Sandbox for Social Network Anonymization using Constraint Programming

by

Sreevidya Iyer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday, September 11, 2026 at 13.30 PM.

Student number:	6216501
Project duration:	November 2025 – September 2026
Thesis committee:	Dr. E. Demirović, TU Delft
	Dr. A. L. D. Latour, TU Delft
	Dr. C. De Bacco, TU Delft

Style: Derived from TU Delft Report Style, with modifications by Daan Zwaneveld

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

1

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr. Anna Latour and Dr. Emir Demirović, for their feedback, support, and guidance throughout my thesis. I would like to thank them and Dr. Caterina De Bacco for being part of my thesis committee and evaluating my final examination. I would also like to thank Imko Marijnissen for his help in setting up the Pumpkin solver and in formulating my approach. The journey of the thesis is individual but does not have to be solitary. I thank my peers from the Algorithmics Research group who have been a constant source of feedback, support, and good conversations that kept me going each week.

I would like to thank my family and friends who poured their love, support, and heart to brighten my good days and anchor me on my hard days. Special thanks to my brother, whose belief and unconditional support have been the wind beneath my wings during the master's program.

Sreevidya Iyer
Delft, September 2026

Abstract

Social science research produces a wealth of valuable data in the form of social networks. Making these empirical data public benefits research on socially relevant causes such as the spread of diseases, social interactions in a geographical location or online, movement and travel of people, flow of information in communication networks, etc. To protect the privacy of people represented in these networks while publicizing the data, anonymization is required. One way to identify an individual in network data is to study the local structure of the network around them. By altering the networks such that no node has a uniquely identifiable neighborhood, we can provide additional anonymity beyond concealing names and private identifiers.

The difficulty in this process of social network anonymization is that the network must remain useful to scientists, retaining properties important for downstream analysis. So far, research in this field has focused on maximizing anonymity and only measuring the utility of the anonymized data after the fact. The literature is also fragmented, with bespoke methods spanning different objectives unique to the dataset under consideration. This makes it costly for the data owner to understand and compare different approaches when their expertise lies in the domain of the data and not complex anonymization algorithms.

The main contribution of this thesis is a constraint programming-based framework in which scientists can easily explore the trade-offs between levels of anonymity for different attack models and the utility of the data for different measures. Within this sandbox, anonymity and utility requirements are expressed as independent, composable constraints. We model two structural anonymity measures of increasing strictness, k-degree and k-(degree, triangle), and introduce a betweenness-centrality preservation constraint that protects utility proactively by making modifications between central nodes expensive. We choose constraint programming for its declarative nature, which makes it easier to interpret the anonymization process. Evaluated on real-world networks, the framework shows that actively incorporating utility into the anonymization model is a promising direction.

Contents

2

Introduction

2.1 Social Network Analysis

Consider an epidemiologist who models the spread of an infectious disease through a network of contacts between potentially exposed individuals. Before the data is analyzed, all explicit identifiers such as names, locations, and demographic attributes are removed, and the network is published after the study is complete so that the findings can be verified. An adversary who knows the identity of a few individuals in the network can locate them by their pattern of connections and also re-identify others. The network structure that makes the data useful for third-party analyses also puts the participants at risk of identity exposure.

The analysis of social networks is valuable across a wide range of domains. In epidemiology, contact networks are used to model how an infection propagates through a population and to evaluate interventions [2]. In finance, the network of transactions between accounts reveals patterns of fraud and money laundering that are invisible when transactions are examined individually [31, 22]. These are domains in which the underlying data is sensitive, and the individuals represented in the network have a right to privacy. This creates an additional overhead for open research practice, because the network underlying a published analysis must itself be available if that analysis is to be reproduced or extended. The Copenhagen Networks Study [27] illustrates the trade-off required. The study recorded the physical proximity, calls, text messages, and online friendships of university students and compiled a multi-layer network. Yet, most of the data was released after an embargo of a few years owing to privacy guidelines. Facebook activity data collected in this process was simply not released due to the risk of re-identification of individuals. Datasets of this kind exist only because of volunteers who deserve to assess the risks associated with the release of their data. Anonymization is what makes it possible to answer that question and is therefore a necessary condition for ethical and effective research.

Each aspect of the underlying network structure of both physical networks (restricted by geography) and virtual networks (online) is analyzed and assessed by various interested stakeholders. To encourage academic use of research data and protect against malicious use, it is not sufficient to remove explicit identifiers; the network itself must be anonymized.

2.2 Social Network Anonymization Problem

Social network anonymization has thus developed as an area of research distinct from social network analysis and graph theory. With the rise of social networks in the late 1990s and early 2000s, online social networks became widespread but the privacy of users was not yet treated as a primary concern. Subsequent works showed that network data, which retains its original structure, can be actively or passively exploited to leak information about the users of the network [3, 20, 15]. Multiple classes of anonymization approaches have been developed, such as k-anonymity [26, 29], differential privacy [11], random perturbations [33], metaheuristic approaches [1, 4], etc. These approaches are specific to the type of data, downstream applications of the data, and the objectives of the data owners. In our thesis, we will focus on the k-anonymity class of algorithms, which ensure that at least k nodes in the network share some structural property such as degree and neighborhood information.

Anonymization is not without cost. Modifying the graph topology so that the nodes become indistinguishable will interfere with the applications of the data. The properties or downstream tasks that make the network valuable are called utility measures. The trade-off between higher privacy guarantees and preserving the utility of the network for downstream analysis forms a core problem in network anonymization. Various approaches focus on minimizing modifications to a network and measure utility as a by-product of anonymization. Any attempt to retain higher utility in the network primarily focuses on reducing the count of modifications and not on actively preserving a utility measure. We further observe that the literature on anonymization is fragmented. Each method fixes a single anonymity measure, a single utility measure, and a search strategy. Implementing them would also require the data owner, who may not be well-versed in these algorithms, to deal with different pieces of research software, the input format, and parameters. Thus, the user who wishes to find an approach to anonymize their network would need to locate, install, and learn several such tools before they can compare them. This cost is considerable for a social scientist or a researcher. This effort needs to be paid before anonymization is performed, which would further pull up the ladder for scientists who wish to publish impactful research.

To address these limitations, this thesis investigates the consequences of expressing utility preservation as an explicit constraint on a network property, similar to anonymity constraints. We develop a framework for anonymization based on constraint programming, a declarative paradigm using the Pumpkin solver, which supports different solving modes, constraints, and propagations. Constraint programming [24] is particularly suitable for our approach, as we can express these requirements as modular constraints, which can be easily combined by the user. This enables a data owner, familiar with the data but not with the internal workings of a solver, to describe the guarantees required and rely on the framework to find a suitable approach for their problem. The user selects the anonymity measure and utility measure to preserve and chooses how to combine them. We intend for the framework to work as a sandbox rather than an optimization of a single algorithm. Anonymity and utility constraints can be added, changed, or combined to understand the privacy-utility trade-off offered. Jong *et al.*[8] show that the betweenness centrality property of an anonymized graph faces a sharp discrepancy in the process of anonymization. Thus, we have developed a proxy constraint to control the effect of anonymization on the betweenness centrality of a graph.

2.3 Contributions

- **User-defined constraint model.** We model two anonymity measures of increasing strictness, k-deg and k-(deg, Tri), as constraint optimization problems. We further introduce a utility constraint that preserves betweenness centrality [13] by making edge modifications between central nodes expensive. Expressing anonymity and utility as constraints allows a data owner more freedom to explore multiple configurations and pick the most appropriate one for their use. This LEGO-like architecture, which gives the user control over the building blocks of anonymization, forms the basis of our sandbox environment.
- **Anonymization objective.** We implement two objectives within a single framework, one minimizing edge modifications for a given anonymity target and one maximizing anonymity under a utility budget, and compare the advantages of each. In this way, we provide an alternative to the default mode of minimizing modifications by proposing a new way to constrain modifications when the objective is to obtain an any-time or partially anonymous solution.
- **Empirical analysis.** We measure the effectiveness of the utility constraint in preserving betweenness centrality and observe factors that influence the constraint. The main finding points us in a promising direction: actively incorporating constraints on anonymization, targeted towards utility preservation, can be an effective avenue for utility-centered anonymization. We also provide our insights after testing with $\approx$ 30 sample networks. This establishes a reference point for the problems this framework can currently solve.

The remainder of this thesis is organized as follows. Chapter 3 reviews the related work on classes of network anonymization approaches and utility measures and aims to position our contribution in the existing landscape. Chapter 4 introduces the necessary background on graph terminology, graph properties, the anonymity measures, and the constraint programming paradigm. Chapter 5 formalizes the anonymization problem, attack models, the privacy-utility trade-off, and betweenness centrality as a utility. Chapter 6 presents our approach in detail, including the structural, anonymity, and utility constraints of the model, the search-space reductions,

and the overall pipeline. Chapter 7 describes the experimental setup comprising the datasets, infrastructure, configurations, and parameters. Chapter 8 reports the results organized around four research questions and the limitations of the current framework. We conclude our thesis in Chapter 9 by reviewing our objectives and contributions and suggesting possible directions for future work.

3

Related Work

This chapter systematically reviews the principal anonymization algorithms proposed in the literature and identifies the common framework underlying them, comprising node signatures, anonymization operations, objective functions, and utility preservation strategies, several components of which are adopted in this thesis. The review proceeds thematically, grouping methods by the anonymity measure and solving strategy they share so that related approaches may be compared directly. Proceeding in this manner, the chapter isolates the gaps in the existing work that motivate the contribution of this thesis.

3.1 Social Network Anonymization.

Naive anonymization, is inadequate to protect the privacy of users in real-world networks. Backstrom *et al.* [3] and Narayanan and Shmatikov [20] demonstrated the likelihood of active and passive attacks in exploiting social network interactions to reidentify network data. Hay *et al.* [15] demonstrated that simply removing identifying attributes is not enough. They implement a random perturbation algorithm to enhance anonymity beyond naive anonymization. They show that statistical equivalence between the perturbed and original graphs requires model-driven edge perturbations.

Liu and Terzi [19] first proposed the k-degree anonymity measure and minimizing modifications as the objective that we adopt. Early work in network anonymization adapted techniques originally devised for tabular data anonymization. k- degree anonymity transforms a graph G into an anonymized graph G' where every node u has the same degree as at least $k-1$ other nodes through a series of edge addition and deletion operations. We look at some more classes of k-anonymity and their different implementations. Then we review other implementations of structural anonymity to identify gaps in the literature and design our own approach.

3.1.1. K-Anonymity and variations

K-anonymity was initially developed for anonymization of sensitive data in relational stores [29] [26]. The exact structural properties considered when anonymizing the network differ across approaches and affect how strict a privacy guarantee can be achieved. Jong *et al.* [8] suggest that for k-anonymity, we need to choose an anonymity measure to determine when two nodes are considered equivalent and an anonymization algorithm to implement the measure.

K-Degree Anonymity. The ADegree algorithm [28], formulates k-degree graph anonymization as a constraint satisfaction problem. This approach enforces degree anonymity while minimizing edge changes subject to constraints. Compared to the approach by Liu Terzi [19], ADegree can incorporate richer constraints (e.g., limiting perturbation in local neighbourhoods).

However, the CSP formulation scales poorly: constraint solving becomes expensive on large networks. Many online social networks exhibit heavy-tailed degree distributions, making degree anonymity challenging: anonymizing “hubs" requires substantial structural modification. Most approaches, therefore, prefer to operate within a budget and achieve as much anonymity as possible.

This approach is interesting because it connects anonymization with declarative modelling. It incorporates domain-specific structural information to preserve graph properties aligned with the queries of interest.

K-Neighborhood. Zhou and Pei [34] propose the use of neighborhood component code to label and index neighborhoods and use this as an equivalence measure in k-anonymization. In their approach, they test the performance of the anonymized network on answering aggregate queries. However, we consider use-cases for which the network is published as-is to the user without a query layer on it. Romanini *et al.* [23] focus on neighbourhood uniqueness and model how vulnerable networks are to re-identification based on the degree, density and network structure. Their method uses edge-sampling to reduce uniqueness of neighborhoods but does not focus on preserving utility.

K-Isomorphism/Automorphism. These approaches, while guaranteeing stricter anonymity are too complex to be efficiently modelled and scaled in a CSP format. Cheng *et al.* [6] propose k-isomorphism, which requires partitioning the vertex set into groups such that each group is part of isomorphic subgraphs. This is a strong anonymity concept: an adversary with full 1-hop neighbourhood knowledge cannot re-identify any vertex within an isomorphic class of size k. In practice, constructing k-isomorphic groups requires substantial modification of the graph, often including insertion of nodes and distortion to community structures. The k-automorphism model of Zou *et al.* [35] generalises k-isomorphism by exploiting structural symmetries: an anonymized graph is k-automorphic if its automorphism group maps each vertex to at least k-1 others in the same orbit. As observed by algorithmic surveys [7]), for a network to follow this anonymity measure, all its components should be symmetric in at least k-1 points. This is not realistic for most real-world network datasets.

K-(Degree, Triangle). This hybrid anonymity measure which combines anonymity based on degree and triangle count of a node has been recently adopted in multiple works [4, 1]. The empirical findings of Jong *et al.* [8] suggest that this local measure frequently yields results comparable to measures that examine the exact structure of a node's neighborhood. It also requires substantially less structural information and computation.

