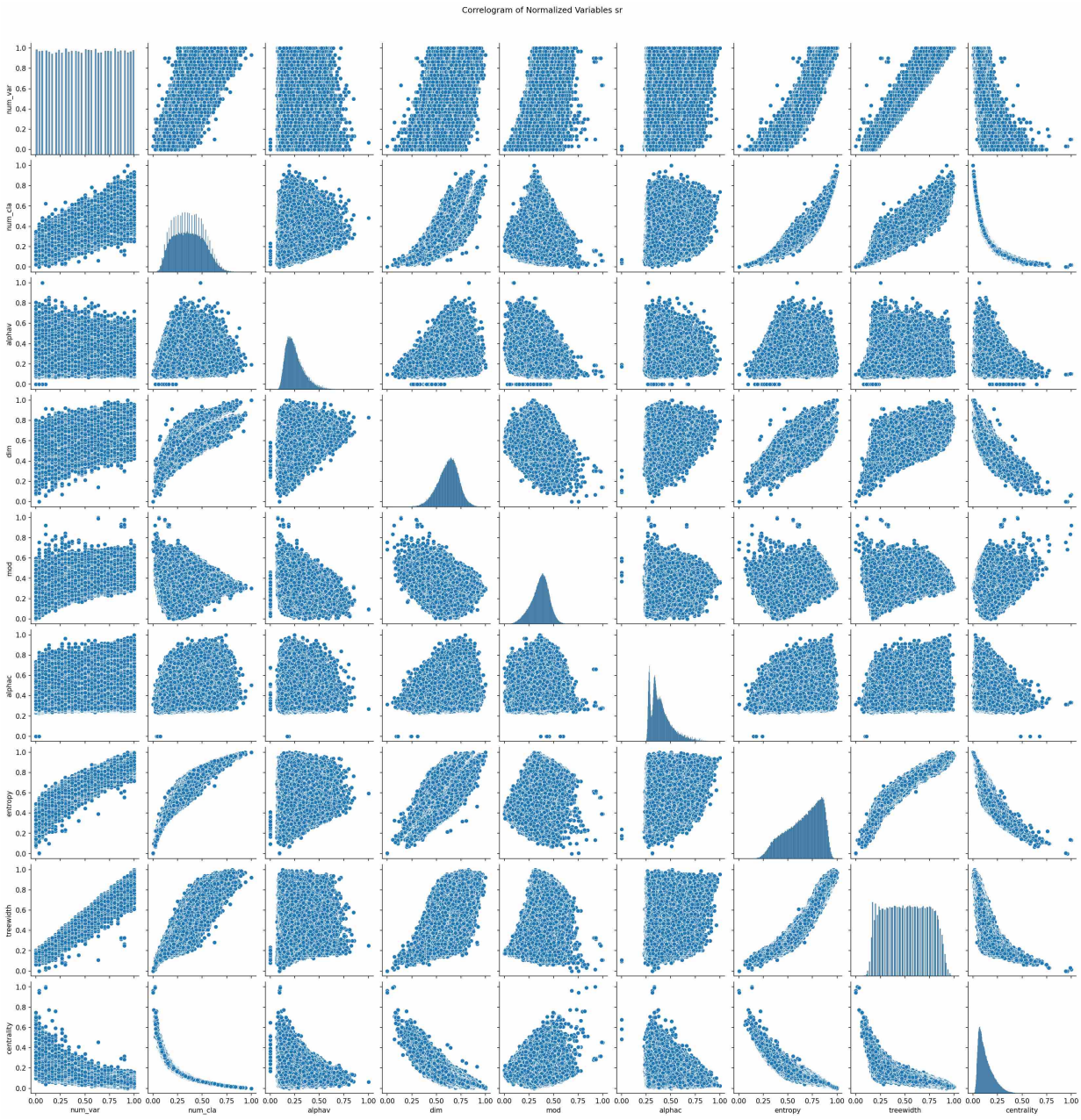

Figure 4: SR correlogram.

