A Pseudo-Boolean Approach to Full Graph Anonymisation

Emke de Groot[1]
Supervisor: Dr. Anna L.D. Latour[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Emke de Groot
Final project course: CSE3000 Research Project
Thesis committee: Dr. Anna L.D. Latour, Dr. Carolin Brandt

Abstract

This work proposes a pseudo-Boolean optimisation model that computes the minimum number of edge deletions needed to fully anonymise a graph, according to the (n, m)-k-anonymity measure. We survey the usability of this model, by comparing it with an unpublished Integer Linear Program (ILP) optimisation model in terms of running time and memory consumption. For both models, we provide a comparison between two different model solvers. We find a noticeable difference in running time and quality of the solution of the different model solvers. The experiments suggest that both running time and memory consumption of solving the pseudo-boolean model are greater than solving the ILP model by a constant factor. This opens up the possibility for the model to be used in tandem with a pseudo-Boolean proof verifier to provide certificates of solution optimality.

1 Introduction

Studies of networks, or graphs, can be highly societally impactful. A social scientist can learn about polarisation on the social platforms by the study of who follows whom on Facebook and we can learn about how information spreads over the internet by the study of hyperlink networks. When network data is publicly available, we enable social scientists to carry out this type of research.

Institutions and companies that hold these graph data are cautious of releasing it to the public, or even privately to researchers, because these graphs can represent privacy sensitive data. This data is a likely target for attackers and releasing it forms a privacy risk. When we look at a graph in which the vertices are users, vertex labels are usernames and edges are friend connections, only removing the vertex labels does not anonymise the graph. An attacker can still identify individual users, for example if they are friends with a unique number of people.

We can measure the anonymity of a network in various ways. Choosing a measure often entails a trade-off between the scope of the anonymity and the complexity of measuring. Taking the complete structural information of the graph into account, prevents an attacker from identifying individual vertices of the graph, but anonymising a graph to this extend is shown to be NP-hard [Cheng *et al.*, 2010]. Measures that only account for information about individual vertices are computationally simple, but insufficiently anonymous for most attacker scenarios [de Jong *et al.*, 2024]. In this work, we use a measure that balances complexity and anonymity by taking the direct neighbourhood of vertices into account [Zhou and Pei, 2008].

To eliminate a privacy risk, we can minimally alter the graphs to anonymise individual vertices by removing, adding or swapping edges. Since de Jong *et al.* [de Jong *et al.*, 2025] have shown that algorithms using edge removal are most promising in terms of both running time and memory consumption, we use edge deletions to anonymise the graphs.

De Jong *et al.* identified three methods of graph anonymisation: full, partial and budgeted anonymisation [de Jong *et al.*, 2025]. Methods for partial anonymisation only focus on anonymisation of a fraction of all vertices in a graph. Budgeted anonymisation aims to maximise the amount of anonymous vertices in the output graph by only removing some amount of edges, according to the budget. Full graph anonymisation ensures that all vertices in the graph are anonymised, while removing as few edges as possible. Fully anonymising a graph is equivalent to repeatedly solving the budgeted anonymisation problem until one of the optimum solutions is found. This means that full anonymisation is more computationally expensive than the budgeted version.

Multiple methods exist that partially anonymise a graph using edge deletions [Xie, 2023; de Jong *et al.*, 2025]. Efficient algorithms that fully anonymise a graph exist, but only anonymise a graph for the simplest attacker scenarios [Casas-Roma *et al.*, 2013]. There does not exist a state of the art for full graph anonymisation methods that use a measure that is realistic for an attacker scenario. In an unpublished research note, Latour proposed an Integer Linear Program approach to full graph anonymisation using such a measure [Latour, 2024], but it has not been compared to other methods thus far.

To ensure that we have to remove at least a certain amount of edges to fully anonymise a graph, we have to prove that our full graph anonymisation method arrived at an optimum solution. Similarly, to ensure that a graph cannot be fully anonymised within some budget, we have to prove that our budgeted graph anonymisation did not miss any fully anonymous solutions. Because model solvers are very complex in nature, there is always a possibility that some reasoning step was not correct. When this is the case, we cannot guarantee that we have found the optimum solution.

To prove that we arrived at an optimum solution, we have to prove that every step we took to arrive at that solution was correct. We can only provide a *certificate* to show that the solution is optimal, if we can mathematically verify every inference step that a solver takes. Certification is a difficult problem, since the inference steps need to be small enough to prove, but small inference steps slow down a solver. Additionally, we need to invent new methods for certification for every new algorithm or network.

We can, however, provide a certificate for any problem if it is modelled as a *pseudo-Boolean* problem. The log provided by the solver of this model can then be used as input for an independent proof verifier, such as VeriPB [Gocht, 2025]. This verifier is able to provide a certificate, without any knowledge of the actual problem. This allows us to create a certificate for any problem modelled as pseudo-Boolean, as has been done for subgraph isomorphism [Gocht *et al.*, 2020].

We introduce a *pseudo-Boolean* (PB) variant on an existing *Integer Linear Program* (ILP) approach [Latour, 2024] to full graph anonymisation, using the graph anonymity measure described by Xie [Xie, 2023] and minimising the number of edge deletions.

The main research question of this thesis is as follows:

How does a Pseudo-Boolean approach to full network anonymisation compare to a Integer Linear Program approach in terms of solving time, memory consumption and quality of the solution on networks with differing graph topologies?

We introduce necessary preliminaries in Section 2, such as definitions, notes on notation, an example graph and the formal problem statement. In Section 3, we introduce the new PB approach, supported by an explanation of the conversion to PB constraints. The detailed sub-research questions, experimental setup and results can be found in Section 4. We conclude this paper in Section 5, including the limitations and some notes about future work. In Section 6, we reflect on the ethics and reproducibility of this research.

2 Preliminaries

We introduce definitions of the concepts used in this work and lay out our assumptions. The definitions are then used for the problem statement and example in Subsection 2.3.

2.1 Graphs and Anonymity Measures

We consider undirected, unweighted, self-loop free graphs $G = (V, E)$, with vertices V and edges E. Single vertices are denoted using lower-case letters. Edges are denoted as sets of two vertices, meaning that if there exists an edge between vertices u and v, we can write either $(u, v) \in E$ or $(v, u) \in E$. The distance $dist(u, v)$ between two vertices $u, v \in V$ denotes the length of the smallest path between vertices u and v. If no such path exists, $dist(u, v) = \infty$. For all $v \in V$, we define $dist(v, v) = 0$.

The degree $\deg(v)$ of a vertex v is then the number of vertices u for which $dist(u, v) = 1$. This is equivalent to the size of $V_1(v) \setminus \{v\}$. We let $\delta_G := \max_{v \in V}(\deg(v))$ denote the maximum degree of a graph G.

Let $\Theta := \{(u, v, w) \mid (u, v), (v, w), (u, w) \in E;\ u \neq v \neq w\}$ denote the set of all triangles in G. We say that a triangle $(u, v, w) \in \Theta$ is only incident to the vertices u, v and w. $Tri(v)$ then denotes the set of all triangles incident on vertex v. We let $\theta_G := \max_{v \in V}(Tri(v))$ denote the maximum number of incident triangles per vertex of graph G.

Definition 1 (d-Neighbourhood)**.** We define the d-Neighbourhood of a vertex u as $N_d(u) = (V_d(u), E_d(u))$, where $V_d(u)$ consists of all $v \in V$ with $dist(u, v) \leq d$. We use $E_d(u)$ to denote the set of all edges $(v, w) \in E$, for which $v, w \in V_d(v)$.

Definition 2 ((n, m)-k-Anonymity)**.** All vertices $v \in V$ have a (n, m)-signature, where n is the degree of v and m is the number of triangles incident to v. We consider some graph G to be (n, m)-k-Anonymous, if for all $n, m \in \mathbb{N}_0^+$, there are either zero or at least k vertices in G with a (n, m)-signature, assuming that $k \in \mathbb{N}^+$.

2.2 Variables and Constraints

We restrict constants in constraints to integers and limit the domain of variables to non-negative integers. We then define a constraint as some combination of variables, constants, elementary arithmetic operators ($\{+, -, \cdot, \div\}$), classical relational operators ($\{=, >, \geq, <, \leq\}$) and logical connectives ($\{\wedge, \vee, \Rightarrow, \Leftarrow, \Leftrightarrow\}$. A literal x can furthermore be negated as $\overline{x}$.