Bonello *et al.* [4] minimize the number of unique nodes with a genetic algorithm. They encode a solution as a bit string E in which a set bit marks a deletion, subject to a budget Γ enforced as a penalty. Arsene *et al.* [1] optimize the same criterion by simulated annealing under a deletion budget β. Both restrict modification to deletion, since deletion shrinks neighborhoods and thereby improves anonymity most effectively. Allowing addition would enlarge the encoding from the $|E|$ existing edges to all $\binom{|V|}{2}$ possible ones. We adopt the same measure and restriction as it bounds the number of decision variables to $|E|$, and, since deletion only ever destroys triangles, it enables a prior bound on the change in a node's triangle count that we exploit for pruning in Section 6.3.1. We further implement this measure in combination with betweenness centrality preservation while considering another objective of maximizing anonymity.

3.1.2. Other anonymity measures and algorithms

We study some other anonymity measures and anonymization algorithms insofar as is necessary to distinguish our choices and understand the differences in treatment of the problems in the anonymization space.

Each of the following paragraphs introduces the relevance first before expanding on the approach.

Differential privacy Differential privacy does not support the publication of the anonymized graph so that analyses conducted upon it may be reproduced. This is a reason for our focus on k-anonymity, which ultimately permits publication of the graph itself. Differential privacy [11] provides probabilistic guarantees against arbitrary adversaries, and network-specific approaches divide broadly into interactive query-based mechanisms and synthetic data generation [17, 32]. Under the former, a third party never receives the dataset, and under the latter, the released graph is synthesized from a globally perturbed distribution.

Randomized perturbation. These measures are less complex than k-anonymity but are always bound to distort the utility metrics unpredictably, so we do not consider these simpler anonymity measures. Among the earliest approaches [15], these methods randomly insert, delete or rewire edges and have since diversified into spectrum-preserving and weight-based variants [32]. Random insertion and deletion preserve global edge density while scrambling local motifs, collapsing triangle structure and transitivity even under small perturbations. Random switching preserves the degree distribution but ends up erasing community structure. Random-walk anonymization better preserves path-length distributions at the expense of clustering [33]. All three lack formal anonymity guarantees and remain vulnerable to structure-based de-anonymization attacks [33].

Clustering approaches. Zhang *et al.* [33] observe that these approaches poorly preserve several utility metrics such as transitivity, shortest path, betweenness centrality, etc. In clustering-based algorithms, nodes and/or edges are grouped into clusters, and all the nodes in the same cluster are anonymized [33]. The number of nodes/edges in a cluster is revealed to the user, but the exact relationship between nodes in the same cluster is hidden. This approach is simple and efficient; however, cluster boundaries can induce artificial structural blocks not present in the original graph. k-NeuroSVM [18] enhances clustering using learned feature embeddings, enabling clusters that reflect structural similarity beyond degree.

Dynamic graphs. We consider only static, undirected and unweighted graphs and treat the temporal dimension as beyond the scope of this work. Where networks evolve over time, anonymizing each snapshot independently can leak identity through temporal linkage, so consistency and anonymity must be enforced across snapshots. DSNGA-CSP [14] uses community evolution to anchor anonymization decisions, thereby reducing the perturbation required per snapshot, and anonymizes community subgraphs by k-composition and inter-community edges by edge isomorphism.

3.2 Anonymity Utility Trade-off

The works reviewed above differ not only in the anonymity guarantees but also, and more consequentially for this thesis, in how they deal with utility.

Utility is generally treated as something preserved by minimization of structural changes while simultaneously achieving anonymity. Liu and Terzi [19] minimize the number of edge modifications, and ADegree [28] bounds the structural deviation between the original and anonymized graphs; in both cases, it is assumed that a minimally altered graph remains useful.

Beyond this, when we consider partial anonymization, a budget is assigned to the allowed modifications. Bonello *et al.* [4] and Arsene *et al.* [1] cap the number of permitted deletions, which likewise bounds how much edges may change.

To understand how well utility is preserved, it is measured before and after anonymization. Bonello *et al.* [4] evaluate their anonymized graphs on the number of edges deleted, the average clustering coefficient, the average shortest path length, and three downstream tasks, namely network robustness, the overlap of the one hundred most central nodes by betweenness centrality, and the similarity of detected communities; Zhang *et al.* [33] conduct a comparable analysis across anonymization families, and SecGraph [16] generalizes this into a uniform evaluation system.

Zhou and Pei [34] optimize the accuracy of aggregate queries on the released graph, though only under a query-based release model, and DSNGA-CSP [14] preserves community structure by construction, though within a dynamic and differentially private setting.

Random modifications also perturb utility. Zhang *et al.* [33] show that random perturbation collapses triangle structure and transitivity; random switching erases community structure. Clustering-based methods preserve transitivity, shortest paths, and betweenness centrality poorly.

Jong *et al.* [8] compare various k-anonymity measures in their ANO-NET framework and also measure the trade-off between anonymity and utility metrics such as largest connected component size, community, and betweenness centrality. Betweenness centrality shows the sharpest decline in the anonymized graph as the anonymization process progresses, indicating that it is a measure that is usually most affected by anonymization. This, combined with the need for a way to dynamically control anonymization in tandem with utility, also motivates us to approach betweenness centrality as a utility constraint.

3.3 Research Contributions

A substantial line of work formulates anonymization as the maximization of the number of k-anonymous nodes subject to a bound on the number of edge modifications. Jong *et al.* [9] consolidate some of these approaches in ANO-NET, a unified framework that distinguishes full, partial, and budgeted variants of the problem and evaluates several heuristic algorithms on real-world datasets. Building on this formulation, Bonello *et al.* [4] solve it with genetic algorithms, encoding a candidate solution as a bitstring over the edge set and maximizing anonymity under a fixed deletion budget, while Arsene *et al.* [1] apply simulated annealing to the same objective. In each case, utility is only passively addressed: the budget bounds *how many* edges may change, and the utility retained is measured after the fact rather than requested in advance. We address two limitations that

appear here. First, metaheuristic approaches, while being faster, do not certify optimality. Second, and more fundamentally, equating utility preservation with minimal modification, while tried and tested in the literature, is blind to preservation of specific properties of the network for a utility in question. Bonello *et al.* [4] also identify the incorporation of downstream task performance into the objective as an open problem.

Two further works also influence and make a case for our constraint programming-based multi-configuration sandbox environment. The ADegree method [28] formulates k-degree anonymization as a constraint satisfaction problem, enforcing anonymity while bounding the structural deviation from the original graph. We adopt this constraint programming paradigm as a natural means of expressing independent requirements on the solver's behaviour while acknowledging the poor ability of this approach to scale to larger problems. SecGraph [16] provides a uniform, open-source system implementing eleven anonymization algorithms, nineteen utility metrics, and fifteen structure-based de-anonymization attacks, enabling a data owner to compare schemes and thereby make a more informed choice regarding the strictness of the anonymity measure. This work, being on a much larger scale, sets a precedent for anonymization frameworks but is still missing the ability to control the utility of the networks in the process of anonymization.

For the anonymity measures, we adopt the degree-based [19] and degree-and-triangle signature (k-(deg, Tri)) employed by Bonello *et al.* and Arsene *et al.*, restricted to edge-deletion operations, which prior work [9] identifies as the most effective operation for anonymization. We model this measure within our constraint programming framework, since restricting modification to deletion has proven effective in reducing the solver's search space.

The approach to maximize utility by minimizing modifications has been widely adopted and considered the natural objective for anonymization in the above studies. However, it does not offer targeted, proactive control over which structural properties survive anonymization, and for networks in which full anonymization requires extensive modifications, the minimal-edit objective either exceeds the available time budget or does little to bound utility meaningfully. This thesis addresses both concerns through two objective formulations. The first minimizes edge modifications while attaining a target level of privacy and also combines this with a budgeted utility constraint. The second maximizes anonymity subject to a targeted utility constraint that limits the modification of connections between the most central nodes of the network, thereby preserving utility proactively even where the network cannot be fully anonymized. Chapter 5 formalizes these problems, and Chapter 6 presents the constraint model that realizes them.

4

Background

This chapter introduces the concepts and notations required for the remainder of the thesis, agnostic to the specific anonymization framework we propose. We first establish graph terminology and neighborhood information of a node and then provide some background on the betweenness centrality measure in network analysis. Finally, we describe the constraint programming paradigm that the framework is built on.

4.1 Graph Terminology

Graphs. We consider a social network graph to be $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, where the set of vertices V represents entities such as the people in the network, and the set of edges E represents relationships or interactions between them. We focus on simple, undirected, and unweighted graphs, where an edge $\{u, v\} \in \mathsf{E}$ indicates a bidirectional relationship between nodes u and v.

Degree. The degree of a node $u \in \mathsf{V}$ is defined as $\mathsf{deg}_{\mathsf{G}}(u) = |\{v \in \mathsf{V} : \{u, v\} \in \mathsf{E}\}|$, that is, the cardinality of the set of nodes connected to u [7]. The degree sequence of a graph G is a vector of values consisting of the degrees of each of the nodes in the graph. If D is the degree sequence of the graph, each of the degrees $d_i \in D$ should be an integer in the range of [0, n-1], the bounds of which indicate the maximum and minimum possible degrees. [5].

Triangle Count. The triangle count of a node $u \in \mathsf{V}$ is

$$\mathsf{Tri}_{\mathsf{G}}(u) = |\{\, v, w \in \mathsf{V} \setminus \{u\} : \{u, v\} \in \mathsf{E},\ \{u, w\} \in \mathsf{E},\ \{v, w\} \in \mathsf{E}\,\}|$$

i.e. , the number of triangles in G that include node u.

Betweenness Centrality. Betweenness centrality is one of the most widely used measures for identifying structurally important nodes in a network. Freeman [13] originally introduced the measure to quantify the extent to which a node lies "between" other nodes in the communication structure of a network. In the context of social networks, nodes with high betweenness centrality are often interpreted as brokers or bridges connecting other regions of the graph. Such nodes may influence information flow, coordination, or communication between communities, making the preservation of this property important for downstream social network analysis tasks.

For a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, the betweenness centrality value of a node $u \in \mathsf{V}$ is defined as

$$BC_{\mathsf{G}}(u) = \sum_{\substack{s,t \in \mathsf{V} \\ s \neq t \neq u}} \frac{\sigma_{st}(u)}{\sigma_{st}},$$

where σ_{st} denotes the total number of shortest paths between nodes s and t, and $\sigma_{st}(u)$ denotes the number of those shortest paths that pass through u [13]. Nodes that lie on a large fraction of shortest paths therefore have higher centrality values.

4.2 Constraint Programming Paradigm

Constraint Programming (CP) is a declarative optimization and decision-making paradigm in which problems are modelled as variables together with constraints restricting the values those variables may take [24]. Rather than explicitly describing how to construct a solution, CP specifies the properties that a valid solution must satisfy. A Constraint Satisfaction Problem (CSP) is typically defined as a tuple (X, D, C), where:

- $X = \{x_1, x_2, \ldots, x_n\}$ is a set of variables,
- D is the set of finite domains associated with the variables,
- C is a set of constraints restricting allowable assignments.

Definition 4.1 (Constraint Satisfaction Problem)**.** *A Constraint Satisfaction Problem (CSP) is a tuple* $(\mathsf{X}, \mathsf{D}, \mathsf{C})$*, where* $\mathsf{X} = \{x_1, \ldots, x_n\}$ *is a set of variables,* D *is the set of finite domains associated with the variables, and* C *is a set of constraints restricting the allowable assignments. A* solution *is an assignment of a value from its domain to every variable such that all constraints in* C *are satisfied.*

CP solvers iteratively reduce variable domains through constraint propagation and search procedures until a consistent assignment is found or infeasibility is proven. Propagation helps prune the domain of a variable by making sure that whenever a value is assigned to a variable, the domains of the other variables connected by a constraint to the assigned variable are re-evaluated and the infeasible assignments are pruned from all the variables' domains.

Definition 4.2 (Constraint Optimization Problem)**.** *A Constraint Optimization Problem (COP) is a tuple* $(\mathsf{X}, \mathsf{D}, \mathsf{C}, o)$*, where* $(\mathsf{X}, \mathsf{D}, \mathsf{C})$ *is a CSP and* o *is an objective expressed over* X*. A solution is* optimal *if it satisfies all constraints in* C *and no other feasible assignment yields a better value of* o*.*

Constraint programming is particularly suitable for combinatorial optimization problems because complex requirements can be expressed independently as modular constraints [24]. In the context of graph anonymization, this allows anonymity requirements, utility requirements, and the optimization objective to be combined flexibly within a single model so that new attacker models or utility measures can be incorporated without redesigning it. Solonets *et al.* [28] model graph anonymization as a CSP whose objective is to construct an anonymized graph satisfying k-deg anonymity while preserving utility, formalizing anonymization through an ADegree constraint that enforces anonymity and bounds the structural deviation between the original and anonymized graphs.

Constraint programming is particularly suitable for combinatorial optimization problems because complex requirements can be expressed independently as modular constraints. [24] In the context of our graph anonymization problem, one of the important reasons why the constraint programming approach is chosen is that it provides a natural framework for expressing anonymity and utility requirements simultaneously. Solonets *et al.* [28] model graph anonymization as a constraint satisfaction problem where the objective is to construct an anonymized graph $\mathsf{G}' = (\mathsf{V}', \mathsf{E}')$ that satisfies k-degree anonymity while preserving the utility of the original graph as much as possible. Their approach formalizes anonymization through the ADegree constraint, which enforces anonymity and bounds the structural deviation between the original and anonymized graphs.