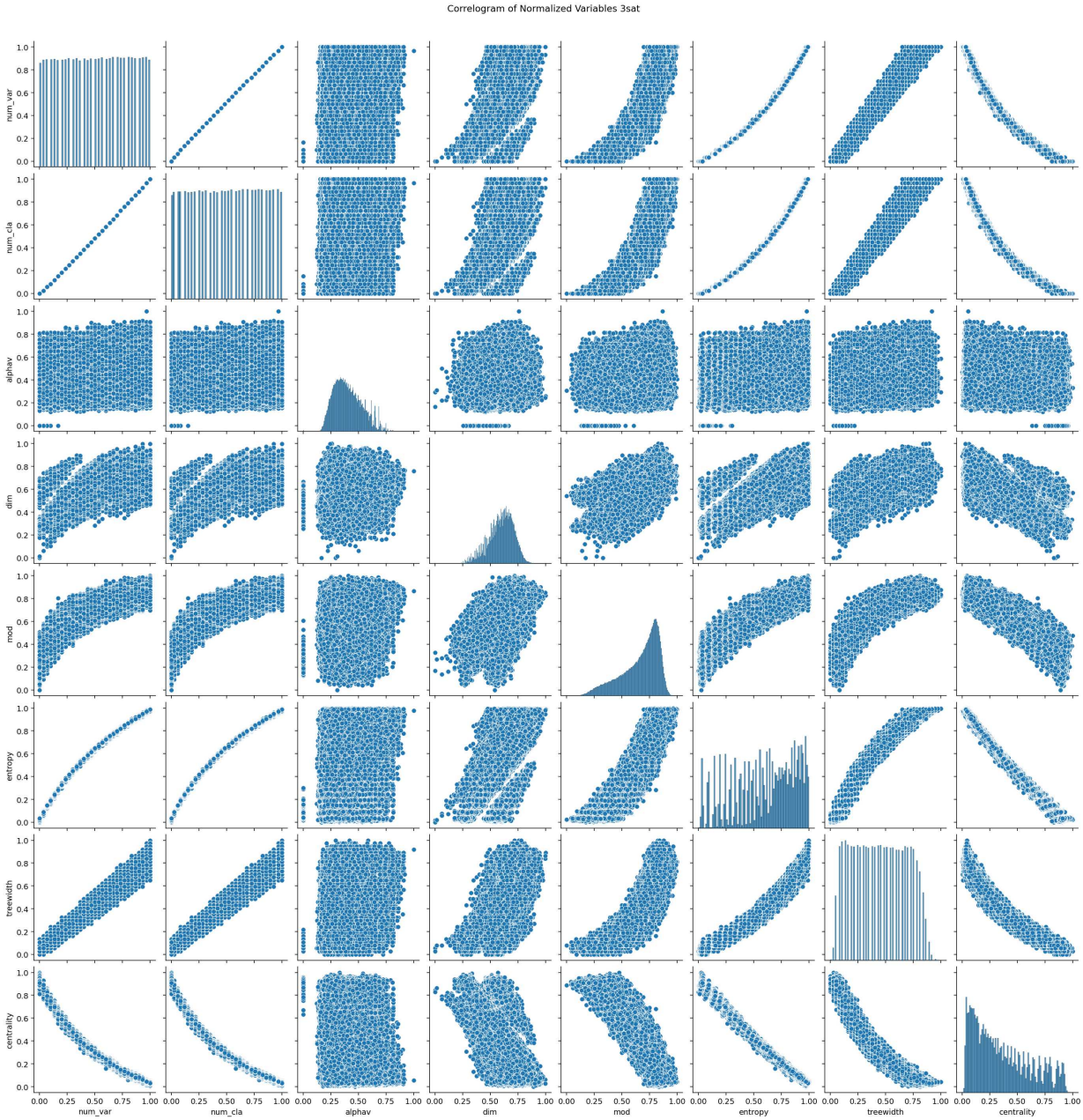

Figure 5: Random 3-SAT correlogram.