We can *reify* some constraint c by setting it as the logical equivalence of some literal x. Reified constraints thus take on the following form: $x \Leftrightarrow c$.

Definition 3 (Pseudo-Boolean Constraints)**.** A linear pseudo-Boolean (PB) constraint is a constraint that has the following form [Biere *et al.*, 2021]:

$$\sum_j a_j l_j \bowtie b$$

where a_j and b are integer constants, l_j are literals and $\bowtie$ is one of the classical relational operators. We do not consider non-linear pseudo-Boolean constraints in this thesis.

In polynomial time, we are able to transform all linear pseudo-Boolean constraints to normalised pseudo-Boolean constraints of the following form [Barth, 1995]:

$$\sum_{j=1}^{n} a_j l_j \geq b, \quad n, a_j, b \in \mathbb{N}_0^+$$

2.3 Problem Statement

We formally define the graph anonymisation problem that we aim to solve in this work in Definition 4. To illustrate this further, we give a instance of an input graph and the solution in Example 1. In Figure 1, we provide a visualisation of this instance.

Definition 4 (Graph Anonymisation Problem)**.** Given a undirected, unweighted, self-loop free input graph $G = (V, E)$, and some $k \in \mathbb{N}^+$, find the minimum number of edge deletions ϵ needed to create a (n, m)-k-anonymous graph. There then exists some (n, m)-k-anonymous output graph $G' = (V, E')$, with $E' \subseteq E$ and deleted edges $D' = E \setminus E'$, such that there does not exist a (n, m)-k-anonymous graph $G'' = (V, E')$, with $E'' \subseteq E$, deleted edges $D'' = E \setminus E''$ and $|D''| < |D'|$. We then know that $\epsilon = |D'|$.

Example 1. *We let $k = 2$ and consider the graph $G = (V, E)$, with $V = \{a, b, c, d\}$ and $E = \{(a, b), (a, d), (a, c), (b, c), (b, d), (c, d), (c, e), (d, e)\}$. Since only vertex $e \in V$ has the signature $(2, 1)$, G is not (n, m)-k-anonymous and we know that $\epsilon > 0$. We consider the graph $G' = (V, E')$, with $E' \subset E$ and $D' = (a, b)$. Because there does not exist a graph $G'' = (V, E'')$, with $E' \subseteq E$ and $|D''| < |D'|$, we know that $\epsilon = 1$ for G.*

Example 1 is a mock-up of a real-world network. If we let the input network represent following data of some social networking platform, the vertices are the users of the platform and vertices are connected by an edge if they follow each other. In the input network, user e can be uniquely identified. When we remove the follow connection between users a and b, we cannot identify any single user by only their (n, m) signature.

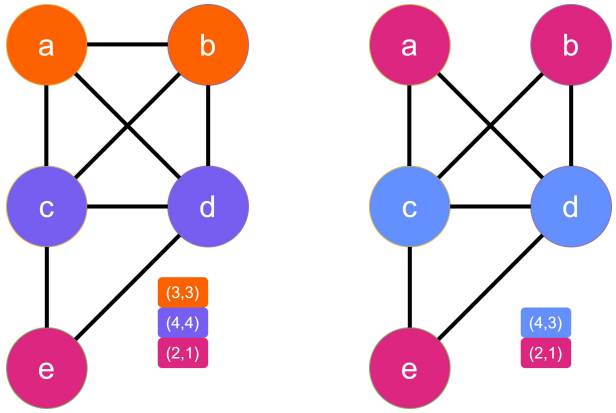

Figure 1: Graphs G (left) and G' (right), as detailed in Example 1, with vertices coloured according to their (n, m)-signatures.

3 Approach

In this section, we explain how certain reified constraints can be converted to PB constraints and then describe our new pseudo-Boolean model, based on the existing ILP encoding of Latour [Latour, 2024]. This should be enough for a complete understanding of the model, but a full formal description of all variables and constraints can be found in the appendix.

3.1 Constraint Conversion

Previous work [Gocht and Nordström, 2021] has detailed the conversion from a reified constraint with normalised PB right-hand side to two normalised PB constraints. This conversion to normalised PB constraints has not been done for reified constraints where the right-hand side is a PB constraint with the relational operator $=$. In this section we describe this conversion, which we used to convert all constraints of the subsection below to PB constraints.

A general form of the aforementioned constraint is used, more formally defined as follows:

$$z \Leftrightarrow \left(\sum_i a_i \ell_i = A\right) \qquad a_i, A \in \mathbb{N}_0^+ \tag{0}$$

We introduce two helper variables b and c and constrain them as follows:

$$b \Rightarrow \left(\sum_i a_i \ell_i < A\right) \tag{1}$$

$$c \Rightarrow \left(\sum_i a_i \ell_i > A\right) \tag{2}$$

The original constraint can now be substituted with the conjunction of the following three constraints:

$$z \Rightarrow \sum_i a_i \ell_i \geq A \tag{3}$$

$$z \Rightarrow \sum_i a_i \ell_i \leq A \tag{4}$$

$$\overline{z} \Rightarrow (b \vee c) \tag{5}$$

Because we can rewrite the right-hand side of the five newly introduced constraints as normalised PB constraints, we can use the conversion from Gocht and Nordström [Gocht and Nordström, 2021] to rewrite constraints 1-5 to the following final normalised PB constraints:

$$\left(-A + 1 + \sum_i a_i\right) \cdot \overline{b} + \sum_i a_i \overline{\ell_i} \geq -A + 1 + \sum_i a_i \tag{6}$$

$$(A+1) \cdot \overline{c} + \sum_i a_i \ell_i \geq A + 1 \tag{7}$$

$$A\overline{z} + \sum_i a_i \ell_i \geq A \tag{8}$$

$$\left(-A + \sum_i a_i\right) \cdot \overline{z} + \sum_i a_i \overline{\ell_i} \geq -A + \sum_i a_i \tag{9}$$

$$z + b + c \geq 1 \tag{10}$$

3.2 Pseudo-Boolean Model

Our pseudo-Boolean model is an adaptation of the aforementioned ILP model. We do not change the meaning of most of the sets of Boolean variables and corresponding constraints. We replace all integer variables and corresponding constraints with sets of Boolean variables and constraints and introduce new variables and constraints to help with PB modelling.

We first introduce some notation to allow us to use the same small optimisations that the ILP model uses. We use δ^* and θ^* as, respectively, an upper bound on the degree and number of triangles incident to any vertex $v \in V$. For each vertex $v \in V$, we also define $\delta_v := \min(\deg(v), \delta^*)$ and $\theta_v := \min(Tri(v), \theta^*)$ as vertex-specific upper bounds.

Variables

Equal to the ILP model, we introduce a Boolean variable $\ell_{u,v} \in L$ for every edge $(u, v) \in E$ in the input graph to indicate if the edge is present in the output graph. This means that $\ell_{u,v} = 1$ only if edge $(u, v) \in E'$. The set of deleted edges D' then only contains some edge $(u, v) \in E$ if $\ell_{u,v} = 0$.

We similarly introduce a Boolean variable $t_{u,v,w} \in T$ for every triangle $(u, v, w) \in \Theta$ in the input graph to indicate if it is present in the output graph.

We replace the integer variables n_v and m_v in the ILP model that respectively describe the number of neighbours and triangles incident to a vertex v in the output graph by two sets of Boolean variables. The first set N, includes a variable $n_{v,i}$ for every vertex $v \in V$ and every value for i up to and including the vertex-specific upper bound on the degree δ_v. We define the set of variables M in the same manner, but let the upper bound be θ_v. The variable $n_{v,i} \in N$ then indicates if vertex v has precisely i neighbours and $m_{v,j} \in M$ then indicates if vertex v has precisely j incident triangles in the output graph.

The Boolean variable $s_{v,i,j} \in S$ indicates if vertex v has precisely i neighbours and j incident triangles in the output graph, equal to the ILP model. We define S in a similar manner to N and M, introducing a variable for every vertex $v \in V$, every value for i up to and including δ_v and every value j up to and including θ_v.