The CP paradigm is particularly useful in this context because anonymity constraints, utility preservation constraints, and optimization objectives can be combined flexibly within a unified framework. New attacker models or utility metrics can therefore be incorporated smoothly without redesigning the anonymization model.

5

Problem Description

Social networks model interactions between individuals in the form of graph structures. A social network is typically represented as a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, where the vertices V are the users and edges E represent interactions such as friendships, follows, or communication events between the users. We consider undirected and unweighted graphs in our problem, where nodes connected by an edge exert equal influence over each other. Anonymization aims to transform G into a sanitized graph $\mathsf{G}' = (\mathsf{V}', \mathsf{E}')$ with minimal structural identifiers. The privacy risk in social network publication arises because attackers can exploit the structural or attribute-based uniqueness of users. Several attacker models are commonly assumed in the anonymization literature, each corresponding to a different form of background knowledge and motivating a different privacy requirement.

The remainder of this chapter is organized as follows. We first define the anonymity measures formally. Section 5.2 describes the structural attacker models we target and the two anonymity measures we adopt. We then provide an example to visualize the problem. Section 5.4 introduces the levels of anonymity and their relevance. Section 5.5 discusses the privacy-utility trade-off and how we use betweenness centrality to study the same. Section 5.6 states the optimization problem formally as two solving modes.

5.1 Node signatures and anonymity measures

Liu and Terzi [19] defined k-degree anonymity using the frequency of distinct degree values appearing in the degree sequence.

Definition 1: A set of integers is S is k-anonymous when every distinct value v in $v_i \in S$ appears atleast k times in the set.

Definition 2: A graph G is k-degree anonymous if the set of degree sequences D of the graph is k-anonymous.

A stronger anonymity measure is obtained by extending the node signature to include triangle count.

Definition 3: A graph G is k-(deg, tri) anonymous if the set of (degree, triangle) signature tuples is k-anonymous. $(degree, triangle)$-*signature* of a node u is the pair

$$\sigma(u) = \big(\mathsf{deg}_{\mathsf{G}}(u),\ \mathsf{Tri}_{\mathsf{G}}(u)\big).$$

5.2 Attack models.

We focus on attack models that target the structural properties of users. This is because the graphs we are anonymizing are undirected social network graphs, assuming influence between connected nodes is bi-directional. While key attributes, quasi-attributes, and sensitive attributes [8] may be present in such networks, the sandbox environment is aimed at owners or publishers of the data to experiment using a combination of different constraints and parameters and identify the right approach for their problem. The approaches that focus on

edge- or node-related attributes are more suitable for datasets containing sensitive information such as patient medical records, social security, employment information, etc., and are not applicable in the context of social networks.

We have chosen some approaches from the k-anonymity class as described by Jong *et al.* [8] to model the initial framework and observe the effectiveness of creating such a sandbox with different anonymity and utility constraints.

Degree-based node signature. In this attack, we assume that the attacker knows the degree value $\mathsf{deg}_{\mathsf{G}}(u)$ of a target node u. Liu and Terzi [19] showed that degree sequences in real networks are heavy-tailed, meaning many vertices have low degrees and few vertices have extremely high degrees. Such uniqueness makes direct re-identification possible by finding the vertex in the anonymized graph G' whose degree matches the known value. One of the most common and straightforward ways to anonymize graphs for degree is the k-degree anonymity approach. When at least k vertices share the same degree, it makes the probability of re-identification for each node at most $1/k$. Compared to other measures, this one uses the least amount of structural information and can scale to larger networks, depending on the optimizations made.

Degree and triangle signature. In this attack, we assume that the attacker knows both the degree $\mathsf{deg}_{\mathsf{G}}(u)$ and the number of incident triangles $\mathsf{Tri}_{\mathsf{G}}(u)$ of a target node u. Together, these form the (n, m)-signature $\sigma_{(n,m)}(u) = (\mathsf{deg}_{\mathsf{G}}(u), \mathsf{Tri}_{\mathsf{G}}(u))$ of a node. Two nodes u, v are considered equivalent if they share the same signature, and a node is k-anonymous if its equivalence class contains at least k nodes. This measure is deliberately positioned between the simpler degree-only signature and the stricter isomorphism signature [8, 6], capturing more structural information than degree alone while remaining less rigid than full subgraph comparison. It is shown that this measure often provides results similar to those of measures that use exact neighborhood structure of a node. Bonello *et al.* [4] and Arsene *et al.* [1] adopt this as their anonymity criterion, anonymizing graphs by deleting up to a fixed budget β of edges so that the maximum number of nodes are anonymous based on (n, m)-signature criterion. Compared to the degree-only approach, this measure is harder to satisfy, since obtaining equal signatures requires coordinating both degree connectivity and incident triangles, making it more sensitive to small structural changes in the network.

5.3 Motivating example.

In the Fig. 5.1, 2 nodes share the same degree and triangle signature and are 2-(deg, tri) anonymous. All the graphs that satisfy k-(deg, tri) anonymity will also satisfy k-degree anonymity, as it is a less strict measure. To achieve 100% 2-(deg, tri) anonymity, we would need to remove only one edge connecting nodes 2 and 5.

Node	(Degree, Triangle) Signature	Is Anonymous?
1	(4,4)	Yes
2	(4,4)	Yes
3	(4,2)	No
4	(3,2)	No
5	(2,1)	No
6	(1,0)	No

Table 5.1: Table showing the degrees and triangles of the nodes from Fig. 5.1 before anonymization. There are 2 anonymous nodes out of 6 in the graph based on the 2-(degree, triangle) signature.

5.4 Levels of Anonymity

A publisher may not require, or be able to afford, that every node be anonymous. The framework therefore supports a tunable level of anonymity, expressed as a target fraction $\tau \in (0, 1]$ of nodes that must be anonymous under the chosen measure. The case $\tau = 1$ requires full anonymity, in which every node shares its signature with at least $k - 1$ others. Any value $\tau < 1$ requires partial anonymity, in which at least $\lceil \tau |\mathsf{V}| \rceil$ nodes are anonymous while the remainder may stay unique.

Node	(Degree, Triangle) Signature	Is Anonymous?
1	(4,2)	Yes
2	(3,2)	Yes
3	(4,2)	Yes
4	(3,2)	Yes
5	(1,0)	Yes
6	(1,0)	Yes

Table 5.2: Table showing the degrees and triangles of the nodes from Fig. 5.1 after anonymization. There are 6 anonymous nodes out of 6 in the graph based on the 2-(degree, triangle) signature.

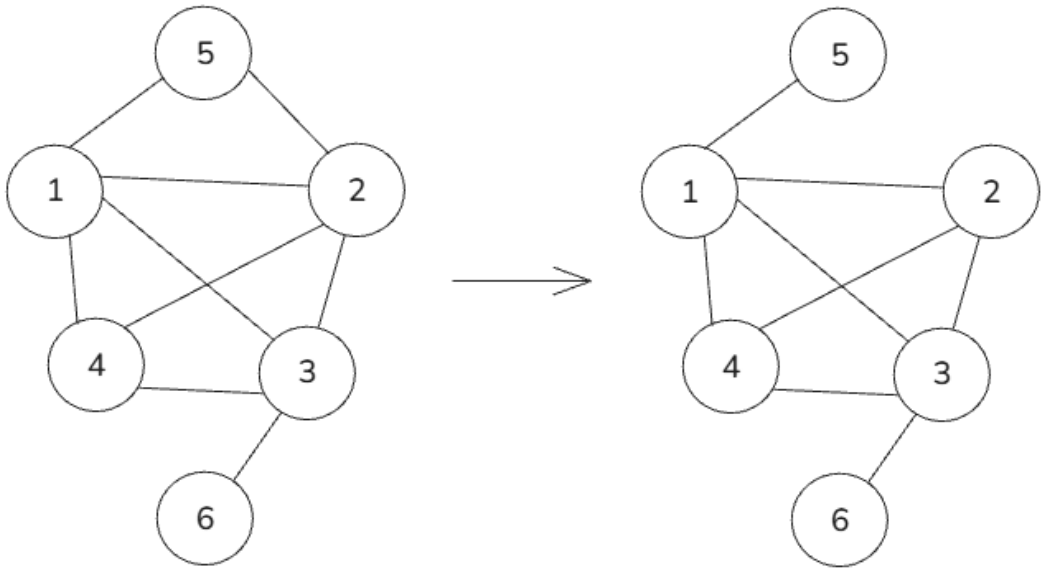

Figure 5.1: Diagram indicating the 2-degree and 2-degree+triangle signature anonymization problem on a toy example of 6 nodes. The left figure indicates the original graph, and the right figure indicates the anonymized graph with modifications.

Partial anonymity is useful for two reasons. First, full anonymity under a strict measure can demand a large number of structural changes or may not even be feasible, so allowing a fraction of nodes to remain unique can substantially reduce the modifications required and hence the utility lost. Second, considering various targets τ lets the user observe how the cost of anonymization grows as the requirement tightens and identify the best target their graph can achieve within a specified budget.

5.5 Privacy vs. utility trade-off problem

The privacy vs. utility trade-off in the network data anonymization problem has been studied extensively in the literature on anonymization techniques [1, 4, 32, 33]. The consensus is that achieving stricter anonymization guarantees, *i.e.* protecting against attackers with intricate knowledge of the structure of the network, will require a high level of structural changes in the network. Naturally, this will negatively impact the utility of the network for downstream analysis.

Utility measures.
To quantify the trade-off between protecting privacy and ensuring the usability of network data for downstream tasks, we use the network utility metric [33]. The utility of the network can be a single metric or a combination of metrics, depending on how the anonymized network will eventually be used. As different anonymization

techniques are used to achieve some privacy guarantee for the network, so far, different metrics such as degree distribution, average clustering coefficient, average shortest path between node pairs, transitivity, or more complex application utility measures such as epidemic simulation, network robustness (fraction of nodes in the largest connected component), betweenness centrality, community structure preservation, etc. are used. Various studies have shown that complex network analysis tasks depend on the underlying structure of the network, modelled by the simpler network properties. Therefore, when we want to optimize for the utility of a downstream task, we consider an important measure, that is, betweenness centrality, as considered by previous work in this field [4, 33].

Unlike degree-based properties, betweenness centrality depends on global graph structure. Consequently, small perturbations to strategically important edges may substantially alter shortest paths and redistribute centrality scores across the network. Existing literature therefore suggests that utility degradation for betweenness centrality is often more severe than for local structural measures when stronger anonymization guarantees are enforced [33]. In particular, anonymization methods based on edge addition or deletion may unintentionally remove bridge edges, causing significant changes to the set of most central nodes.

5.6 Problem Statement

The anonymization problem has two competing objectives: maximize the anonymity of the graph, and minimize the number of structural modifications while keeping the utility loss within a budget. We do not optimize both objectives jointly. Instead, we hold one criterion at a chosen level and optimize the remaining quantity. We implement two solving modes.

Problem 5.1 (Minimize modifications)**.** *Given a graph* G*, an anonymity measure, a target fraction* $\tau \in (0, 1]$*, and an optional betweenness-centrality preservation level* ρ*, find an anonymized graph* G' *that makes at least* $\lceil \tau |\mathsf{V}| \rceil$ *nodes anonymous and keeps the centrality cost within the budget set by* ρ*, while minimizing the number of edge modifications.*

Problem 5.2 (Maximize anonymity)**.** *Given a graph* G*, an anonymity measure, and a betweenness-centrality preservation level* ρ*, find an anonymized graph* G' *that maximizes the number of anonymous nodes while keeping the centrality cost within the budget set by* ρ*.*

Problem 5.1 fixes the anonymity requirement and reports the minimum cost of meeting it; varying τ (and ρ, where used) maps out where anonymization is feasible and how expensive it is. Problem 5.2 fixes the utility budget and reports the most anonymity attainable within it for different values of ρ.

6

Approach

As discussed in Chapter 5, this thesis aims to solve for network anonymity while also preserving some utility of the network. This utility measure can either be the amount and type of anonymity to be achieved or structural information of a network, such as the betweenness centrality values of the nodes. The proposed framework models social network anonymization as a constraint optimization problem in which users combine anonymity and utility constraints, while the system enforces structural feasibility rules.

Constraint programming translates the requirements that a valid solution must satisfy into solver criteria; the solver prunes values from variable domains based on the feasibility of each constraint, exploring only the part of the search space consistent with all constraints.

The remainder of this chapter is organized as follows. Section 6.1 introduces relevant notation for understanding the approach. Section 6.2 presents three types of constraints used in the model: structural constraints that map the graph onto solver variables, anonymity constraints that encode the k-deg k-$(\mathsf{deg}, \mathsf{Tri})$ anonymity measures, and utility constraints for optimizing betweenness centrality. Section 6.3 walks through the elements of the optimization problem, such as pruning the model Section 6.3.1, solving with pumpkin Section 6.3.2, and Section 6.3.3 presents the overall anonymization pipeline.