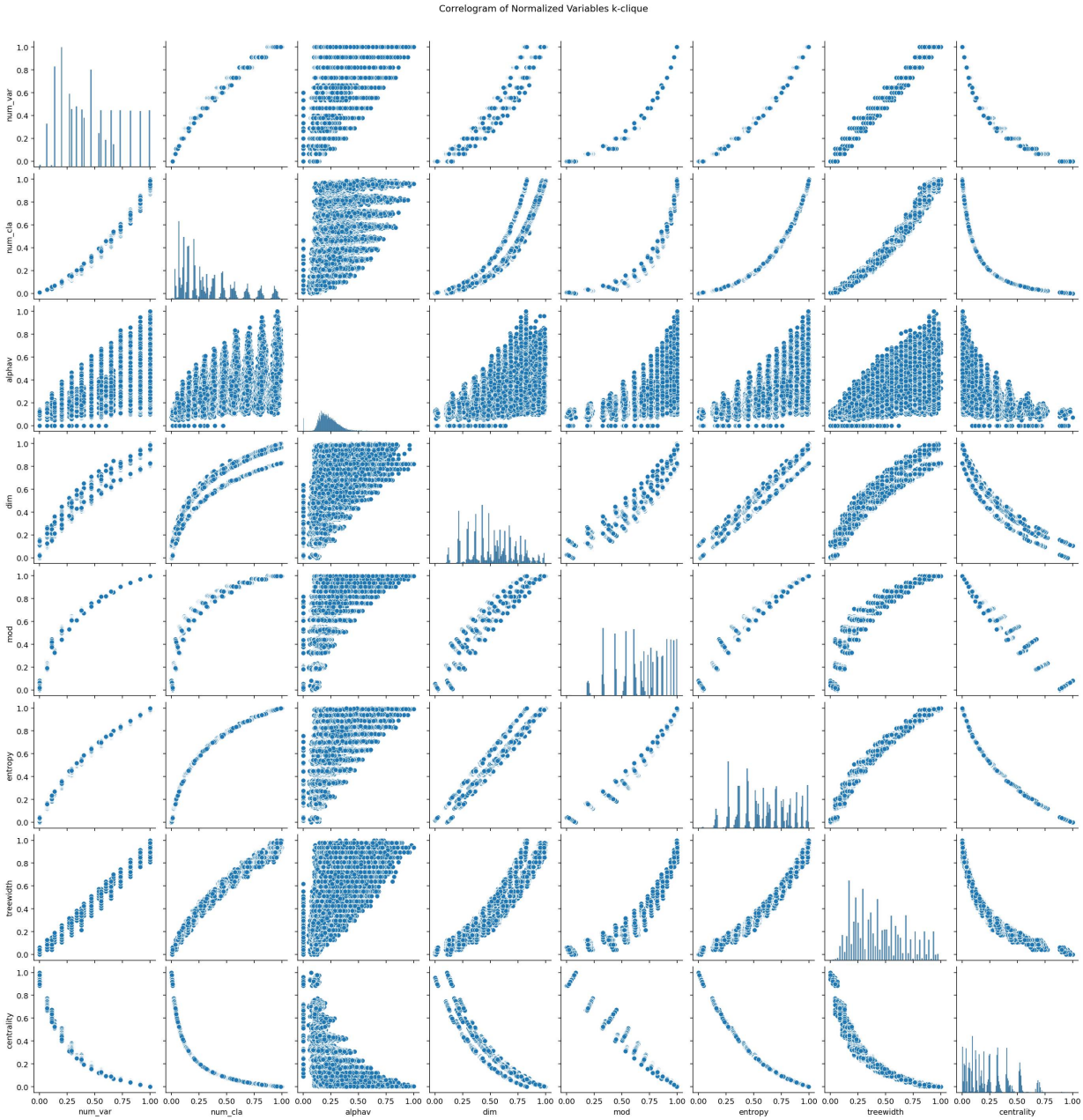

Figure 6: k-clique correlogram.