For every $n_{v,i} \in N$ and $m_{v,j} \in M$ we add four helper variables to allow for the PB modelling. For $n_{v,i}$, these are $n_{<,v,i}$, and $n_{>,v,i}$, indicating if vertex v has, respectively, less than or more than i neighbours. The other two helper variables are similarly defined for $m_{v,j}$.

Lastly, we introduce a Boolean variable $z_{i,j,p} \in Z$ for each possible combination of (i,j)-signature and every value p from 1, up to and including k. The possible combinations of (i,j)-signatures only include values of i up to and including the general upper bounds on the degree δ^* and up to and including θ^* for j. Variable $z_{i,j,p}$ indicates if there are at least p vertices in the output graph with a (i,j)-signature.

Constraints

We introduce sets of constraints to enforce the definitions of each of the variables, as given in the previous subsection. For example, for each of the triangle variables, we say that a triangle (u,v,w) can only exist in the output graph precisely if all three edges (u,v), (v,w) and (u,w) exist in the output graph:

$$t_{u,v,w} \Leftrightarrow (\ell_{u,v} + \ell_{v,w} + \ell_{u,w} \geq 3) \qquad t_{u,v,w} \in T$$

We then introduce sets of constraints to ensure that each vertex can only have one degree, one number of incident triangles and one (degree, incident triangle) combination. This is, taking S as an example, specified as follows:

$$\sum_{s_{v,i,j} \in S} s_{v,i,j} = 1 \qquad v \in V$$

Our final set of constraints says that for each possible (n,m)-signature, either zero or at least k vertices in the graph have that signature. This is equivalent to saying that all $z_{i,j,p}$ are equal for the same (i,j)-signature, represented in the following constraint:

$$\sum_{p=1}^{k} z_{i,j,p} = k \cdot z_{i,j,1} \qquad 0 \leq i \leq \delta^*;\ 0 \leq j \leq \theta^*$$

Objective Function

Since we want to minimise the number of deleted edges D', we want to maximise the total number of edges E' in the output graph. In our model, this objective is formally defined as follows:

$$\text{maximise} \sum_{\ell \in L} \ell$$

4 Experiments

In this section, we first describe the sub-research questions and the experimental setup. We end the section by laying out our experimental results.

4.1 Research Questions

In this work, we answer the following four sub-research questions:

RQ1. *How does the running time of the PB approach compare to that of the ILP approach and which graph characteristics have the most influence on running time?*

RQ2. *What are the differences in model size between the ILP and the PB variant for networks with differing graph topologies?*

RQ3. *How does the memory consumption during solving of the PB model compare to that of solving the ILP model and which graph characteristics have the most influence on memory consumption?*

RQ4. *What is the influence of using different model solvers on the running time for this instance of graph anonymisation for both the ILP and PB models?*

4.2 Experimental Setup

To answer our four sub-questions, use the existing ILP implementation, supplement this with an implementation using a different model solver and implement the PB model using the same two model solvers. We use a collection of real-world network data as input for the four implementations and compare the output.

Problem instances

We run the experiments on a set of real-world network data, a selection of 31 animal social networks from a network database [Rossi and Ahmed, 2015]. We choose animal social networks, because with our limitations on computational resources and time, most other real-world networks are too large to solve with model solvers. They also exhibit a great variety in network characteristics. In Table 1, we list statistics about the characteristics of the networks.

For the fourth sub-research question, we only use the 18 smallest networks. The larger networks are not feasible for the experiment of the fourth question, because we can only run the experiments for the first three questions on a high-performance computing system. A full table with all networks and the selection of the smaller networks can be found in the appendix.

Table 1: Statistics about networks used as problem instances in the experiments. The set of small networks is a subset of the set of all networks.

	All networks				Small networks	
	Min	Median	Mean	Max	Median	Max
$\lvert V\rvert$	4	28	40.94	171	17	103
$\lvert E\rvert$	5	67	111.90	378	36	190
$\lvert\Theta\rvert$	0	54	292.06	3276	26	1140
δ_G	3	9	10.94	31	6.5	19
θ_G	0	17	48.45	351	9	171

Software

We implement the pseudo-Boolean model in Python, using NetworkX [Hagberg *et al.*, 2008] for network processing and Gurobi [Gurobi Optimization, LLC, 2024] as the model

solver, similar to the existing ILP implementation.[1] To answer the fourth research question, we create a new implementation in which both the ILP and PB models, encoded using Gurobi, are solved with the SCIP model solver [Bolusani *et al.*, 2024]. For this, Gurobi encodes the ILP model in the MPS format and the PB model in the OPB format. We chose SCIP, because it is a non-commercial solver and is commonly used to solve PB problems [Mexi *et al.*, 2025].

For implementation, we use Python version 3.9.8, SCIP version 9.2.2 and Gurobi version 12.0.0. For both model solvers, we use the default parameter settings of the mentioned versions. The only exception is that for the experiments for the first three research questions, we set a model solving time limit of four hours and limit the memory consumption during solving to 9 GB.

Hardware

We run the experiments for the first three sub-questions on the high-performance computing system DelftBlue [Delft High Performance Computing Centre (DHPC), 2024]. Using three CPUs, with each 3968 MB per run, we run the experiments on nodes equipped with the Intel(R) Xeon(R) Gold 6226R CPU, running at 2.90 GHz. For the fourth sub-question, we run experiments on a device with an Intel i7-9750H CPU, running at 2.6 GHz, with six physical cores.

Independent Variables

To answer our first three sub-questions, we identify five graph characteristics as our independent variables. Given some input network $G = (V, E)$, we can calculate the total number of vertices $|V|$, edges $|E|$ and triangles $|\Theta|$. We can also calculate the maximum degree δ_G and maximum number of incident triangles θ_G. These characteristics are relatively fast to compute compared to other commonly used characteristics such as the *assortativity coefficient* [McNulty, 2022]. We use characteristics related to degrees and number of triangles, as they are closely related to the (n, m)-k-anonymity measure.

We identify the model size as the independent variable to answer the fourth sub-question. The model size does not differ between the same instance on different solvers, because we first encode the model using one model solver and use this as input for the different solvers. We define the model size as its number of variables and number of constraints.

Dependent Variables

We run every problem instance, or input graph, on the four implementations. During each run, we measure the encoding and solving time (in seconds and in CPU seconds) and the number of variables and constraints of the model. We only measure the maximum memory used during optimisation (in GB) for the Gurobi runs. We are then able to compare the running time, model size and memory consumption of the runs to answer the sub-questions.

To answer what graph characteristics have the most influence on the running time, model size and memory consumption, we calculate the Pearson correlation coefficients [Pearson, 1895] between each of the independent variables and the dependent variables. From this, we will be able to conclude which of the chosen graph characteristics is the best linear predictor of each of the dependent variables.

To be able to compare the average running times between the implementations, while including runs that have reached the time limit, we use the *Penalized Average Runtime* score, with a factor of two (PAR2) [Kerschke *et al.*, 2018].

4.3 Results

In this subsection, we describe our experimental results, as they relate to each of the sub-research questions. For question two, we present both empirical and experimental results.

RQ1

For all networks, the encoding time is slower for the PB model than the ILP model, as can be seen in Figure 2. This is the case for the running time in wall-clock seconds and in CPU seconds. When we look at the model solving time, visualised in Figure 3, we can see that the points are on and around the midline. This means that we cannot conclude that the solving time is larger for either the ILP or PB model. We note that for one of the networks, Gurobi reached the time limit when solving the PB model, but not the ILP model. None of the solving instances reached the memory limit. We also note that for two of the largest networks, the solving time of the PB approach is shorter than that of the ILP approach.

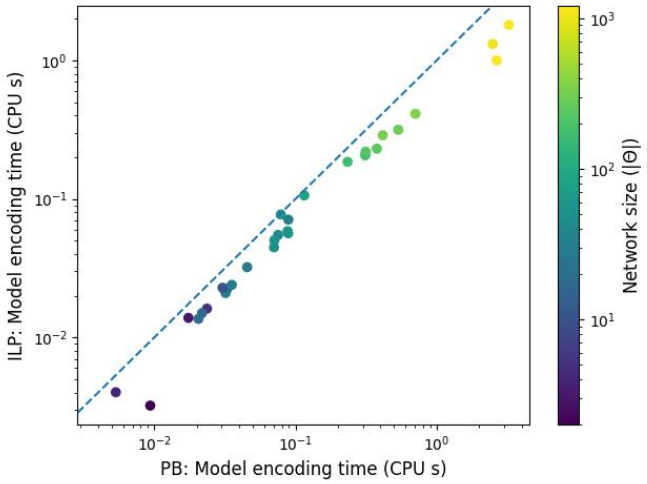

Figure 2: Comparison of the model encoding time in CPU seconds between the ILP and PB approach. Each point represents one network, coloured according to its size in number of triangles.