6.1 Notation

The original graph is $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with $|\mathsf{V}|$ vertices; the anonymized graph $\mathsf{G}' = (\mathsf{V}, \mathsf{E}')$ shares the same vertex set. For a node $\mathsf{u} \in \mathsf{V}$ we apply the concepts of neighborhood, degree, and incident triangles for the 1-hop neighborhood [4]:

$$\mathsf{N}(\mathsf{u}) := \{\, v \in \mathsf{V} : \{\mathsf{u}, v\} \in \mathsf{E},\ u \neq v\}, \quad \mathsf{deg}_{\mathsf{G}}(\mathsf{u}) := |\mathsf{N}(\mathsf{u})|$$
$$\mathsf{Tri}(\mathsf{u}) := \big\{\, \{\mathsf{u}, v, w\} : v, w \in \mathsf{N}(\mathsf{u}),\ \{v, w\} \in \mathsf{E},\ u \neq v \neq w\big\},$$

Anonymization operates on a subset of edges that could possibly exist by connecting the vertices in the original graph. The set of *candidate edges* the solver considers for modifications is

$$\mathcal{E} = \begin{cases} \{\{u, v\} : u, v \in \mathsf{V},\ u \neq v\} & \text{(addition/deletion mode)}, \\ \mathsf{E} & \text{(deletion-only mode)}, \end{cases}$$

and the incident-edge set of a node is $\delta(\mathsf{u}) := \{\, \{\mathsf{u}, v\} \in \mathcal{E} : v \in \mathsf{V}\,\}$.

6.2 Constraint model

We can broadly classify our constraints into three categories: structural constraints, anonymity constraints, and utility constraints. *Structural* constraints help us map the neighborhood information of the nodes in our graphs, such as their edges, node degrees, and incident triangles, onto solver variables so that the solver can track these assignments as it modifies the graph and explore only valid graph configurations. *Anonymity* constraints are used to encode the privacy guarantee of the structural information shared by the nodes in the graph. We use

the structural variables to express the rules of anonymity. We define a budget for the number of modifications allowed, which will be explored in Section 6.3. *Utility* constraints bound how much theoretical utility should be preserved by the modifications performed in the bi-objective optimization problem, which optimizes or satisfies the anonymity target under bounded utility.

6.2.1. Structural Constraints.

We begin with the logical building blocks of our model: the structural constraints. They describe the possible graph edges, each node's degree, and the rules for edge modification.

Edge variables. For each candidate edge, $e \in \mathcal{E}$ we introduce a binary variable x_e that indicates whether e is present in G', with $\mathsf{D}(x_e) = \{0, 1\}$.

Degree. An integer variable d_u equals the number of edges incident on the node:

$$d_u := \sum_{e \in \delta(u)} x_e, \qquad 0 \leq d_u \leq |\mathsf{V}| - 1 \qquad \forall u \in \mathsf{V}. \tag{6.1}$$

Minimum degree constraint. The need for this constraint arises from the fact that a graph with all nodes or any nodes having a shared 0-degree signature can satisfy k-anonymity but will reduce the usefulness of the node in any analysis. We do not consider any original graphs with single-node components, *i.e.*, any disconnected nodes. Similarly, for anonymized graphs, we also need to ensure that the solver does not trivially solve the constraints by setting node degrees to 0.

$$d_u \geq 1 \quad \forall u \in \mathsf{V}$$

Edge Modification. An important step in the anonymization problem is defining when an edge is considered modified. This will ensure we map the original and new graph edges correctly and will help us keep track of the total budgeted modifications.

Let $m_e \in \{0, 1\}$ be a binary variable indicating whether the edge $e \in E$ differs from G. Both deletion of an existing edge and addition of a new edge count as one modification, respectively.

$$m_e := \begin{cases} 1 - x_e & \text{if } e \in \mathsf{E} \quad \text{(For existing edges, deletion is a modification)}, \\ x_e & \text{if } e \notin \mathsf{E} \quad \text{(For new edges, addition is the modification)}. \end{cases} \tag{6.2}$$

The total number of modifications performed would be $M_{\text{total}} = \sum_{e \in \mathcal{E}} m_e$. We set the objective to minimize the number of modifications so as to preserve as much of the original graph as possible.

6.2.2. Anonymity Constraints

An anonymity constraint encodes the requirement that nodes need to be indistinguishable to an attacker under a given measure: a node is anonymous when enough other nodes share its structural signature. We encode two measures of increasing strictness. The first, k-deg anonymity, considers only node degree; the second, k-(deg, Tri) anonymity, additionally requires matching incident-triangle counts. We will express these requirements as constraints.

Choice of Big-M formulation. To encode equivalence of degree or triangle signatures: $d_u = d_v$, a natural choice would have been reification and a global cardinality constraint using the reified variables [12]. Reification binds the equivalence to a Boolean variable *i.e.* $b_{uv} \leftrightarrow (d_u = d_v)$, which a dedicated propagator then enforces in both directions. Pumpkin does not currently provide such a dedicated *cumulative constraint for cardinality*, which would reason over the whole graph instead of pairwise equalities. Implementing a propagator together with an explanation procedure for lazy clause generation lies outside the scope of this work. We aim to assemble anonymity and utility requirements from simple, independent constraints to support our framework and therefore adopt a big-M formulation in one direction that expresses the same conditional requirement in linear inequalities, with the required anonymity count implicitly enforcing the other direction of this constraint.

k-deg **Anonymity**

In k-deg anonymity, the parameter k indicates the number of nodes that share the same structure information (in this case, degree) and are indistinguishable from each other.

For each unordered pair $\{u, v\}$, let $b_{uv} \in \{0, 1\}$ be a binary variable that indicates whether nodes u and v have the same degree. We use the big-M formulation, where M = $|\mathsf{V}| - 1$, which is the maximum degree possible.

$$d_u - d_v + M\, b_{uv} \leq M, \qquad d_v - d_u + M\, b_{uv} \leq M \qquad \forall \{u, v\},\ u \neq v, \tag{6.3}$$

so that $b_{uv} = 1 \Rightarrow d_u = d_v$, while $b_{uv} = 0$ leaves the pair unconstrained. We chose to encode a less restrictive one-way implication to reduce the number of pairwise equality constraints. As we try to maximize the number of pairs of nodes sharing the same degree, the solver is incentivized to set the binary indicator b_{uv}=1 when any match is found. Let $matches_u = \sum_{v \in \mathsf{V}, v \neq u} b_{uv}$ count the nodes sharing the same degree as u.

We encode full and partial anonymity separately. For *full anonymity* ($\tau = 1$), every node must share its degree with at least $k - 1$ others:

$$matches_u \geq k - 1 \qquad \forall u \in \mathsf{V}. \tag{6.4}$$

For partial anonymity, let $a_u \in \{0, 1\}$ be a binary variable indicating whether node u satisfies $k-$anonymity and let $P(u)$ be the set of candidate partners of u, that is, the nodes $v \neq u$ which is suitable for k-deg anonymity based on the difference in the degrees of the nodes and the modification budget. $matches_u = \sum_{v \in P(u)} b_{uv}$. The equivalence $a_u = 1 \leftrightarrow matches_u \geq k - 1$ is enforced by

$$matches_u \geq (k - 1)\, a_u, \qquad matches_u \leq |P(u)|\, a_u, \tag{6.5}$$

where the coefficient $|P(u)|$ is the tightest constant that bounds $matches_u$, so that $a_u = 0$ forces $matches_u = 0$. Now we ensure that the target fraction $\tau \in [0.8, 1]$ of nodes are anonymous

$$\sum_{u \in \mathsf{V}} a_u \geq \lceil \tau\, |\mathsf{V}| \rceil. \tag{6.6}$$

k-(deg, Tri) **Anonymity**

This is a stricter anonymity measure that additionally accounts for the number of triangles a node participates in, so that two nodes are not distinguishable from each other only when they share both degree and incident triangle count.

We use $\mathcal{E} = \mathsf{E}$ as the deletion-only mode is better for pruning the search space for a stricter anonymity guarantee. A triangle $T = \{u, v, w\} \in \mathsf{Tri}(u)$ is present in G' if and only if its three edges $\{u, v\}, \{v, w\}, \{u, w\}$ are all present. We track this using a binary variable $alive_T \in \{0, 1\}$ and use the edge modification indicator from Eq. (6.2):

$$alive_T \leq x_e \ \ \forall e \in T, \qquad alive_T \geq \sum_{e \in T} x_e - 2. \tag{6.7}$$

Two nodes match under this measure when they agree on both degree and triangle count. We test each of these separately and then combine them.

Comparing triangle counts. The number of triangles a node still has after deletions is

$$t_u = \sum_{T \in \mathsf{Tri}(u)} alive_T.$$

To test whether two nodes have equal triangle counts, we introduce an indicator b_{uv}^{Tri} that, when set, forces $t_u = t_v$. The triangles common to u and v appear in both counts and cancel, so we post the constraint only over the triangles unique to each node. The largest difference in triangle count that any eligible pair of nodes can have in the original graph is, the maximum between the triangle count of each node excluding the shared triangles of both: $M_{uv} = \max\big(|\mathsf{Tri}(u) \setminus \mathsf{Tri}(v)|,\ |\mathsf{Tri}(v) \setminus \mathsf{Tri}(u)|\big)$. We then use this as the M value for our triangle equality constraints:

$$\begin{aligned}\sum_{T\in\mathsf{Tri}(u)\setminus\mathsf{Tri}(v)} alive_T \;-\; \sum_{T\in\mathsf{Tri}(v)\setminus\mathsf{Tri}(u)} alive_T \;+\; M_{uv}\, b_{uv}^{\mathsf{Tri}} \;\le\; M_{uv}\\ \sum_{T\in\mathsf{Tri}(v)\setminus\mathsf{Tri}(u)} alive_T \;-\; \sum_{T\in\mathsf{Tri}(u)\setminus\mathsf{Tri}(v)} alive_T \;+\; M_{uv}\, b_{uv}^{\mathsf{Tri}} \;\le\; M_{uv},\end{aligned} \tag{6.8}$$

This ensures that when $b_{uv}^{\mathsf{Tri}} = 1$ the surviving triangle counts of u and v are equal.

Combining degree and triangle count into a signature. Degree equality is expressed by an indicator b_{uv}^{deg}, posted as in Eq. (6.3). A pair of nodes matches on the full signature only when both the degree and the triangle indicators are set to 1, so b_{uv} is their conjunction $b_{uv} = b_{uv}^{\mathsf{deg}} \wedge b_{uv}^{\mathsf{Tri}}$, encoded linearly as

$$b_{uv} \le b_{uv}^{\mathsf{deg}}, \qquad b_{uv} \le b_{uv}^{\mathsf{Tri}}, \qquad b_{uv} \ge b_{uv}^{\mathsf{deg}} + b_{uv}^{\mathsf{Tri}} - 1. \tag{6.9}$$

From here the model proceeds exactly as for k-deg anonymity: the signature-match indicators b_{uv} are considered in place of degree-match indicators, and the per-node counting, the anonymity indicators a_u, and the target τ follow Eq. (6.5)–Eq. (6.6).

6.2.3. Utility Constraints.

In existing works, the change in utility measures in a network is only passively calibrated as a by-product of anonymization. It is naturally presumed that any utility is preserved with minimum modifications to the network. We postulate that optimizing for utility could be more beneficial for preserving utility in a network. To be able to preserve utility actively, we have implemented utility as a constraint. We now explore how to constrain betweenness centrality, which identifies the nodes that act as bridges in the network, by making modifications near such nodes expensive.

Betweenness Centrality Cost Function. Let $\mathsf{bc}(u)$ denote the normalized betweenness centrality of node u as derived earlier in Chapter 5 in the original graph G, computed before any anonymization. Following the intuition that it would be more important to preserve the edges between highly central nodes, we weight each candidate edge $\{u, v\} \in \mathcal{E}$ by the product of its endpoints' centralities, so we get the cost of modifying an edge as:

$$c_{uv} = \mathsf{bc}(u) \cdot \mathsf{bc}(v). \tag{6.10}$$

An edge between two high-centrality nodes therefore receives a high cost, while an edge between two peripheral nodes costs little to modify; this cost is fixed before solving and does not depend on the solver's decisions. The *total centrality cost* of a solution, C_{bc} sums this cost over every edge the solver actually modifies:

$$C_{\mathsf{bc}} = \sum_{\{u,v\}\in\mathcal{E}} c_{uv}\, m_{uv}, \tag{6.11}$$

where m_{uv} is the modification indicator of Eq. (6.2). An unmodified edge contributes nothing to C_{bc}; a modified edge contributes its full cost c_{uv}. Thus C_{bc} measures how much centrality-weighted disruption a given solution causes, rather than merely how many edges it changes.

Bounding the cost. We bound C_{bc} by a fraction of the cost the solver could incur in the worst case, that is, if every candidate edge were modified ($m_{uv} = 1$). Let $\rho \in [0, 1]$ be the user defined *BC preservation parameter*, denoting the fraction of structure that must be preserved, and let $\alpha = 1 - \rho$ be the corresponding *loss budget*. The constraint

$$C_{\mathsf{bc}} \le \alpha \sum_{\{u,v\}\in\mathcal{E}} c_{uv} \tag{6.12}$$

restricts the solver to solutions whose total cost does not exceed the fraction α of the total cost pool $\sum_{\{u,v\}\in\mathcal{E}} c_{uv}$. A small α (high preservation ρ) provides a small budget, pushing the solver away from edges incident to high-centrality nodes; a large α relaxes the restriction, allowing the solver to prioritize minimizing modifications or maximising anonymity, over preserving centrality.