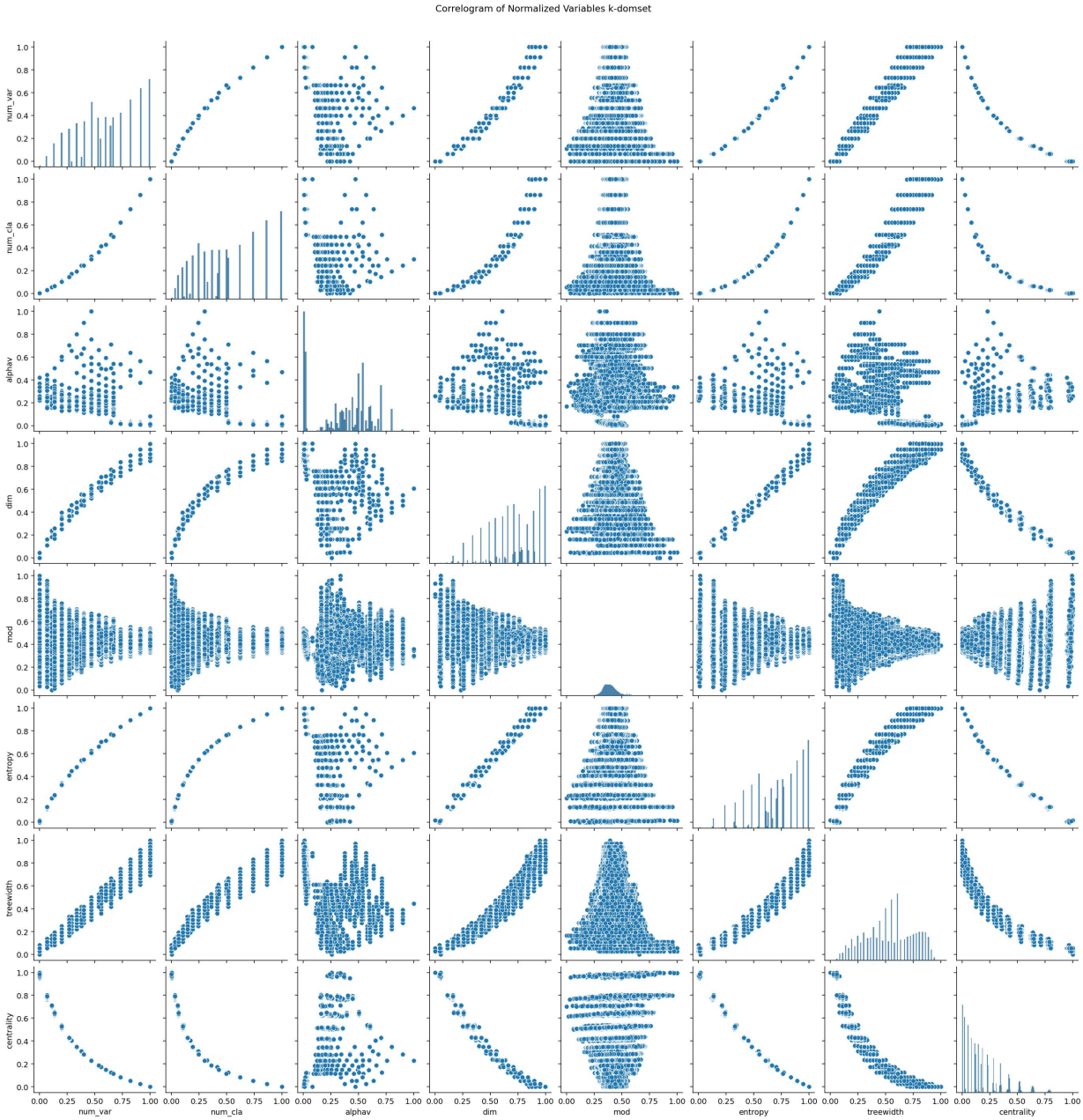

Figure 7: k-domset correlogram.

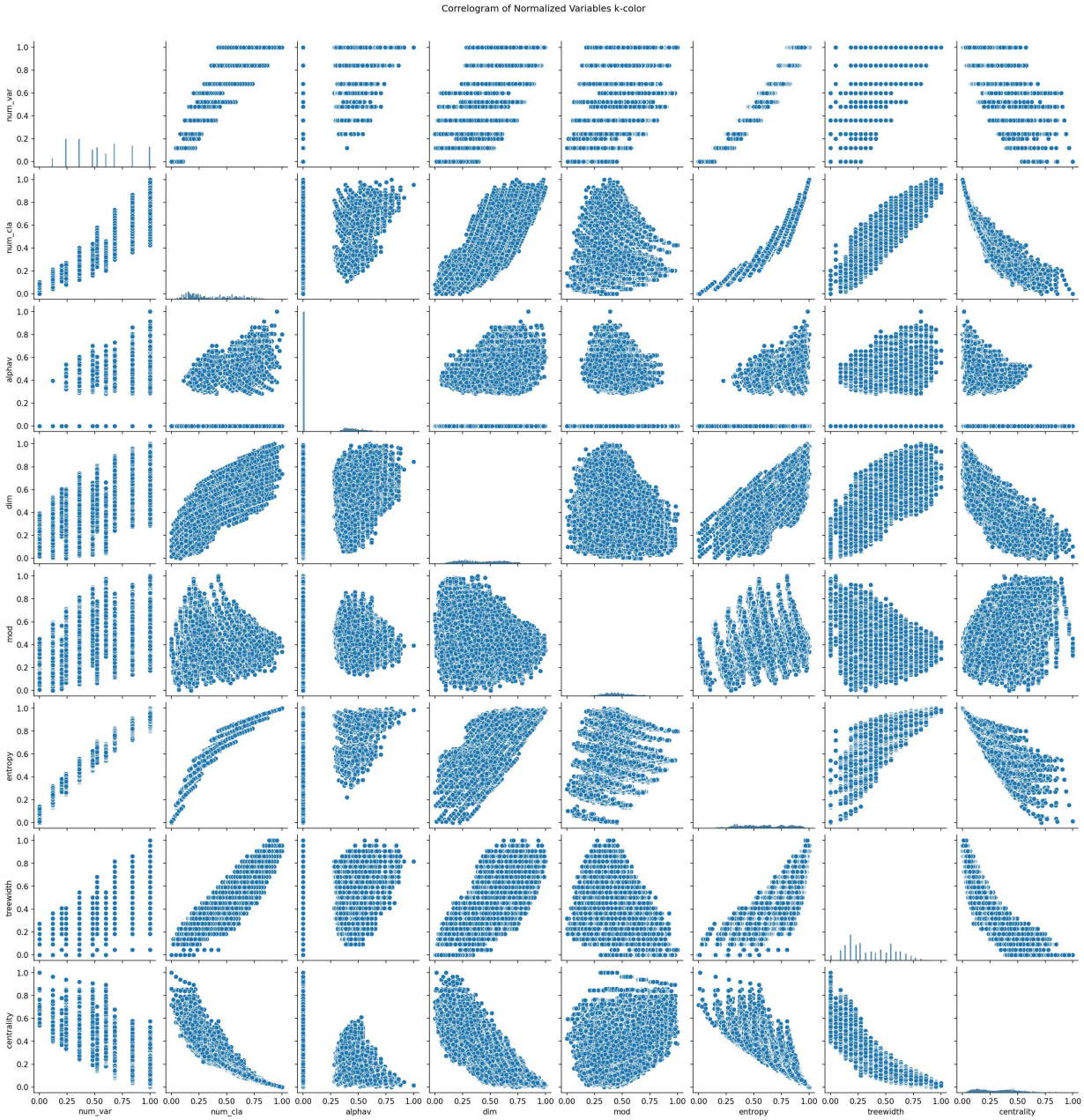

Figure 8: k-color correlogram.