When we look at the average times to encode and solve the models in Table 2, the running time for the ILP approach is lower for all measures. We also notice that the average encoding time takes at most 0.006% of the total running time. However, the PB model takes 74.06% more time in wall-clock seconds and 83.16% more time in CPU seconds to encode than the ILP model. We can compare this to the difference in solving time, where the PB model takes only 6.00% more time in wall-clock seconds and 2.23% more time in CPU seconds.

We map the colours of the points in Figures 2 and, to different measures of network size. These measures correspond

[1]The implementation of both models is available at github.com/Emkedg/network-anonymisation.

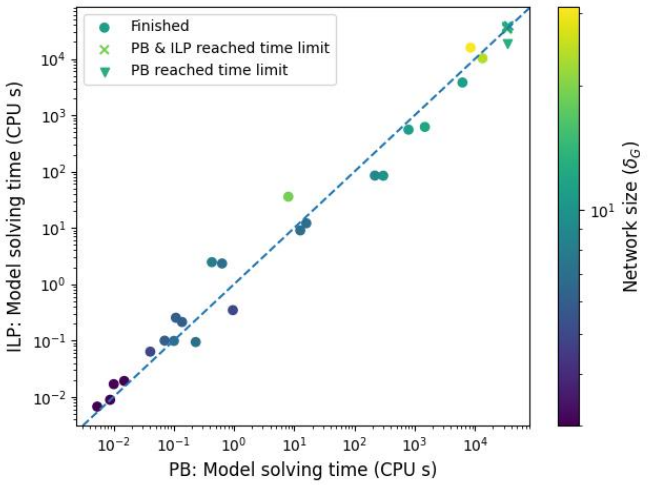

Figure 3: Comparison of the model solving time in CPU seconds between the ILP and PB approach. Each point represents one network, coloured according to its size, as the maximum degree of the network. The shape of the markers indicate if a run finished before the time limit.

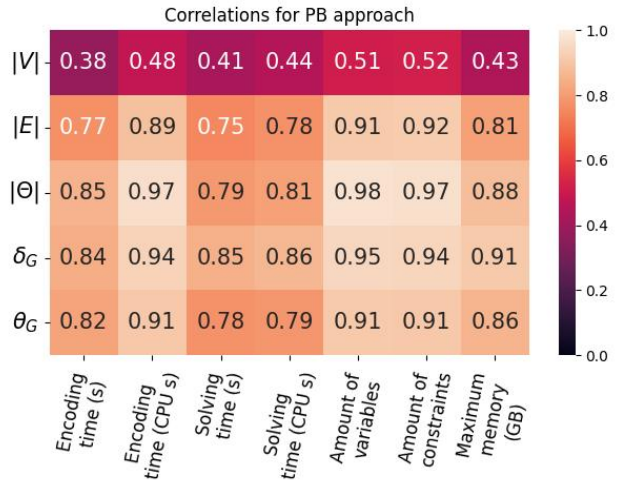

Figure 4: Heatmap of the Pearson correlation coefficients between the dependent (x-axis) and independent variables (y-axis) for the PB implementation.

Table 2: Average measures for the ILP and PB approaches for all networks. For the solving times, we use PAR2 scores.

	ILP	PB
Encoding time (s)	0.25	0.44
Encoding time (CPU s)	0.22	0.41
Solving time (s)	6852.48	7028.76
Solving time (CPU s)	17555.23	17946.54
Amount of variables	9883.03	9924.90
Amount of constraints	1640.33	18778.53
Maximum memory (GB)	1.54	1.68

to those that are most linearly correlated to the respective running time measures of the plots. In Figure 4, we show the correlation coefficients between the running time and all chosen network characteristics for the pseudo-Boolean approach. The coefficients for the ILP approach are highly similar. When we use the null hypotheses between each of the running time measures and each of the independent variables that their correlation is zero, the lowest two-tailed p-value, for encoding time in seconds and $|V|$, is 0.0319, indicating that there is a statistically significant difference and thus evidence for a correlation. The number of vertices is the least correlated with each of the dependent variables.

RQ2

Similarly to the running time, there is evidence for a correlation between the each of the independent variables and the number of variables and constraints of the model. We show the specific correlation coefficients for the PB model in Figure 4. The weakest correlation of the number of variables and constraints is with the number of vertices of the graphs. The strongest correlation is with the number of triangles. This is also the case for the ILP model.

When we express the model size as a function of network size, we can see a reflection of this finding. We discard some of the constraint-reducing optimisations that use δ_v, δ^*, θ_v and θ^* in order to be able to express the number of constraints in the model as a function of the five graph characteristics. In Big O notation, the number of constraints of the ILP is the following:

$$O(|E| \cdot |\Theta|)$$

The number of constraints of the PB model is the following:

$$O(|E| \cdot |\Theta| + \delta_G * \theta_G * k)$$

This means that they are equal for $k = 2$. The Big O notation is exactly the same for the number of constraints.

Turning to the experimental results, partially visualised in Figure 5 we see that the amount of constraints of the PB model is consistently larger than the amount for the ILP model. The colour of the points also reflects the strong correlation of number of constraints with number of triangles. When we turn to the averages in Table 2, we see that the average number of constraints for the PB model is larger than that of the ILP model, by a factor of 11.45. We also see that the average number of variables is almost exactly equal for the PB and ILP models.

RQ3

The comparison of memory consumption between the PB and ILP approach, visualised in Figure 6, shows us that the PB approach performs worse for most of the networks, but uses less memory for some of the larger networks that did not reach the time limit. The two largest networks used the most memory for both of the approaches. We cannot conclude anything about the networks that reached the time limit.

The average maximum memory consumption, displayed in Table 2, is 1430 MB lower for the ILP implementation than the PB implementation. This means that solving the PB model, requires on average 9.31% more memory than solving the ILP model. For the PB approach, the maximum de-

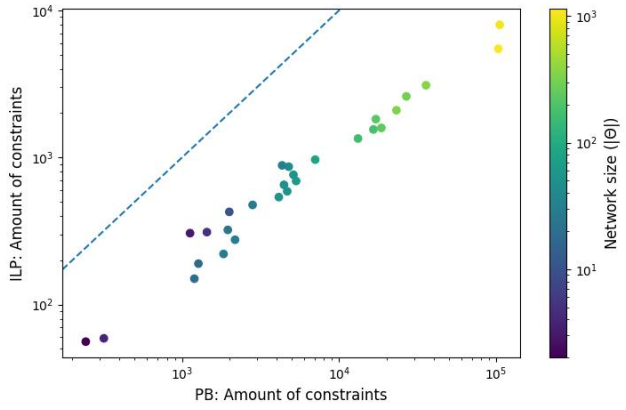

Figure 5: Comparison of the number of constraints of the PB and ILP models. Each point represents one network, coloured according to its size in number of triangles.

gree is the best predictor for memory consumption, followed by the number of triangles, then the maximum triangles and the number of edges, as is shown in Figure 4. The number of vertices is the worst predictor, though there is still evidence for a linear correlation. This also holds for the ILP approach.

RQ4

Figure 7 shows the comparison between the Gurobi and SCIP model solvers, in terms of solving time in CPU seconds. We see that for almost all models, Gurobi performed better. In Table 3, we show the average solving times per combination of solver and model type. In this experiment, SCIP takes, on average, 3677% more wall-clock time and 1179% more CPU time than Gurobi for the ILP models. For the PB models, SCIP also performs worse, taking on average 2735% more wall-clock time and 513% more CPU time.

	Gurobi		SCIP	
	ILP	PB	ILP	PB
Solving time (s)	4.00	4.41	151.19	125.03
Solving time (CPU s)	11.82	20.38	151.15	124.96

Table 3: Average model solving times of the small networks, for every combination of solver and model type.