This value is derived from the candidate-edge set $\mathcal{E}$, which differs between the deletion-only and add/delete models; a given ρ corresponds to the same fraction of each model's own cost pool but not to the same absolute budget across models. We therefore treat ρ as a budget parameter rather than a guarantee and measure the resulting centrality preservation directly in Chapter 8.

Since the coefficients c_{uv} are floating-point values, both sides of Eq. (6.12) are scaled by a fixed integer factor $S = 10\,000$ and rounded before the constraint is posted, so that all subsequent propagation is performed in integer arithmetic.

Mode based approach. The betweenness-centrality models for the anonymity constraints differ in the candidate-edge set and some constraints. For constraining betweenness centrality in k-deg mode (edge add/delete), the cost budget Eq. (6.12) is accompanied by a cap on the allowed additions. This is necessary because an edge between two peripheral nodes has cost $c_{uv} \approx 0$, so without such a cap the solver could satisfy the cost budget while freely restructuring the graph through a lot of edge additions to peripheral nodes. While this may not violate betweenness centrality constraints directly, it will distort the graph significantly and in turn, affect the average betweenness centrality scores and rank correlation.

Second, a cap on incident edge modifications per node protects the most central nodes individually. The global cost budget bounds the total centrality-weighted disruption, but we specifically aim to preserve the rank correlation of the sets of top 10% most central nodes before and after anonymization. So, for each node v_k among the top 10% nodes by centrality, we bound the number of its incident edges that may be modified:

$$\sum_{e \in \delta(v_k)} m_e \;\le\; \max\Big(0,\; \Big\lfloor \mathsf{deg}_{\mathsf{G}}(v_k)\,\alpha \;/\; (\mathsf{bc}(v_k)/\mathsf{bc}_{\max}) \Big\rfloor\Big). \tag{6.13}$$

The budget grows with the node's degree $\mathsf{deg}_{\mathsf{G}}(v_k)$, and with the loss budget $\alpha = 1 - \rho$; it shrinks with the node's relative centrality $\mathsf{bc}(v_k)/\mathsf{bc}_{\max} \in (0, 1]$, which appears in the denominator so that more central nodes have a tighter cap on their modifications.

6.3 The Optimization Problem

By combining the above-defined constraints in different configurations, we instantiate a constraint optimization problem. In this section, we will further define the optimizations made to reduce the search space of feasible solutions, the solver mode chosen, and represent the configurations Table 7.2 in a single generalized pipeline, which forms the basis for our solver.

6.3.1. Objective Bounds and Search-Space Reduction

In the *Linear SAT-UNSAT* mode of operation, we start with an objective value that is the loosest bound possible to enable the solver to find a feasible solution quickly. We then reduce the value of the objective (in minimize mode) till we reach the point of unsatisfiability. For minimizing modifications, we need to first find the objective value of modifications that is satisfiable in the first pass of the solver. We use the same value for pruning unsatisfiable pairs of nodes as described below.

Posting an indicator as described in Eq. (6.3), Eq. (6.8) for every node pair is wasteful when most pairs can never match within a reasonable number of modifications, especially in scale-free networks (a few nodes have a massive number of connections). Before posting the anonymity constraints, we therefore prune pairs whose signatures are too far apart to be reconciled.

Solonets *et al.* [28] implement optimization by maintaining the degree sequence in sorted order during solving, using sorted arrays in Gecode, and Erdős–Gallai criterion for a satisfiable degree sequence of a graph. They also note that in the sorted sequence, k nodes of equal degree can only occur within a window of $k - 1$ neighbors, so the anonymity constraint ranges over $2k - 1$ variables rather than pairwise constraints over $|\mathsf{V}|$. Due to limitations on the constraints available, we cannot use the sorting constraint and so implement a static bound before solving.

We establish a bound β on the maximum number of modifications any optimal solution can require. For the k-deg measure, β is obtained by sorting the degree sequence $d_1 \le d_2 \le \cdots \le d_{|\mathsf{V}|}$ and partitioning it into consecutive groups of k nodes; within each group, the node with maximum degree is the target degree, and we find the number of operations needed for all other nodes to reach the target (assuming addition operations). The total count of modifications needed for k-deg would be:

$$\beta = \sum_{j=1}^{\lfloor |\mathsf{V}|/k \rfloor} \sum_{i \in G_j} \left(d_{\max(G_j)} - d_i \right) \tag{6.14}$$

where G_j is the j-th group/set of k consecutive nodes and $d_{\max(G_j)}$ is its largest degree. Any solution that requires more modifications than β can be discarded, so β is a valid upper bound on the modification count of any optimal solution.

Lemma 6.1 (Degree-gap pruning)**.** *Let β be the number of edge modifications applied to G as defined above. A single modification changes the degree of exactly two nodes by ± 1. Since we allow only addition or deletion operations, it can only bridge the gap between the degrees of any pair of nodes by at most 1. Hence, if $|\mathsf{deg}_{\mathsf{G}}(u) - \mathsf{deg}_{\mathsf{G}}(v)| > \beta$, then $d_u = d_v$ is unattainable within β modifications, and the indicator b_{uv} may be omitted without excluding any feasible solution.*

6.3.2. Solving with Pumpkin

We solve each model with Pumpkin[1], a lazy-clause-generation constraint solver.
Pumpkin combines constraint propagation with conflict-driven clause learning: when search reaches a dead end, it derives a reusable nogood explaining the failure, prunes the corresponding region of the search space, and backtracks. This is useful as the anonymity indicators and modification variables induce many interacting constraints whose infeasible combinations are best discovered and recorded once.

To optimize rather than merely satisfy, Pumpkin applies a *Linear SAT-UNSAT* scheme to the objective. Given an objective variable o to minimize, it repeatedly searches for a solution with $o \leq \beta$; each solution found tightens β to $\beta' = o - 1$, and the procedure continues until the bounded problem is unsatisfiable. The last solution found is then a proven optimal one, since unsatisfiability of $o \leq \beta'$ certifies that no better solution exists. If the solver does not reach unsatisfiability before the timeout, the last solution found is not proven optimal, but it is still the best solution so far.

In minimize-modifications mode, the solver returns the fewest edge modifications that achieve the target anonymity τ. In maximize-anonymity mode, it returns a maximal set of anonymous nodes within the betweenness-centrality budget ρ.

6.3.3. The Anonymization Pipeline

Algorithm 1 summarizes the solving pipeline for a single (graph, configuration, parameter) combination. Degrees, triangle counts, and betweenness centrality are computed once before posting the constraint model. The greedy cap β (Eq. (6.14)) bounds the total modifications as defined in Section 6.3.1. In minimize-modifications mode, an anonymity target τ is posted and M is minimized; in maximize-anonymity mode, the anonymity achieved is maximized for the betweenness centrality preservation budget ρ. A single call to Pumpkin's *Linear SAT-UNSAT* solver returns the best solution, modifications for the best solution so far, and an optimality flag.

[1] `https://github.com/consol-lab/pumpkin`

Algorithm 1: Anonymization pipeline

Input: graph G, choice $c \in \{1, 2, 3, 4\}$, parameter k, target τ (choices 1-3) or preservation ρ (choice 4)
Output: anonymized graph G', modification count M, optimality flag

compute $\mathsf{deg}_{\mathsf{G}}(u)$, $|\mathsf{Tri}(u)|$, $\mathsf{bc}(u)$ stats for all $u \in \mathsf{V}$;
compute β via Eq. (6.14);
if $c \in \{1, 2, 3\}$ **then**
  post $M \leq \beta$;
post structural constraints;
post anonymity constraints (k-deg if $c \in \{1, 3\}$; k-(deg, Tri) if $c \in \{2, 4\}$) and β for pair-pruning;
if $c \in \{3, 4\}$ **then**
  post BC budget Eq. (6.12) and additional constraints Section 6.2.3;
if $c \in \{1, 2, 3\}$ **then**
  post $\sum_u \text{is\_anon}[u] \geq \lceil \tau |\mathsf{V}| \rceil$;
  $o \leftarrow M$;
else
  $o \leftarrow |\mathsf{V}| - \sum_u \text{is\_anon}[u]$;
$(\mathsf{G}',\ \text{flag}) \leftarrow$ *Linear SAT-UNSAT*$(o,\ \textsc{Minimize})$;
return G', M, flag;

7

Experiments

To understand the effectiveness of the approaches seen previously and their contribution to our research goals, we have set up an experimental pipeline. This experimental setup uses benchmark datasets from Network Repository [25] to evaluate the anonymity and utility constraints. We have used graphs of varying sizes and properties and compared various factors such as starting anonymity level, graph density, and graph size for solved and unsolved instances. Subsequently, we will cover the resource requirements for computation, configurations used, solver-specific parameters, evaluation criteria, etc., in this chapter.

7.1 Datasets

We evaluate the k-degree anonymity model on 30 network datasets from the Network Repository. All graphs are pre-processed into undirected, unweighted, simple graphs with no self-loops, and the nodes are relabeled to follow cardinal numbering. There are no isolated nodes in the graph, as social network analysis generally does not care about such isolated nodes, so if there is more than one connected component in the graph, each component has at least 2 nodes. Table 7.1 summarizes some structural properties of each dataset. The number of nodes and edges also gives us information about the density of edges in the graph, calculated as $\text{density} = 2|E|/(|V|(|V|-1))$. The collection covers different sizes ($n \approx 30$ to $2k$) ($e \approx 30$ to $39k$ edges) and densities (0.008 to 1.0).

7.2 Solver and infrastructure

All experiments were run on the DelftBlue HPC cluster at TU Delft. The comupte nodes run on RedHat Enterprise Linux 9.8 OS. The nodes are each equipped with two 24-core Intel Xeon Gold 6248R (*Cascade Lake*) processors (48 cores per node) at a base clock of 3.0 GHz, 185 GB of RAM [10]. Each job was allocated 1 core, i.e., 1 CPU, and 4 GB of memory, which is the maximum permissible on the selected partition, with a wall-clock limit of 14,100 s (3 h 55 min). Within each task, the solver is given a per-graph time budget of $3{,}600$ s, and graphs are solved across the benchmark set for that parameter combination. The solver binary was compiled in Rust release mode and uses the Pumpkin constraint-programming solver (Section 6.3.2) to solve each posted model. We explore the experimental pipeline below with the configurations used and details of centrality evaluation with and without the BC-weighted modification constraint.

7.3 Configurations and parameters

Table 7.2 lists the four configurations evaluated. All configurations are run for $k \in \{2, 3\}$. Each SLURM array task fixes one configuration choice, value of parameter k, minimum target anonymity to be reached, and optionally the BC preservation budget ρ. It solves every graph in the benchmark set for that combination sequentially, writing one result record per graph to a shared results log. We first check whether the graphs already satisfy the requested anonymity criterion; if no modifications are needed, we return them immediately,

Dataset	Domain	Nodes	Edges	Density
aves-sparrow-social-2009	Animal social	31	211	0.45
aves-sparrow-social-2010	Animal social	40	305	0.39
aves-sparrowlyon-flock-season2	Animal social	46	348	0.33
aves-thornbill-farine	Animal social	62	1,151	0.61
aves-weaver-social-03	Animal social	42	152	0.17
aves-weaver-social-11	Animal social	39	99	0.13
ca-sandi_auths	Collaboration	86	124	0.03
chesapeake	Road	39	170	0.22
econ-beause	Economic	507	39,428	0.31
email-enron-only	Email	143	623	0.06
fb-pages-food	Social	620	2,091	0.01
ia-email-univ	Interaction	1133	5,451	0.008
southernwomen	Interaction	18	64	0.41
johnson8-4-4	DIMACS	70	1,855	0.55
mammalia-voles-bhp-trapping-21	Animal social	86	138	0.03
mammalia-voles-plj-trapping-46	Animal social	58	69	0.04
mammalia-voles-rob-trapping-04	Animal social	80	119	0.03
mammalia-voles-rob-trapping-25	Animal social	84	128	0.03
mammalia-voles-rob-trapping-47	Animal social	98	184	0.03
mammalia-voles-rob-trapping-62	Animal social	73	96	0.03
opsahl-southernwomen	Interaction	18	64	0.41
p0040	Miscellaneous	63	111	0.05
reptilia-tortoise-network-bsv-1998-inact	Animal social	36	53	0.08
reptilia-tortoise-network-cs-2014	Animal social	39	30	0.04
reptilia-tortoise-network-fi-2013	Animal social	84	83	0.02
reptilia-tortoise-network-pv	Animal social	35	66	0.11
rgg010	Miscellaneous	10	45	1.0
rt-retweet	Retweet	96	117	0.02
soc-dolphins	Social	62	159	0.08
tech-routers-rf	Technological	2,113	6632	0.02

Table 7.1: Summary of the 30 benchmark graph datasets. Density and edge count are for the undirected graph after preprocessing.

without invoking the solver. Every graph is solved with a single call to Pumpkin's *Linear SAT-UNSAT* (LSU) optimization procedure Section 6.3.2.