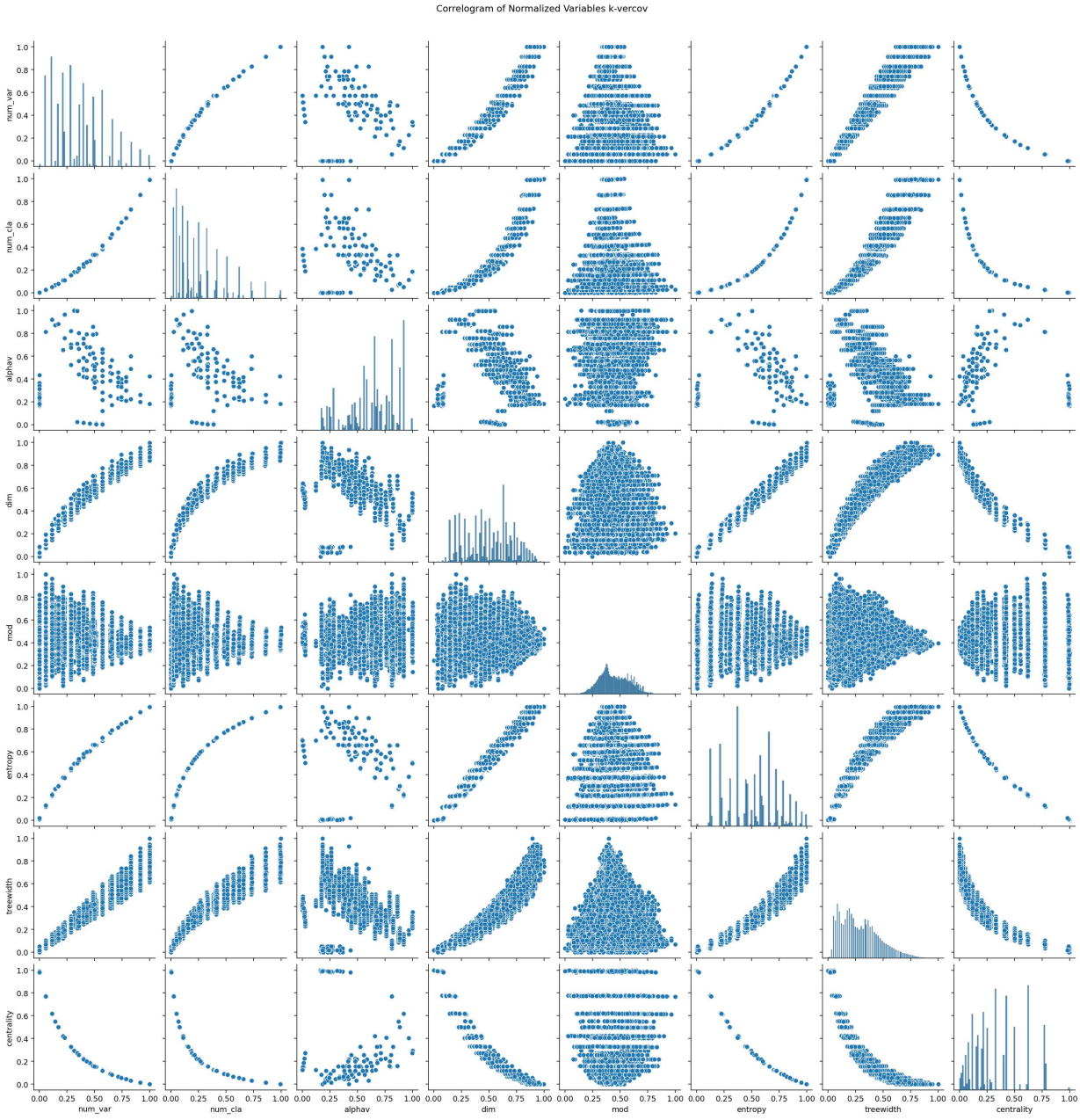

Figure 9: k-vercov correlogram.