Similar to the results of the first research question, Gurobi is on average slower for the PB models, taking on average 10% more wall-clock time and 72% more CPU time than the ILP models. SCIP is however faster for PB models, taking on average 17% less time, both in wall-clock and CPU seconds.

We must note that SCIP does not return the optimum solution to some of the PB models, even when it finishes the optimisation and returns the "optimal" solving status. These solutions are not optimal, because Gurobi found a feasible solution for the same networks with less edge deletions. This was the case for 3 out of the 18 networks. We also note that for 11 out of 18 of the PB models, the "optimal" solutions returned by SCIP are not even feasible solutions. Figure 7 shows these points with a red highlight.

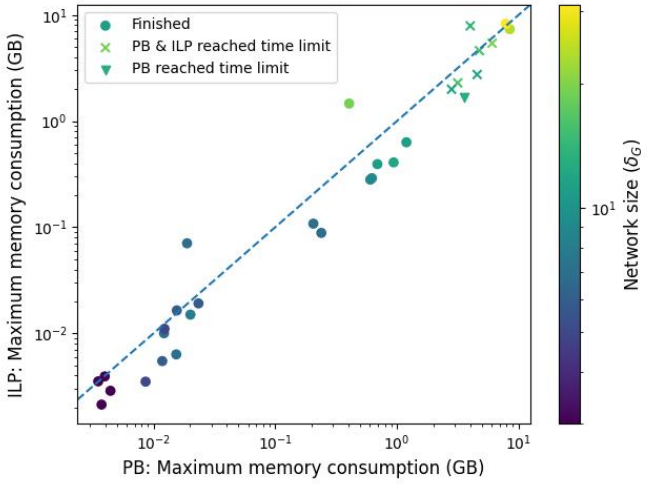

Figure 6: Comparison of the maximum memory consumption during solving of the PB and ILP models. Each point represents one network, coloured according to its size, as the maximum degree of the network. The shape of the markers indicate if a run finished before the time limit.

5 Conclusion

In this work, we introduced a novel approach to the problem of graph anonymisation, using a method in which we find the minimal number of edge deletions that can be performed to make a graph (n, m)-k-anonymous. In this approach, we use two model solvers to solve the problem, encoded as a *pseudo-Boolean* (PB) model.

We compared the running times and memory consumption of our method with an existing *Integer Linear Program* (ILP) method, both encoded and solved in the model solver Gurobi. We found that the PB method performed worse in terms of running time and memory consumption. The running time is however only a small percentage more than the existing ILP method. We expect that the same holds for the memory consumption, though more research needs to be done in order to conclude this.

Futhermore, we conclude that Gurobi is able to solve the ILP and PB models in less time than SCIP. Though SCIP solved the PB models in less time than the ILP models, we found that it did not give a trustworthy result for the PB models. A next step is to generate certificates from the logs of the runs in which PB models were solved and look into potential causes of non-optimal results.

This work shows evidence towards showing that specific instances of the graph anonymisation problem can be modelled using only pseudo-Boolean constraints, without an exponential increase in running time or memory consumption compared to Integer Linear Program models. This opens up the possibility for creating certificates, which we can use to show that some network, that a solver finds is infeasible in a specific setting of graph anonymisation, is definitively infeasible in that setting.

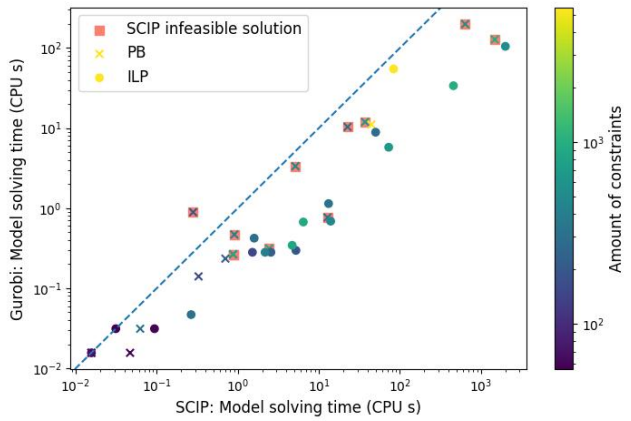

Figure 7: Comparison of the model solving time in CPU seconds between the Gurobi and SCIP model solvers. Each point represents one model, so every network is represented with two points, the cross is the PB model and the dot the ILP model. The colours of the points correspond with the amount of constraints of the model. The points for which the SCIP solver returned an infeasible solution are highlighted in red.

5.1 Discussion

In this section, we present limitations on our research. For the experiments, we only ran the model solver for animal social networks. Though we selected the networks on their variation in the five chosen graph characteristics, the networks might be skewed for some other graph characteristics. This suggests the possibility that some other graph characteristic has a significant influence on running time or memory consumption.

For this research, we were able to run every network only once on DelftBlue. Because the running time of encoding and solving the model is not deterministic, the chance exists that differences in running time are accidental. For the fourth research question, this is also a limitation, in particular because we can only provide the running times of 18 networks. We choose to include this experiment, because it presents a motivation for further research.

5.2 Future Work

A possible next step is to look into creating a PB model for the budgeted version of the graph anonymisation problem. Due to the nature of the budgeted version, we expect faster running times for solving this model. We might be able to use this to inform our choice of an appropriate budget for a network. This could then be used in combination with a faster, heuristic algorithm for budgeted graph anonymisation.

We also suggest future research to look into other settings of the graph anonymisation problem. One such suggestion is to create a pseudo-Boolean model for graph anonymisation with a different anonymity measure. Another suggestion is to look at the setting in which we maximise some other measure of data utility, instead of minimising the number of deleted edges. If we know that social scientists mostly look at certain network statistics, such as graph connectedness, we might want to anonymise a network in such a way that it is fully anonymous, with the smallest possible difference in graph connectedness.

Acknowledgments

We want to thank Dr. Anna L.D. Latour for their feedback and guidance throughout the project and for providing us valuable resources and information. We thank Dr. Frank W. Takes and Rachel G. de Jong for providing us input and feedback. To the peers in the CSE3000 Research Project group, we express gratitude for their feedback, collaboration and willingness to explore topics together. We also want to thank the two anonymous reviewers for their review of the draft. This research would not have been possible without the resources provided by the TU Delft, such as access to publications, software and the DelftBlue Supercomputer.

6 Responsible Research

We aim to provide enough information in the main body of this work to reproduce our method, but attach an appendix with details about the pseudo-Boolean model to ease reproducibility. Of the date of writing, we cannot make the implementation of the model public, as it builds upon work that is yet to be published. We provide a stable link to the repository in a footnote and aim to make this public when this is possible.

To add to the reproducibility of this work, we provide a comprehensive list of the specific networks used as input for the experiments in the appendix. We also describe the software and hardware in detail. In the discussion, we explain the limitations on reproducibility. To increase transparency, we include negative results in this work, when encountered.

During research, the Netherlands Code of Conduct for Research Integrity provided us guidelines on how to conduct our research activity in a responsible manner [KNAW *et al.*, 2018]. We did not make use of generative (large) language models during our research. To reduce our environmental impact, we limited the amount and size of requests to DelftBlue.

As explained in the introduction, allowing social scientists to conduct research on networks is an instrumental motivation of this work. However, we should also consider that any work on graph anonymisation can also be of help to attackers. If we publish information about the minimal number of edge deletions needed to make a privacy-sensitive network (n, m)-k-anonymous, we provide attackers with information about the network that they might be able to use to reveal privacy-sensitive information about the network. To mitigate this risk, we only use datasets that have been published and do not pose a privacy risk.

Our focus on complete graph anonymisation ensures that any network anonymised with this method cannot form a privacy risk, given that the attacker only has knowledge of the number of triangles of the ego networks and the degrees of the vertices. When an attacker has more knowledge, they might be able to identify certain vertices, edges or subgraphs. In the Future work section, we suggest research that would diminish this risk.

References

[Barth, 1995] Peter Barth. A Davis-Putnam based enumeration algorithm for linear pseudo-Boolean optimization. 1995.

[Biere *et al.*, 2021] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of Satisfiability*. Number v. 336 in Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, second edition, 2021.