Configuration	Mode	Measure / utility
k-Degree	Min. modifications	k-deg
k-(Degree, Triangle)	Min. modifications	k-(deg, Tri)
BC + k-Degree	Min. modifications	k-deg with BC budget
BC + k-(Degree, Triangle)	Max. anonymity	k-(deg, Tri) with BC budget

Table 7.2: The four configurations evaluated along with the mode of solving.

A solution returned as *Optimal* by Pumpkin is marked proven; a solution returned as *Satisfiable* after the time limit is marked as best-found only. Runs for which Pumpkin returns *Unsatisfiable* or *Unknown* are recorded as "infeasible" or "timed out" respectively, and are excluded from analysis.

Table 7.3 lists the parameter values used per configuration. For the minimize-modifications configurations, we select anonymity targets for K-Degree mode from $\tau \in \{0.80, 0.85, 0.90, 0.95, 1.00\}$. as for K-(Degree, Triangle), we add 5 additional targets, as a lot of graphs start at a lower anonymity level for the stricter measure. For BC + k-deg we vary BC preservation level and target τ, giving $5 \times 5 = 25$ combinations per graph. For the BC + K-(Degree, Triangle) configuration only ρ is varied across the same five levels.

Configuration	Objective	k	Target τ	BC budget ρ
k-deg	min. mod.	2	0.80–1.00 (step 0.05)	—
		3	0.70–0.95 (step 0.05)	—
k-(deg, Tri)	min. mod.	2, 3	0.55–1.00 (step 0.05)	—
BC + k-deg	min. mod.	2, 3	0.80–1.00 (step 0.05)	0.70–0.95 (step 0.05)
BC + k-(deg, Tri)	max. anon.	2, 3	—	0.70–0.95 (step 0.05)

Table 7.3: Parameters per configuration. A dash means the parameter does not apply to that configuration.

7.4 Evaluation metrics

We compute betweenness centrality both on the original and anonymized graph and compare the resulting node ranking against the original using rank-biased overlap (RBO) [30], a top-weighted similarity measure for ranked lists. This measure can compare the values and relative ranking of the most central nodes, is robust to ties and does not require a fixed list length for comparison. We focus on the top 10% nodes by original centrality, which is an arbitrary choice that is likely to favor small-world graphs along with the design of the betweenness centrality constraint itself.

For each solved instance, we record several solver metrics, including the number of edge modifications, the achieved anonymity value, the solver time, and an optimality flag indicating whether the solution is proven optimal or best-found within the time limit. To assess centrality preservation, we compute betweenness centrality on the original and anonymized graphs and use the RBO metric.

8

Results

In this chapter, we present the results of our experiments, organized around the research questions stated below. We first summarize the solved instances across all configurations (Q1), then compare the two solving modes of the k-(deg, Tri) measure (Q2), examine the anonymity-centrality trade-off under the BC budget constraint (Q3), and finally evaluate how effective the proxy constraint is for preserving betweenness centrality by examining the correlation between the budget and the actual level of utility preservation (Q4). We then discuss the limitations of our approach for an objective assessment of our methodology.

8.1 Research Questions

The experiments performed are aimed at answering the following main research questions:

Q1 How many instances are solved by each anonymity measure in each configuration?
Q2 What is the trade-off achieved when maximizing anonymity while preserving betweenness centrality by penalizing modifications to important links?
Q3 Can we compare the maximum anonymity achieved and betweenness centrality preserved when solving in minimize modifications vs. maximize anonymity modes?
Q4 Is the proxy constraint for preserving betweenness centrality effective?
We now evaluate the answers to our research questions with the help of figures and statistics. Each question is addressed sequentially and the relevant figures and referenced when needed.

8.1.1. Q1. Instances solved

We have summarized the instances solved for each configuration defined in Table 7.2. Table 8.1–Table 8.8 report the number of solved instances, timeouts, and unsatisfiable outcomes for each configuration at every anonymity target or BC budget level. Visualizations for the same can be accessed from the appendix Chapter A.

k-deg **modes (Table 8.1 and Table 8.2).** The k-deg measure is the most feasible to achieve when $k = 2$. For $k = 2$, no instance times out below a 90% target and fewer than 10 time out even at the 100% target. For $k = 3$, the timeout rate grows with the target, reaching 15 out of 28 instances at the 100% target (54%) because of the more demanding anonymity requirement. In both cases, a large fraction of instances are already anonymous before solving, particularly at lower targets—at 80%, 26 of 29 instances for $k = 2$ and 20 of 29 for $k = 3$ required no modifications at all.

k-(deg, Tri) **modes (Table 8.3 andTable 8.4).** The k-(deg, Tri) measure is harder to satisfy as seen by the number of instances solved to optimality. The timeout rate remains lower than what was observed in k-deg modes at most 5 instances per target, suggesting that the solver can find feasible solutions even for demanding instances. This we conjecture, is due to the reduction in search space for the k-(deg, Tri) mode as we allow only deletion operations.

BC-constrained modes (Table 8.5–Table 8.8). Adding the BC budget constraint does not substantially increase the timeout rate for k-deg+BC: 12–16 timeouts per ρ level for $k = 2$ and 19–24 for $k = 3$, comparable to the unconstrained case. Mean modifications are relatively lower for this mode as the objective is to minimize modifications, and the constraints implemented Eq. (6.13) also help cap the modifications.

For k-(deg, Tri)+BC in maximize-anonymity mode, timeouts are rare (at most 1 per ρ level), and achieved anonymity is high (mean above 96% for $k = 2$ and above 94% for $k = 3$), although mean modifications are far larger—between 73 and 99 edges per instance, as a result of deletion of many edges to maximize the anonymity count within the cost budget.

In summary, we find that k-deg instances are solved quickly with fewer modifications compared to k-(deg, Tri) instances, which require more extensive modifications and take longer regardless of k or BC budget. k-(deg, Tri) mode has fewer timeouts due to the ease of finding a solution fast in a smaller search space, which allows only deletion operations.

Note. Mean solving time and modification counts are omitted from the tables because the set of instances requiring modifications changes with each target level, making these means non-monotonic and misleading. For a fixed instance, both metrics increase monotonically with the anonymity target and the strictness of the measure.

Target	Total	T/O	Already	Optimal	Sub-opt
80%	29	0	26	3	0
85%	29	0	23	5	1
90%	29	1	21	6	1
95%	29	2	19	7	1
100%	27	9	4	11	2

Table 8.1: Results for 2-deg anonymity ($k = 2$) by anonymity target. **T/O** = Timed out before finding any feasible solution. **Already** = satisfied before solving (no modifications needed). **Optimal** = proven optimal; solver terminated before the time limit. **Sub-opt** = feasible solution returned at the time limit, optimality unproven.

Target	Total	T/O	Already	Optimal	Sub-opt
80%	29	1	20	6	2
85%	29	1	19	5	4
90%	29	3	19	2	5
95%	29	5	16	4	4
100%	28	15	3	6	4

Table 8.2: Results for 3-deg anonymity ($k = 3$) by anonymity target. **T/O** = Timed out before finding any feasible solution. **Already** = satisfied before solving. **Optimal** = proven optimal; solver terminated before the time limit. **Sub-opt** = feasible solution returned at the time limit, optimality unproven.

Target	Total	T/O	Already	Optimal	Sub-opt
55%	29	1	23	1	4
60%	29	1	23	1	4
65%	29	1	19	5	4
70%	29	1	17	7	4
75%	29	1	17	6	5
80%	28	1	16	6	5
85%	28	1	14	5	8
90%	27	1	8	10	8
95%	27	2	6	9	10
100%	26	2	2	12	10

Table 8.3: Results for 2-(deg, Tri) anonymity ($k = 2$) by anonymity target. **T/O** = Timed out before finding any feasible solution. **Already** = satisfied before solving. **Optimal** = proven optimal; solver terminated before the time limit. **Sub-opt** = feasible solution returned at the time limit, optimality unproven.

8.1.2. Q2. Maximum Anonymity achieved under BC constraint

Fig. 8.1 and Fig. 8.2 show for each graph the achieved k-(deg, Tri) anonymity plotted against the BC preservation budget ρ, grouped by graph density. Diamond markers in the before section indicate the graph's starting anonymity before any modification. Lines connect the budget levels for the same graph. Filled circles are

Target	Total	T/O	Already	Optimal	Sub-opt
55%	29	1	17	6	5
60%	29	1	17	6	5
65%	29	1	17	6	5
70%	28	1	16	4	7
75%	28	1	14	5	8
80%	28	3	13	5	7
85%	27	3	8	7	9
90%	27	3	4	10	10
95%	27	5	3	10	9
100%	27	5	2	8	12

Table 8.4: Results for 3-(deg, Tri) anonymity ($k = 3$) by anonymity target. **T/O** = Timed out before finding any feasible solution. **Already** = satisfied before solving. **Optimal** = certified by Pumpkin's `proven_optimal` flag. **Sub-opt** = feasible solution returned at the time limit, optimality unproven.

ρ	Total	T/O	Already	Optimal	Sub-opt	BC Sat.	Mean Anon.
0.70	143	14	93	32	3	127	96.4%
0.80	143	13	93	32	4	126	96.8%
0.85	142	12	93	34	2	125	96.6%
0.90	143	16	93	32	1	123	96.5%
0.95	141	14	93	27	2	116	96.2%

Table 8.5: Results for 2-deg+BC ($k = 2$) by BC preservation level ρ. **T/O** = Timed out before finding any feasible solution. **Already** = satisfied before solving. **Optimal** = proven optimal; solver terminated before the time limit. **Sub-opt** = feasible solution returned at the time limit, optimality unproven. **BC Sat.** = instances where the BC preservation constraint was satisfied. Mean Anon. is the mean achieved anonymity over all solved instances.

proven optimal solutions (i.e., 100% anonymity), and the hollow circles are the best solutions found within the time limit.

We plot the budget ρ on the x-axis (rather than the measured betweenness centrality) for this figure to be able to compare whether a tighter budget costs anonymity.

Flat fronts. The majority of the solved (graph, k) pairs (about 72%) exhibit flat trade-off fronts: achieved anonymity does not change appreciably as ρ is varied from 0.70 to 0.95. This indicates that regardless of the BC budgets, the solver reaches the same anonymity level, especially in low- and medium-density panels, where nearly all graphs reach 100% anonymity at every budget level.

Non-flat fronts. The remaining (graph, k) instances show a genuine trade-off where 100% anonymity may not be reachable, especially in the high-density graphs. These graphs require more modifications to reach high anonymity, and the BC budget actively limits what can be achieved; it shows a slightly downward trend in anonymity achieved with higher betweenness centrality budget, more perceptible between larger ranges (example 0.7-0.95 rather than 0.85-0.9). The betweenness centrality budget is expected to affect the anonymity achieved when a large number of modifications are required, as it limits modifications to edges between highly central nodes, thereby forcing the solver to delete more edges between peripheral nodes.

Starting anonymity. The diamond markers reveal that graphs in the medium-density panel start with very different anonymity levels-some below 50%-yet all reach near 100% after solving at any budget. However, for dense graphs, we can more clearly see the effect of starting anonymity on the ability of the solver to reach higher anonymity targets within the time and budget limits.

8.1.3. Q3. Comparing Maximize Anonymity vs. Minimize Modifications mode for (Degree, Triangle) Anonymity

Fig. 8.3 and Fig. 8.4 plot achieved anonymity against measured top-10% BC ranking preservation (RBO) for both modes, segregated by graph density. Each point is one (graph, parameter) instance. The green region in the top-right corner marks the ideal of high anonymity and high preservation simultaneously.

We plot RBO values (0-1) on the x-axis rather than the BC budget ρ because ρ is a proxy constraint for intended budget and not a direct measure of what centrality preservation was actually achieved; plotting measured RBO gives a better picture of the utility preservation-anonymity trade-off in practice.

ρ	Total	T/O	Already	Optimal	Sub-opt	BC Sat.	Mean Anon.
0.70	145	22	98	18	7	122	94.0%
0.80	145	19	98	23	5	121	93.9%
0.85	145	20	98	22	5	121	93.9%
0.90	145	19	98	23	5	119	93.9%
0.95	145	24	98	18	5	115	93.6%

Table 8.6: Results for 3-deg+BC ($k = 3$) by BC preservation level ρ. **T/O** = Timed out before finding any feasible solution. **Already** = satisfied before solving. **Optimal** = proven optimal; solver terminated before the time limit. **Sub-opt** = feasible solution returned at the time limit, optimality unproven. **BC Sat.** = instances where the BC preservation constraint was satisfied. Mean Anon. is the mean achieved anonymity over all solved instances.

ρ	Total	T/O	Already	Proven	Sub-opt	BC Sat.	Mean Anon.
0.70	27	1	2	21	3	16	99.4%
0.80	27	1	2	21	3	18	99.2%
0.85	27	0	2	19	6	18	97.1%
0.90	27	1	2	19	5	17	98.8%
0.95	27	0	2	19	6	19	96.5%

Table 8.7: Results for 2-(deg, Tri)+BC ($k = 2$, maximize-anonymity mode) by BC preservation level ρ. **T/O** = Timed out before finding any feasible solution. **Already** = satisfied before solving. **Proven** = certified optimal by Pumpkin's `proven_optimal` flag. **Sub-opt** = feasible solution returned at the time limit, optimality unproven. **BC Sat.** = instances where the BC preservation constraint was satisfied. Mean Anon. is the mean achieved anonymity over all solved instances.