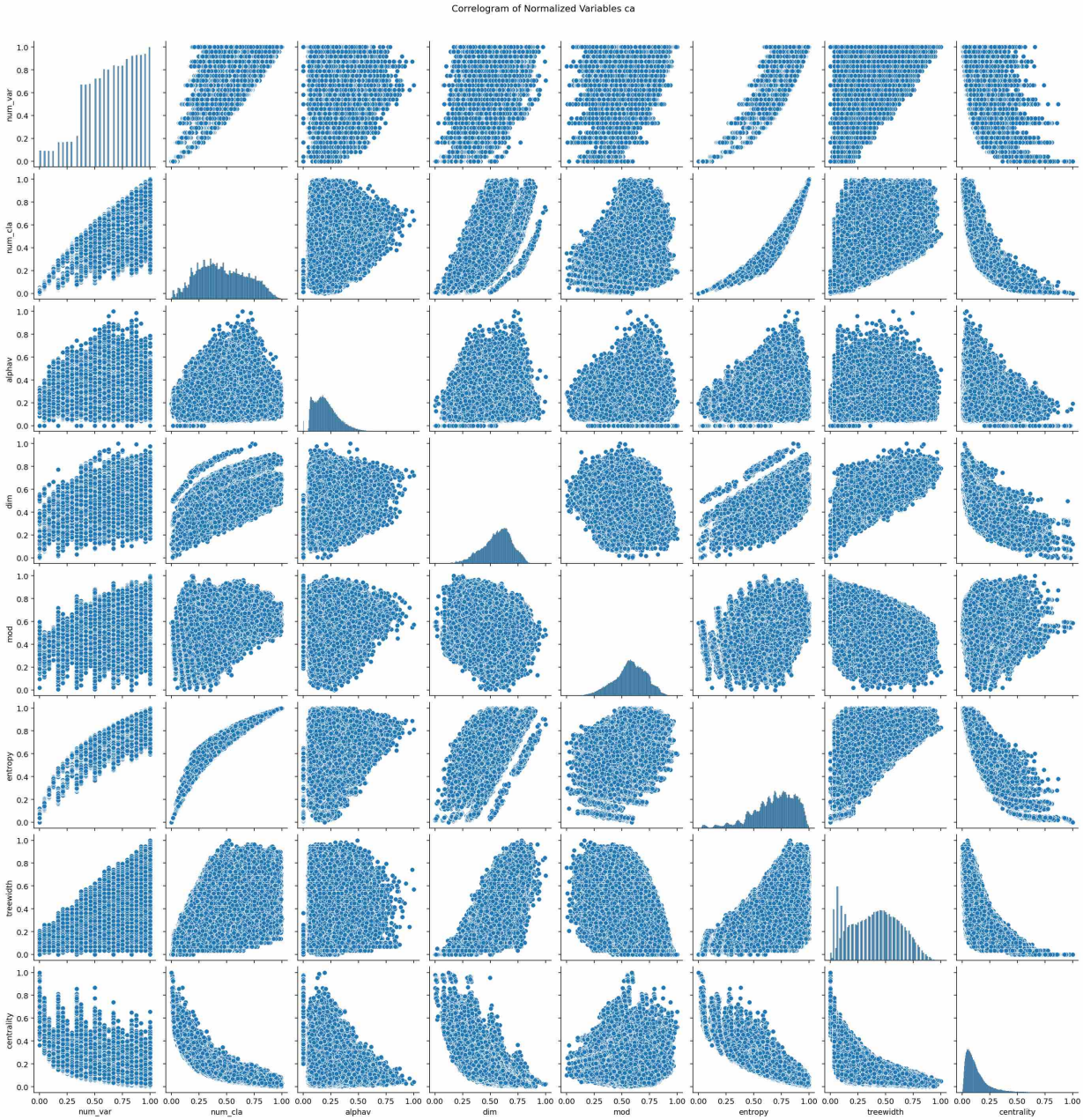

Figure 10: ca correlogram.