[Bolusani *et al.*, 2024] Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, Antonia Chmiela, João Dionísio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Christoph Graczyk, Katrin Halbig, Ivo Hedtke, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Dominik Kamp, Thorsten Koch, Kevin Kofler, Jurgen Lentz, Julian Manns, Gioni Mexi, Erik Mühmer, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Mark Turner, Stefan Vigerske, Dieter Weninger, and Lixing Xu. The SCIP optimization suite 9.0. Technical Report, Optimization Online, February 2024.

[Casas-Roma *et al.*, 2013] Jordi Casas-Roma, Jordi Herrera-Joancomartí, and Vicenç Torra. An algorithm for k -degree anonymity on large networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 671–675, Niagara Ontario Canada, August 2013. ACM.

[Cheng *et al.*, 2010] James Cheng, Ada Wai-chee Fu, and Jia Liu. K-isomorphism: Privacy preserving network publication against structural attacks. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 459–470, New York, NY, USA, June 2010. Association for Computing Machinery.

[de Jong *et al.*, 2024] Rachel G. de Jong, Mark P. J. van der Loo, and Frank W. Takes. A systematic comparison of measures for k-anonymity in networks, July 2024.

[de Jong *et al.*, 2025] Rachel G. de Jong, Mark P. J. van der Loo, and Frank W. Takes. The anonymization problem in social networks, April 2025.

[Delft High Performance Computing Centre (DHPC), 2024] Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 2). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2, 2024.

[Gocht and Nordström, 2021] Stephan Gocht and Jakob Nordström. Certifying Parity Reasoning Efficiently Using Pseudo-Boolean Proofs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):3768–3777, May 2021.

[Gocht *et al.*, 2020] Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Subgraph Isomorphism Meets Cutting Planes: Solving With Certified Solutions. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 1134–1140, Yokohama, Japan, July 2020. International Joint Conferences on Artificial Intelligence Organization.

[Gocht, 2025] Stephan Gocht. VeriPB, April 2025. Available at github.com/StephanGocht/VeriPB.

[Gurobi Optimization, LLC, 2024] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[Hagberg *et al.*, 2008] Aric Hagberg, Pieter J. Swart, and Daniel A. Schult. Exploring network structure, dynamics, and function using NetworkX. Technical Report LA-UR-08-05495; LA-UR-08-5495, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), January 2008.

[Kerschke *et al.*, 2018] Pascal Kerschke, Jakob Bossek, and Heike Trautmann. How do Performance Indicator Parametrizations Influence the Assessment of Algorithm Portfolios?, 2018.

[KNAW *et al.*, 2018] KNAW, NFU, NWO, TO2-Federatie, Netherlands Association of Universities of Applied Sciences, and VSNU. Netherlands Code of Conduct for Research Integrity, 2018.

[Latour, 2024] Anna L.D. Latour. Research note - Anonymisation - ILP encoding. Internal communication, unpublished, 2024.

[McNulty, 2022] Keith McNulty. 8. Assortativity and Similarity. In *Handbook of Graphs and Networks in People Analytics: With Examples in R and Python*. Chapman and Hall/CRC, New York, June 2022.

[Mexi *et al.*, 2025] Gioni Mexi, Dominik Kamp, Yuji Shinano, Shanwen Pu, Alexander Hoen, Ksenia Bestuzheva, Christopher Hojny, Matthias Walter, Marc E. Pfetsch, Sebastian Pokutta, and Thorsten Koch. State-of-the-art Methods for Pseudo-Boolean Solving with SCIP, January 2025.

[Pearson, 1895] Karl Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242, December 1895.

[Rossi and Ahmed, 2015] Ryan A. Rossi and Nesreen K. Ahmed. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*, 2015.

[Xie, 2023] Xinyue Xie. *Anonymization Algorithms for Privacy-Sensitive Networks*. PhD thesis, LIACS, Leiden University, 2023.

[Zhou and Pei, 2008] Bin Zhou and Jian Pei. Preserving Privacy in Social Networks Against Neighborhood Attacks. In *2008 IEEE 24th International Conference on Data Engineering*, pages 506–515, April 2008.

Appendix

Full pseudo-Boolean model

In this section, we give the formal definitions of all variables and constraints of the pseudo-Boolean model. First, we introduce all variables. In the next subsection, we give formal definitions of the original constraints. We then introduce the full pseudo-Boolean constraints derived from the original constraints.

Variables

All variables in the pseudo-Boolean model are Boolean. We give the variables as semantically described in Variables 3.2:

$$\begin{aligned}
L &:= \{\ell_{u,v} \mid (u,v) \in E\} \\
T &:= \{t_{u,v,w} \mid (u,v,w) \in \Theta\} \\
S &:= \{s_{v,i,j} \mid v \in V; 0 \le i \le \delta_v; 0 \le j \le \theta_v\} \\
N &:= \{n_{v,i} \mid v \in V; 0 \le i \le \delta_v\} \\
M &:= \{m_{v,j} \mid v \in V; 0 \le j \le \theta_v\} \\
Z &:= \{z_{i,j,p} \mid 0 \le i \le \delta^*; 0 \le j \le \theta^*; 1 \le p \le k\}
\end{aligned}$$

We supplement these indicator variables with the following four helper variables:

$$\begin{aligned}
N_< &:= \{n_{<,v,i} \mid v \in V; 0 \le i \le \delta_v\} \\
N_> &:= \{n_{>,v,i} \mid v \in V; 0 \le i \le \delta_v\} \\
M_< &:= \{m_{<,v,j} \mid v \in V; 0 \le j \le \theta_v\} \\
M_> &:= \{m_{>,v,j} \mid v \in V; 0 \le j \le \theta_v\}
\end{aligned}$$

Original Constraints

We introduce the following constraints to define the variables:

$$C_{T_{\text{def}}} := \{t_{u,v,w} \quad \Leftrightarrow (\ell_{u,v} + \ell_{v,w} + \ell_{u,w} \ge 3) \mid (u,v,w) \in \Theta\}$$

$$C_{N_{\text{def}}} := \left\{n_{v,i} \quad \Leftrightarrow \left(\sum_{u \in N_1(v)} \ell_{u,v} = i\right) \mid v \in V; 0 \le i \le \delta_v\right\}$$

$$C_{N_{<\text{def}}} := \left\{n_{<,v,i} \quad \Rightarrow \left(\sum_{u \in N_1(v)} \ell_{u,v} < i\right) \mid v \in V; 0 \le i \le \delta_v\right\}$$

$$C_{N_{>\text{def}}} := \left\{n_{>,v,i} \quad \Rightarrow \left(\sum_{u \in N_1(v)} \ell_{u,v} > i\right) \mid v \in V; 0 \le i \le \delta_v\right\}$$

$$C_{M_{\text{def}}} := \left\{m_{v,j} \quad \Leftrightarrow \left(\sum_{(u,v,w) \in Tri(v)} t_{u,v,w} = j\right) \mid v \in V; 0 \le j \le \theta_v\right\}$$

$$C_{M_{<\text{def}}} := \left\{m_{<,v,j} \quad \Rightarrow \left(\sum_{(u,v,w) \in Tri(v)} t_{u,v,w} < j\right) \mid v \in V; 0 \le j \le \theta_v\right\}$$

$$C_{M_{>\text{def}}} := \left\{m_{>,v,j} \quad \Rightarrow \left(\sum_{(u,v,w) \in Tri(v)} t_{u,v,w} > j\right) \mid v \in V; 0 \le j \le \theta_v\right\}$$

$$C_{S_{\text{def}}} := \{s_{v,i,j} \quad \Leftrightarrow (n_{v,i} + m_{v,j} \ge 2) \mid v \in V; 0 \le i \le \delta_v; 0 \le j \le \theta_v\}$$

$$C_{Z_{\text{def}}} := \left\{z_{i,j,p} \quad \Leftrightarrow \left(\sum_{v \in V} s_{v,i,j} \ge p\right) \mid 0 \le i \le \delta_v;\ 0 \le j \le \theta_v;\ 1 \le p \le k\right\}$$

We introduce the following constraints on variables N, M, S and Z to constrain the model, as explained in Section 3.2:

$$C_{N_{\text{cons}}} := \left\{\sum_{n_{v,i} \in N} n_{v,i} = 1 \quad \mid v \in V\right\}$$

$$C_{M_{\text{cons}}} := \left\{\sum_{m_{v,j} \in M} m_{v,j} = 1 \quad \mid v \in V\right\}$$

$$C_{S_{\text{cons}}} := \left\{\sum_{s_{v,i,j} \in S} s_{v,i,j} = 1 \quad \mid v \in V\right\}$$

$$C_{Z_{\text{cons}}} := \left\{\sum_{p=1}^{k} z_{i,j,p} = k \cdot z_{i,j,1} \;\middle|\; 0 \le i \le \delta^*; 0 \le j \le \theta^*\right\}$$

Pseudo-Boolean Constraints

We convert all constraints from the previous section to pseudo-Boolean constraints, using the constraint conversion explained in Subsection 3.1. Some constraints are now additionally numbered, because the constraint conversion can result in one to three pseudo-Boolean constraints per original constraint. Since Gurobi cannot handle negated literals, we converted every negated literal $\overline{x}$ to $(1 - x)$ for implementation.