Minimize-modifications mode (k-(deg, Tri), blue circles) produces a median RBO of 1.0 for low and medium-density graphs (i.e., the top-10% ranking is perfectly preserved in the majority of cases) and less so in graphs with high density. Points in this mode cluster along the right edge of each panel: high preservation, varying anonymity depending on which target was requested. Minimizing modifications implicitly preserves centrality well, even without any explicit BC constraint.

Maximize-anonymity mode (BC + k-(deg, Tri), orange squares) achieves a median anonymity of 100% across instances for both $k = 2$ and $k = 3$, but at the cost of reduced ranking preservation: median RBO drops to 0.77 for $k = 2$ and 0.76 for $k = 3$. Points spread leftward across the panels, particularly for medium- and high-density graphs. This trade-off is expected: maximizing anonymity requires more edge deletions, which disrupts shortest-path structure and disturbs the centrality ranking. Because we do not directly constrain the number of modifications, we also allow the solver to find feasible solutions sooner and achieve the fewest timed-out instances among all modes. This also indicates that this mode is ideal for dense graphs requiring a higher number of modifications on average, which may otherwise take a lot longer to find a feasible solution for the anonymity target.

8.1.4. Q4. Effectiveness of BC constraint

Fig. 8.5 and Fig. 8.6 plot the requested BC preservation level ρ on the x-axis against the measured top-10% BC ranking preservation (RBO) actually achieved on y-axis. Each point is the median across all graphs at that ρ level; bars show the interquartile range. The dashed diagonal marks perfect calibration ($y = x$): a scenario when it is exactly able to preserve the intended centrality value.

We use median and IQR rather than mean and standard deviation because the RBO distribution across graphs is skewed for BC + k-(deg, Tri) mode, as some graphs preserve centrality perfectly while others do not, and the median is more robust to these outliers.

BC + k-deg (Fig. 8.5). The BC constraint is highly effective for both $k = 2$ and $k = 3$: the median RBO sits at exactly 1.0 across all five ρ levels, with a near-zero IQR. This means that for the majority of graphs, the top-10% BC ranking is perfectly preserved after k-deg anonymization regardless of the requested budget. The constraint satisfies the BC preservation condition in 617 of 634 solved instances for $k = 2$ (97%) and 598 of 621 for $k = 3$ (96%). There could be the following reasons for this: a. k-deg anonymization requires very few modifications, which are unlikely to involve central edges regardless of the budget. b. The constraint on the number of modifications to the top 10% central nodes is highly sensitive to how the metric is measured. c. Allowing both addition and deletion operations balances the overall centrality scores, as deletion operations

ρ	Total	T/O	Already	Proven	Sub-opt	BC Sat.	Mean Anon.
0.70	27	1	2	19	5	17	95.5%
0.80	27	0	2	19	6	20	94.7%
0.85	27	1	2	18	6	17	96.6%
0.90	27	1	2	18	6	17	97.1%
0.95	27	0	2	18	7	19	94.4%

Table 8.8: Results for 3-(deg, Tri)+BC ($k = 3$, maximize-anonymity mode) by BC preservation level ρ. **T/O** = Timed out before finding any feasible solution. **Already** = satisfied before solving. **Proven** = certified optimal by Pumpkin's `proven_optimal` flag. **Sub-opt** = feasible solution returned at the time limit, optimality unproven. **BC Sat.** = instances where the BC preservation constraint was satisfied. Mean Anon. is the mean achieved anonymity over all solved instances.

distort centrality rankings more than additions.

BC + k-(deg, Tri) (Fig. 8.6). For this mode of operation, median RBO rises from approximately 0.60 at $\rho = 0.70$ to 0.84 at $\rho = 0.95$, but remains consistently below the diagonal throughout. The IQR is also large, spanning roughly 0.3 at most budget levels indicating high variability across graphs. The BC preservation condition is satisfied in only 178 of 264 solved instances (67%), confirming that the constraint frequently fails to enforce the requested budget. Tightening ρ does produce a measurable improvement in median RBO, which shows the constraint is responsive but only moderately effective in producing the desired level of utility conservation. This is also due to the fact that betweenness centrality is more volatile to deletions and cannot be directly constrained, as each modification changes the centrality values and rankings of local and global structure.

In summary, it is more effective to constrain betweenness centrality by minimizing modifications; however, to be able to achieve high anonymity, especially in denser graphs which start with low anonymity values, maximizing anonymity as an objective can achieve *a solution* without the solver timing out. While maximizing anonymity, it is useful to constrain modifications by tightening the betweenness centrality budget.

8.2 Limitations

To view our research objectively, we assess it critically and provide an overview of what can be improved in the current process:

Solver Configurations. While we've implemented several constraints and experimented with different combinations, solver objectives, solver modes, allowed operations, etc., much remains to be explored even within the current framework. Implementing a maximization of anonymity objective on k-deg measure (where additions and deletions are both allowed) and testing all the configurations exclusively on small-world networks, which are close to impossible to anonymize, would have given us more insights and better confidence in the direction of future possibilities.

Benchmark scope. All 30 benchmark graphs are relatively small (up to 2,113 nodes). On larger networks, the model size grows rapidly due to the $O(n^2)$ indicator variables, and solving times would increase significantly. The degree-based pair pruning (Section 6.3.1) reduces the model size but does not eliminate the quadratic scaling.

Single centrality measure. The utility constraint is built around penalizing modifications to edges incident to central nodes, and we apply this as a proxy for betweenness centrality. Other centrality measures, such as degree centrality, closeness, and eigenvector centrality, are not explored in this thesis, and it cannot be confidently stated if the proxy constraint is most suitable for preserving betweenness centrality or why, without also measuring the effect on adjacent centrality measures.

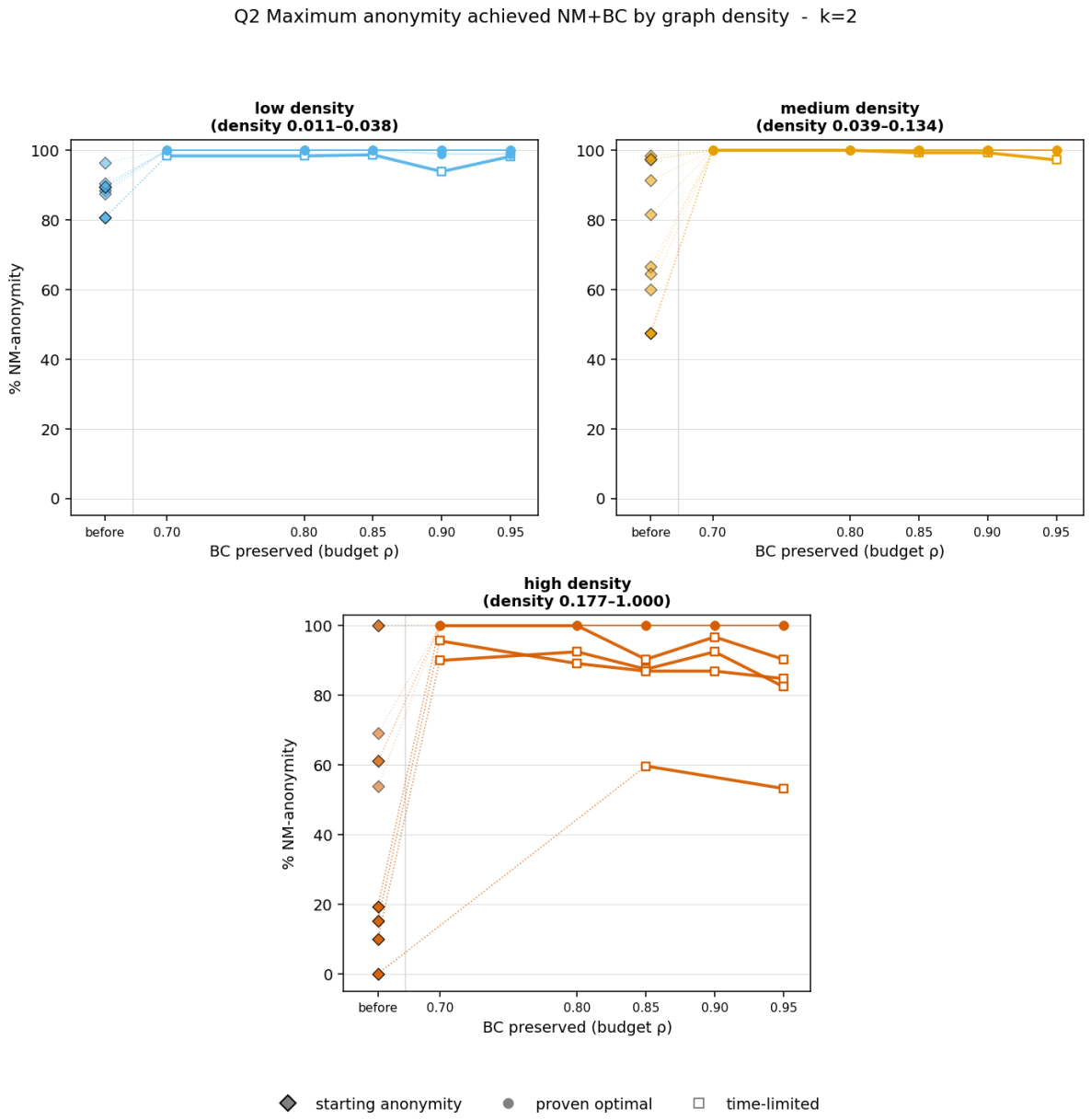

Figure 8.1: Trade-off between BC preservation budget ρ and NM-anonymity achieved for k-(deg,tri)+BC ($k = 2$), grouped by graph density. Each line traces one graph across budget levels $\rho \in \{0.70, 0.80, 0.85, 0.90, 0.95\}$. Diamond markers show starting anonymity before solving.

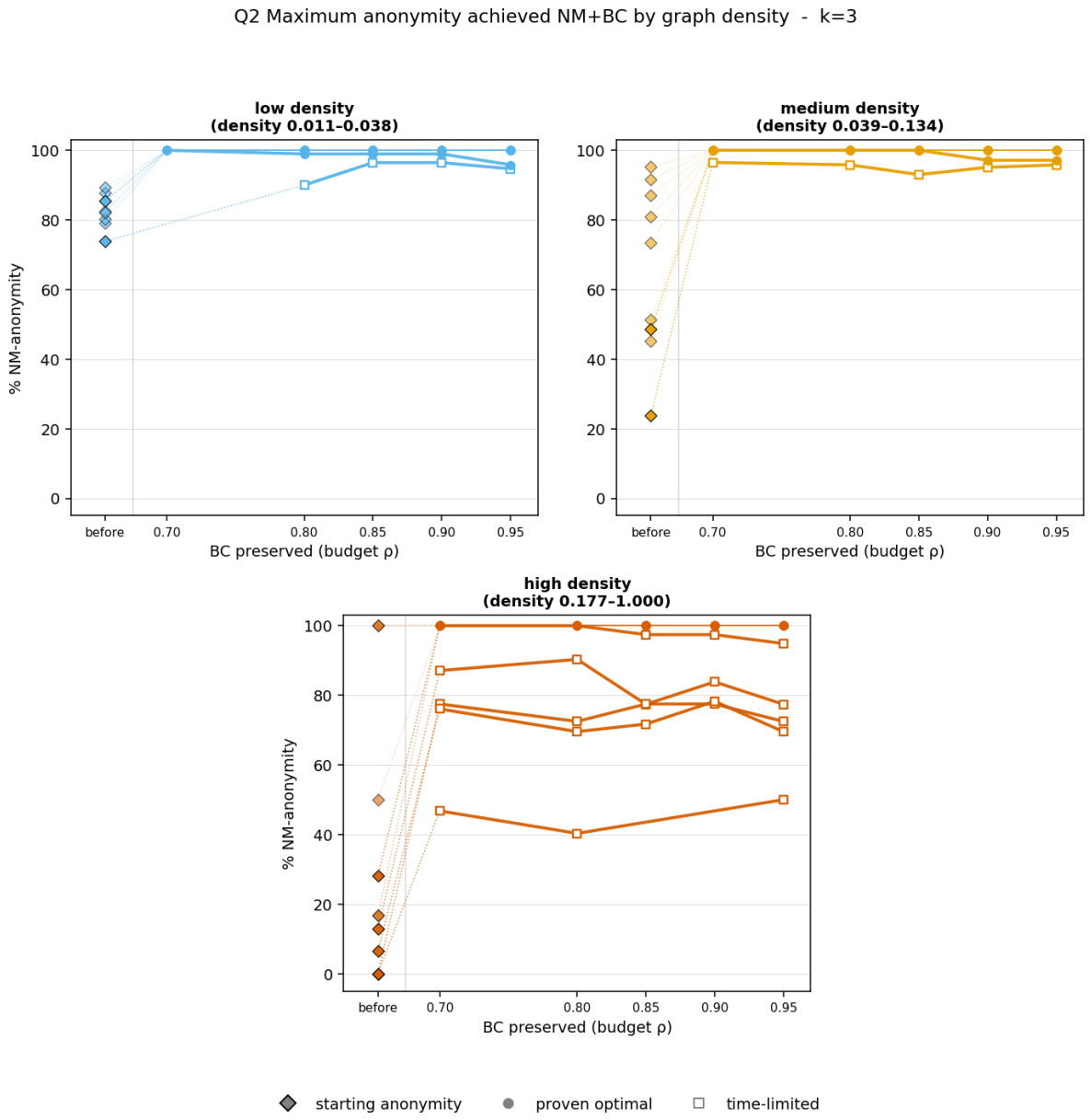

Figure 8.2: Trade-off between BC preservation budget ρ and NM-anonymity achieved for k-(deg,tri)+BC ($k = 3$), grouped by graph density. Each line traces one graph across budget levels $\rho \in \{0.70, 0.80, 0.85, 0.90, 0.95\}$. Diamond markers show starting anonymity before solving.