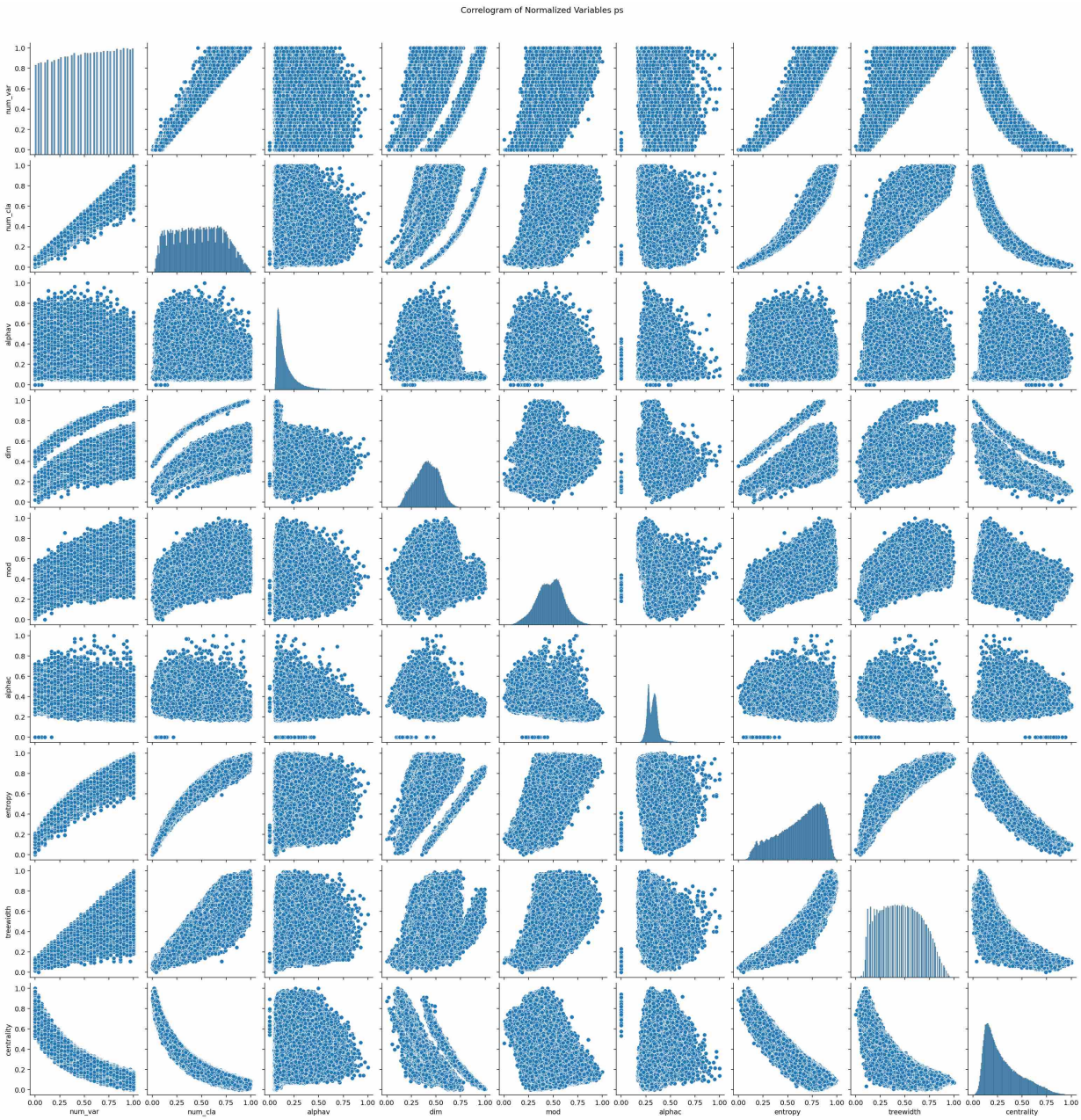

Figure 11: ps correlogram.

C ADDITIONAL EXPERIMENTAL DETAILS

C.1 GNN Baseline

Models used in this work include NeuroSAT [Selsam et al., 2019], GCN [Kipf and Welling, 2016] and GIN [Xu et al., 2018].

C.2 Code Base, experiment set up, and license

Our Dataset and Code base are available in[2]. The link also include detailed instruction on downloading and running experiment with the dataset.

Experiments with GNN are done using existing work from [Li et al., 2024b]. Experimental parameters include $1e-04$ learning rate, $1e-08$ and $1e-07$ weight decay and 32 number of message passing iterations. Batch sizes are selected from 128, 32, 16. All the experiments are run on a machine with a NVIDIA A4500.

StructureSAT is openly licensed via CC BY 4.0.

D ADDITIONAL EXPERIMENTAL RESULTS

This section details some additional experimental results and analysis.

D.1 In-domain Generalisation

D.1.1 In-domain Generalisation to Larger Problem

In Table 9, we train NeuroSAT models using R3, then tested on larger problems with more number of variables(40-100, 100-200). All problems are generated with the ratio c/v at the phase transition. In general, in both medium sized testing sets and large sized testing sets, the bigger splitted groups perform better. This matches with the pattern shown in Figure 4, where larger number of variables results in a larger value in almost all structures focused in Table 9. Thus, from the results we get in the in-domain generalisation experiments so far, we could conclude that for random problem domains, the change in structure values in training set could results in different testing performance to larger problems, where bigger training domain values are almost always better.

Table 9: NeuroSAT testing results. For each metric, we train NeuroSAT on the small and big split of the corresponding R3 training sets. We then test the trained model on the testing data of 10-40, 40-100, and 100-200 variables for each metric. Better training split performance in bold.

Train Split	Test Set	Metric				
		D_f	α_v	T_w	Q	H
small	40-100	78.0	81.0	78.9	66.3	72.0
big		82.0	83.0	81.4	81.4	81.7
small	100-200	70.7	76.0	72.6	54.4	60.2
big		75.2	75.2	74.1	72.7	74.9

D.2 Out-of-domain Generalisation

D.2.1 Out-of-domain Generalisation to Problem with Different Backbones

To further gain insights into Q3 regarding CNF-based properties, we trained a NeuroSAT model using the R3 dataset and conducted tests on the satisfiable problems of different domains, segmented according to their backbone sizes as shown in Table 10. For R3, PS, CA and KCL domains on NeuroSAT, the results indicate that the

[2]Our dataset and codebase are available here.

performance is better on problems with smaller backbone sizes compared to those with larger ones. In contrast, for SR, KD and KI, KV domains, the testing results are more consistent between the two splits, with larger backbone sizes achieving slightly higher accuracy for specific measures like KD and KI. These observations underscore the variability in how backbone size influences performance across different problem domains, suggesting that backbone size, as a proxy for problem complexity within CNF structures, plays a significant role in the effectiveness of GNN-based SAT solvers.