We derive the following constraints from $C_{T_{\text{def}}}$:

$$C_{T_{\text{def1}}} := \{3 \cdot \overline{t_{u,v,w}} + \ell_{u,v} + \ell_{v,w} + \ell_{u,w} \ge 3 \mid (u,v,w) \in \Theta\}$$

$$C_{T_{\text{def2}}} := \{t_{u,v,w} + \overline{\ell_{u,v}} + \overline{\ell_{v,w}} + \overline{\ell_{u,w}} \ge 1 \mid (u,v,w) \in \Theta\}$$

We derive the following constraints from $C_{N_{\text{def}}}$:

$$C_{N_{\text{def1}}} := \left\{i \cdot \overline{n_{v,i}} + \sum_{u \in N_1(v)} \ell_{u,v} \ge i \;\middle|\; v \in V; 0 \le i \le \delta_v\right\}$$

$$C_{N_{\text{def2}}} := \left\{(-i + |N_1(v)|) \cdot \overline{n_{v,i}} + \sum_{u \in N_1(v)} \overline{\ell_{u,v}} \ge -i + |N_1(v)| \;\middle|\; v \in V; 0 \le i \le \delta_v\right\}$$

$$C_{N_{\text{def3}}} := \{n_{v,i} + n_{<,v,i} + n_{>,v,i} \ge 1 \mid v \in V; 0 \le i \le \delta_v\}$$

We derive the following constraints from $C_{M_{\text{def}}}$:

$$C_{M_{\text{def1}}} := \Big\{ j \cdot \overline{m_{v,j}} + \sum_{(u,v,w)\in Tri(v)} t_{u,v,w} \geq j \;\Big|\; v \in V; 0 \leq j \leq \theta_v \Big\}$$

$$C_{M_{\text{def2}}} := \Big\{ (-j + |Tri(v)|) \cdot \overline{m_{v,j}} + \sum_{(u,v,w)\in Tri(v)} \overline{t_{u,v,w}} \geq -j + |Tri(v)| \;\Big|\; v \in V; 0 \leq j \leq \theta_v \Big\}$$

$$C_{M_{\text{def3}}} := \{ m_{v,j} + m_{<,v,j} + m_{>,v,j} \geq 1 \mid v \in V; 0 \leq i \leq \theta_v \}$$

From the constraints defining the helpers of N and M, we derive the following constraints:

$$C_{N_{<_{\text{def}}}} := \Big\{ (-i + 1 + |N_1(v)|) \cdot \overline{n_{<,v,i}} + \sum_{u\in N_1(v)} \overline{\ell_{u,v}} \geq -i + 1 + |N_1(v)| \;\Big|\; v \in V; 0 \leq i \leq \delta_v \Big\}$$

$$C_{N_{>_{\text{def}}}} := \Big\{ (i+1) \cdot \overline{n_{>,v,i}} + \sum_{u\in N_1(v)} \ell_{u,v} \geq i+1 \;\Big|\; v \in V; 0 \leq i \leq \delta_v \Big\}$$

$$C_{M_{<_{\text{def}}}} := \Big\{ (-j + 1 + |Tri(v)|) \cdot \overline{m_{<,v,j}} + \sum_{(u,v,w)\in Tri(v)} \overline{t_{u,v,w}} \geq -j + 1 + |Tri(v)| \;\Big|\; v \in V; 0 \leq j \leq \theta_v \Big\}$$

$$C_{M_{>_{\text{def}}}} := \Big\{ (j+1) \cdot \overline{m_{>,v,j}} + \sum_{(u,v,w)\in Tri(v)} t_{u,v,w} \geq j+1 \;\Big|\; v \in V; 0 \leq j \leq \theta_v \Big\}$$

From $C_{S_{\text{def}}}$ and $C_{Z_{\text{def}}}$, we derive the following constraints:

$$C_{S_{\text{def1}}} := \{ 2 \cdot \overline{s_{v,i,j}} + n_{v,i} + m_{v,j} \geq 2 \mid v \in V; 0 \leq i \leq \delta_v; 0 \leq j \leq \theta_v \}$$

$$C_{S_{\text{def2}}} := \{ s_{v,i,j} + \overline{n_{v,i}} + \overline{m_{v,j}} \geq 1 \mid v \in V; 0 \leq i \leq \delta_v; 0 \leq j \leq \theta_v \}$$

$$C_{Z_{\text{def1}}} := \Big\{ p \cdot \overline{z_{i,j,p}} + \sum_{v\in V} s_{v,i,j} \geq p \;\Big|\; 0 \leq i \leq \delta^*; 0 \leq j \leq \theta^*; 1 \leq p \leq k \Big\}$$

$$C_{Z_{\text{def2}}} := \Big\{ (-p + 1 + |V|) \cdot z_{i,j,p} + \sum_{v\in V} \overline{s_{v,i,j}} \geq -p + 1 + |V| \;\Big|\; 0 \leq i \leq \delta^*; 0 \leq j \leq \theta^*; 1 \leq p \leq k \Big\}$$

From $C_{N_{\text{cons}}}$, $C_{M_{\text{cons}}}$, $C_{S_{\text{cons}}}$ and $C_{Z_{\text{cons}}}$, we derive the following constraints:

$$C_{N_{\text{cons1}}} := \Big\{ \sum_{n_{v,i}\in N} n_{v,i} \geq 1 \quad \mid v \in V \Big\}$$

$$C_{N_{\text{cons2}}} := \Big\{ \sum_{n_{v,i}\in N} n_{v,i} \leq 1 \quad \mid v \in V \Big\}$$

$$C_{M_{\text{cons1}}} := \Big\{ \sum_{m_{v,j}\in M} m_{v,j} \geq 1 \quad \mid v \in V \Big\}$$

$$C_{M_{\text{cons2}}} := \Big\{ \sum_{m_{v,j}\in M} m_{v,j} \leq 1 \quad \mid v \in V \Big\}$$

$$C_{S_{\text{cons1}}} := \Big\{ \sum_{s_{v,i,j}\in S} s_{v,i,j} \geq 1 \quad \mid v \in V \Big\}$$

$$C_{S_{\text{cons2}}} := \Big\{ \sum_{s_{v,i,j}\in S} s_{v,i,j} \leq 1 \quad \mid v \in V \Big\}$$

$$C_{Z_{\text{cons1}}} := \Big\{ \sum_{p=1}^{k} z_{i,j,p} \geq k \cdot z_{i,j,1} \;\Big|\; 0 \leq i \leq \delta^*; 0 \leq j \leq \theta^* \Big\}$$

$$C_{Z_{\text{cons2}}} := \Big\{ \sum_{p=1}^{k} z_{i,j,p} \leq k \cdot z_{i,j,1} \;\Big|\; 0 \leq i \leq \delta^*; 0 \leq j \leq \theta^* \Big\}$$

Input networks

Table 4 shows the names and graph characteristics about all of the input networks that we use for the experiments.

Supplementary figures

In this appendix, we supply some supplementary figures. Figure 8 shows the correlation coefficients for the ILP approach. In Figures 9 and 10, we give a comparison between the two approaches on encoding and solving time in wall-clock seconds. In Figure 11, we show the same comparison but for the number of variables of the models. In the last figure, Figure 12, we visualise the comparison between the two different model solvers in terms of model solving time in wall-clock seconds.

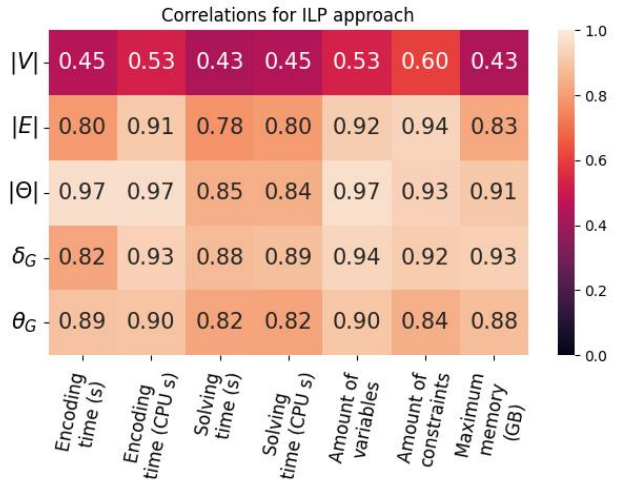

Figure 8: Heatmap of the Pearson correlation coefficients between the dependent (x-axis) and independent variables (y-axis) for the ILP implementation.

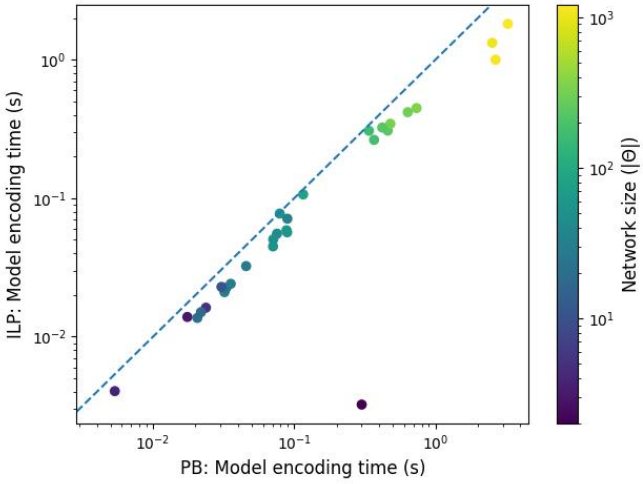

Figure 9: Comparison of the model encoding time in wall-clock seconds between the ILP and PB approach. Each point represents one network, coloured according to its size in number of triangles.

Table 4: All input networks with graph statistics, with the names as listed in the network database [Rossi and Ahmed, 2015]. The ticks and crosses indicate if the network is part of the small networks that we use for the experiments of the fourth research question.

		$\|V\|$	$\|E\|$	$\|\Theta\|$	δ_G	θ_G
aves-barn-swallow-contact-network	✗	17	53	55	11	22
aves-geese-female-foraging	✓	20	190	1140	19	171
aves-sparrowlyon-flock-season2	✗	46	348	1210	31	215
fish-guppy-familiar-1	✓	6	15	20	5	10
insecta-ant-trophallaxis-colony2-day4	✓	31	41	5	7	3
insecta-beetle-group-c1-period-1	✗	30	185	359	19	82
insecta-beetle-group-c4-period-1	✗	30	138	180	16	42
mammalia-baboon-association-group01	✓	4	6	4	3	3
mammalia-baboon-grooming-group05	✓	4	5	2	3	2
mammalia-baboon-grooming-group17	✓	7	7	0	3	0
mammalia-bison-dominance	✗	26	222	1018	24	188
mammalia-macaque-contact-sits	✗	28	378	3276	27	351
mammalia-primate-association-12	✗	16	89	241	14	64
mammalia-primate-association-17	✓	7	20	30	6	14
mammalia-primate-association-8	✗	19	114	304	16	81
mammalia-raccoon-proximity-24	✓	14	41	54	10	23
mammalia-raccoon-proximity-27	✓	14	27	24	8	15
mammalia-raccoon-proximity-30	✓	8	19	18	7	12
mammalia-raccoon-proximity-44	✓	12	31	29	7	13
mammalia-raccoon-proximity-5	✗	22	67	63	11	23
mammalia-voles-bhp-trapping-07	✓	47	52	11	5	4
mammalia-voles-bhp-trapping-17	✓	97	113	42	6	8
mammalia-voles-bhp-trapping-21	✓	86	138	81	9	17
mammalia-voles-bhp-trapping-24	✗	171	363	331	12	27
mammalia-voles-bhp-trapping-29	✓	103	105	32	6	5
mammalia-voles-kcs-trapping-26	✗	137	271	223	13	33
mammalia-voles-plj-trapping-18	✓	49	66	28	7	8
mammalia-voles-rob-trapping-45	✓	59	89	51	7	11
mammalia-voles-rob-trapping-53	✓	36	26	3	3	1
reptilia-tortoise-network-bsv-1998	✗	88	184	164	12	31
reptilia-tortoise-network-pv	✗	35	66	56	12	23

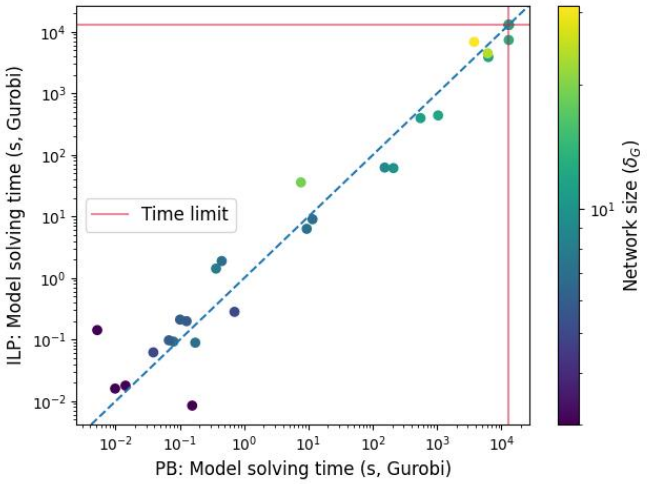

Figure 10: Comparison of the model solving time in wall-clock seconds between the ILP and PB approach. Each point represents one network, coloured according to its size, as the maximum degree of the network. Two lines show the cut-off time for solving.

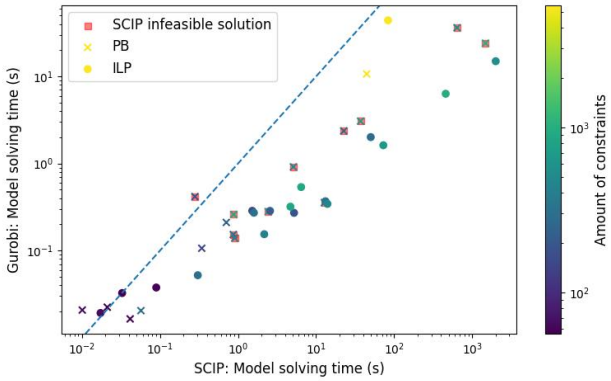

Figure 12: Comparison of the model solving time in wall-clock seconds between the Gurobi and SCIP model solvers. Each point represents one model, so every network is represented with two points, the cross is the PB model and the dot the ILP model. The colours of the points correspond with the amount of constraints of the model.

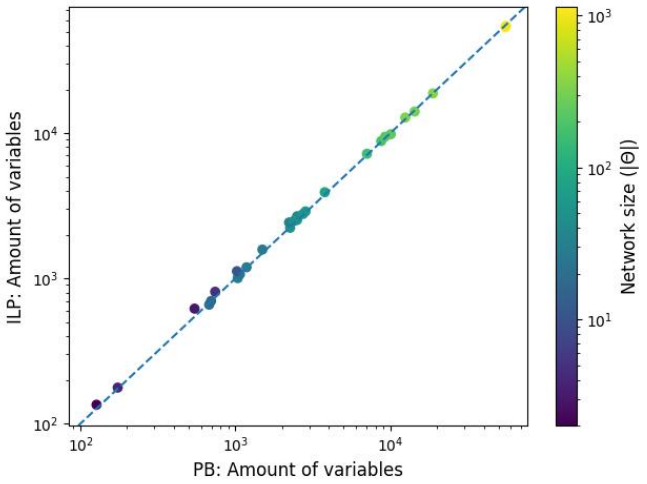

Figure 11: Comparison of the number of variables of the PB and ILP models. Each point represents one network, coloured according to its size in number of triangles.