Table 10: Testing accuracy on problems with different backbone sizes. Model is trained on R3 dataset on NeuroSAT. Higher accuracy between big and small split is highlited in bold.

Domain	R3	SR	PS	CA	KCL	KD	KV	KI
big	89.5	**56.7**	62.1	81.1	86.2	**100**	100	**83.7**
small	**98.2**	53.6	**92.4**	**87.4**	**100**	98.0	100	74.2

D.3 Out-of-domain Generalisation on Graph-based Properties

In this section, we extend previous experiments on out-of-domain generalisation test to other domains. Results are shown in Table 11. As discussed earlier, the general out-of-distribution testing performances vary depending on the training set, training structure focus and model used. The extend varies between structures as well, with Q having a bigger difference influence on models trained using PS than using KCO overall. To some domains such as k-color, structural splits does little effect on the overall generalisation results across all three domains, while to domains like R3, α_v influences GIN more than GCN. Note that all experiments are run on a testing iteration of 32. Higher testing iterations could results in better generalisation, while the extent also varies depending on the model and domains. E.g., testing on a 200 number of iterations results in a 1% increase in accuracy on KCO. Future work could be analysing the effect of higher iteration to the learning and embedding of the graph and its corresponding structure.

Furthermore, we trained NeuroSAT on SR(10-40) with different structural splits on satisfying assignment prediction task, which is considered a node classification task. Results in Table 12 also indicate the importance of stucture while finding the correct assignment solution for a problem. The used loss function is UNS_2 following [Li et al., 2024b]. The in-domain resulted accuracy on SR (third column) is similar to the baseline method. As satisfying solution is a much harder task than binary prediction, the overall accuracies on out-of-domain domains are lower than the corresponding accuracies on satisfiability prediction. Some of the test results are reaching less than 10% accuracy, since even getting 1 number wrong in the entire variables of a solution might lead to the problem being unsatisfiable. Nevertheless, the general findings are consistent with the main trends observed in our study: smaller $\alpha_c$demonstrates better generalization performance, while larger Qgeneralize more effectively, which indicate the importance of structure while finding the correct assignment solution for a problem.

Table 11: Accuracy of models trained with different structural splits.

R3	Split	Testing Accuracies		
		NeuroSAT	GIN	GCN
α_v	small	67.5 ± 1.3	63.4 ± 1.1	61.5 ± 0.6
	big	67.8 ± 2.1	67.1 ± 2.3	61.5 ± 0.5
PS	Split	Testing Accuracies		
		NeuroSAT	GIN	GCN
Q	small	55.8 ± 4.0	55.1 ± 0.1	57.3 ± 0.6
	big	53.1 ± 1.0	56.3 ± 0.7	55.1 ± 0.6
KCO	Split	Testing Accuracies		
		NeuroSAT	GIN	GCN
D_f	small	50.2 ± 0.06	50.2 ± 0.6	50.1 ± 0.6
	big	50.5 ± 0.05	49.8 ± 0.2	49.2 ± 0.4
T_w	small	50.1 ± 0.04	50.4 ± 0.2	49.5 ± 0.8
	big	64.5 ± 1.4	50.1 ± 0.1	49.0 ± 0.4
Q	small	50.2 ± 0.04	48.0 ± 0.5	50.3 ± 0.3
	big	50.4 ± 0.2	48.6 ± 0.9	49.1 ± 0.9
H	small	51.4 ± 0.9	50.0 ± 0.9	49.9 ± 1.1
	big	50.3 ± 0.08	50.8 ± 0.8	49.2 ± 0.9

Table 12: Accuracy of NeuroSAT trained on SR(10-40) with different structural splits, task is satisfying assignment prediction.

Metric	Split	Testing Domains										
		SR	R3	KCL	KD	KV	KCO	AM	CA	PS	G2	IN
α_v	small	77.28	75.29	13.23	61.58	58.88	0.00	1.87	59.50	84.55	0.00	0.00
	big	76.82	75.60	24.02	62.02	64.80	0.00	21.42	58.12	84.24	0.00	66.67
α_c	small	78.98	76.57	24.75	63.17	69.87	0.00	3.09	54.22	85.01	0.00	66.67
	big	77.50	73.01	27.17	52.96	58.41	0.00	0.88	44.47	85.11	0.00	66.67
Q	small	76.17	72.21	4.13	43.99	37.87	0.00	1.99	43.11	83.80	0.00	0.00
	big	77.15	74.87	26.03	63.13	66.88	0.00	1.54	60.15	83.53	0.00	33.33