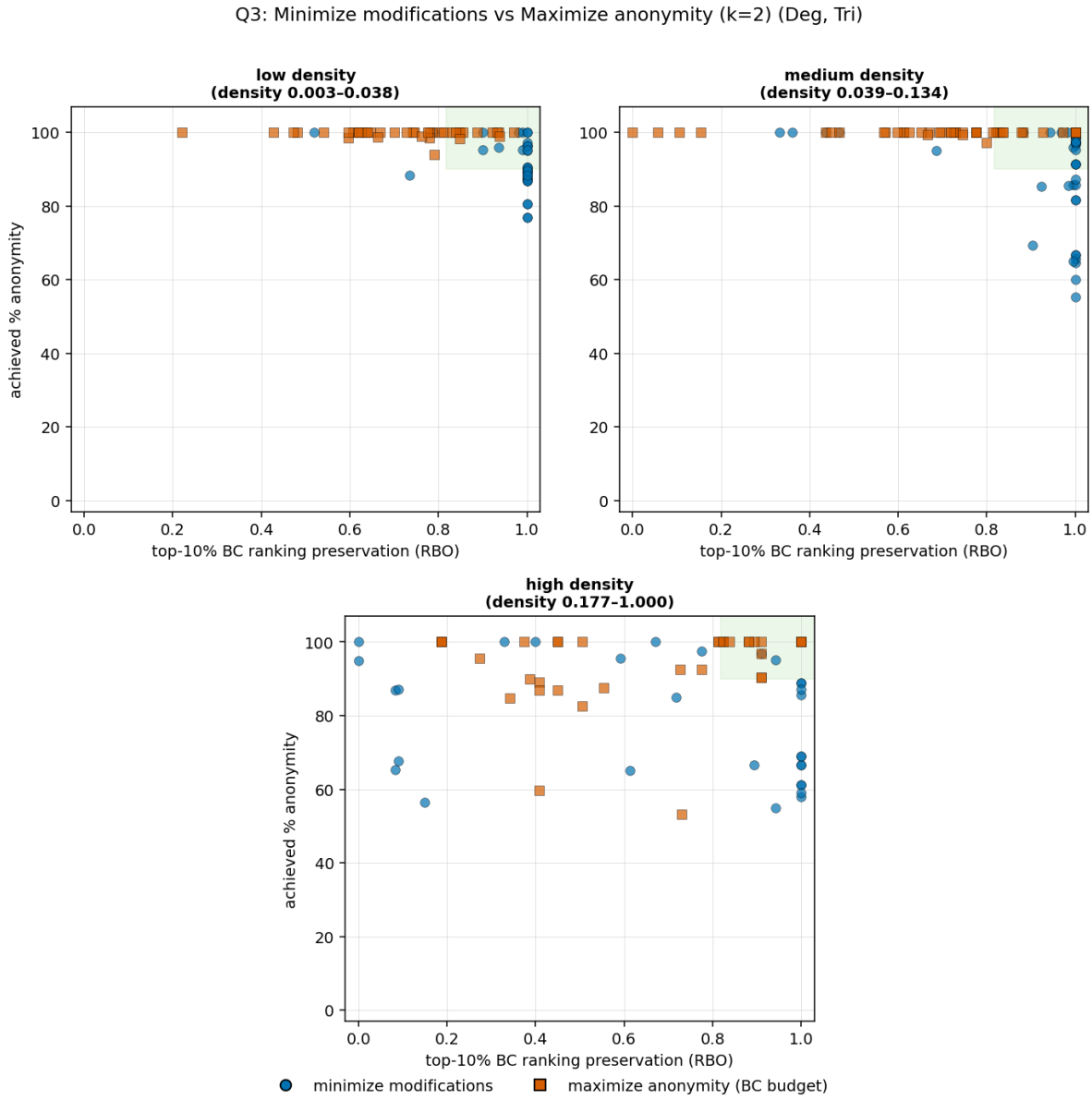

Figure 8.3: Achieved k-(deg, Tri)-anonymity versus top-10% BC ranking preservation (RBO) for minimize-modifications (k-(deg, Tri), blue circles) and maximize-anonymity (k-(deg, Tri)+BC, orange squares) modes, $k = 2$, grouped by graph density. Each point is one (graph, parameter) instance. The green region marks the ideal of high anonymity and preserved ranking.

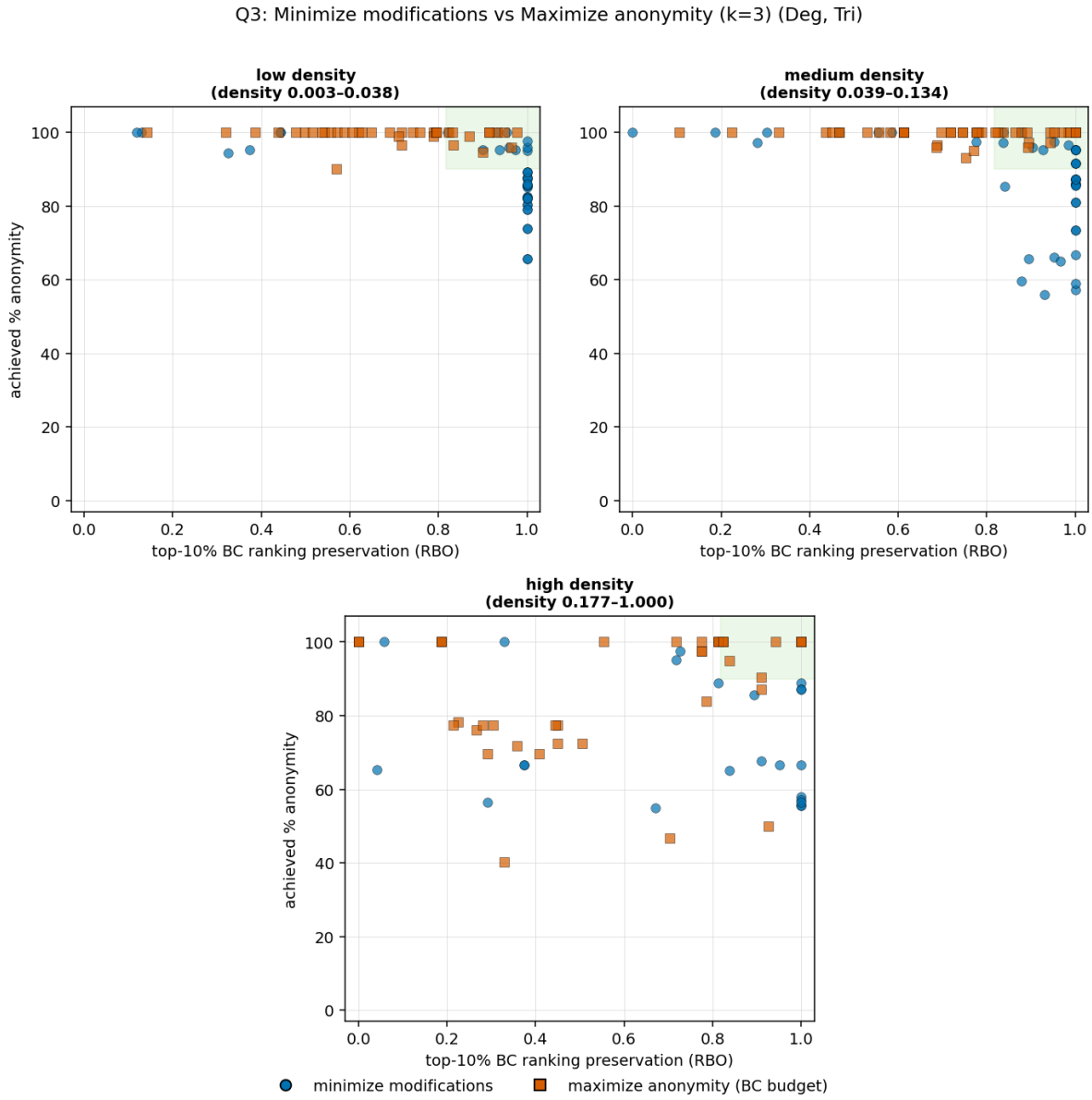

Figure 8.4: Achieved k-(deg, Tri)-anonymity versus top-10% BC ranking preservation (RBO) for minimize-modifications (k-(deg, Tri), blue circles) and maximize-anonymity (k-(deg, Tri)+BC, orange squares) modes, $k = 3$, grouped by graph density. Each point is one (graph, parameter) instance. The green region marks the ideal of high anonymity and preserved ranking.

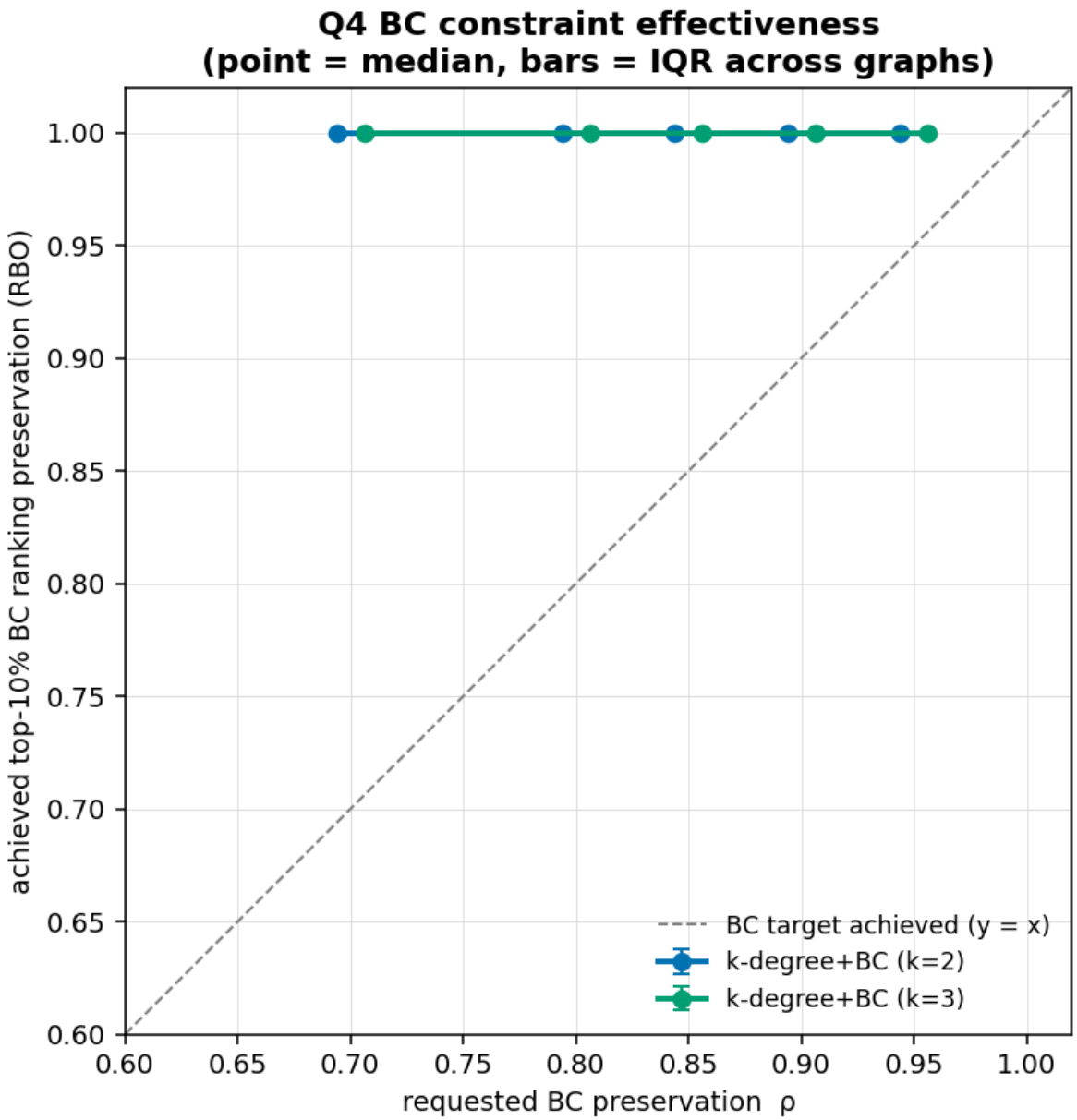

Figure 8.5: BC constraint effectiveness for k-deg+BC modes ($k \in \{2, 3\}$): requested BC preservation budget ρ vs. achieved top-10% BC ranking preservation (RBO), measured as the median across graphs with interquartile range. The dashed diagonal indicates perfect preservation of target BC ($y = x$).

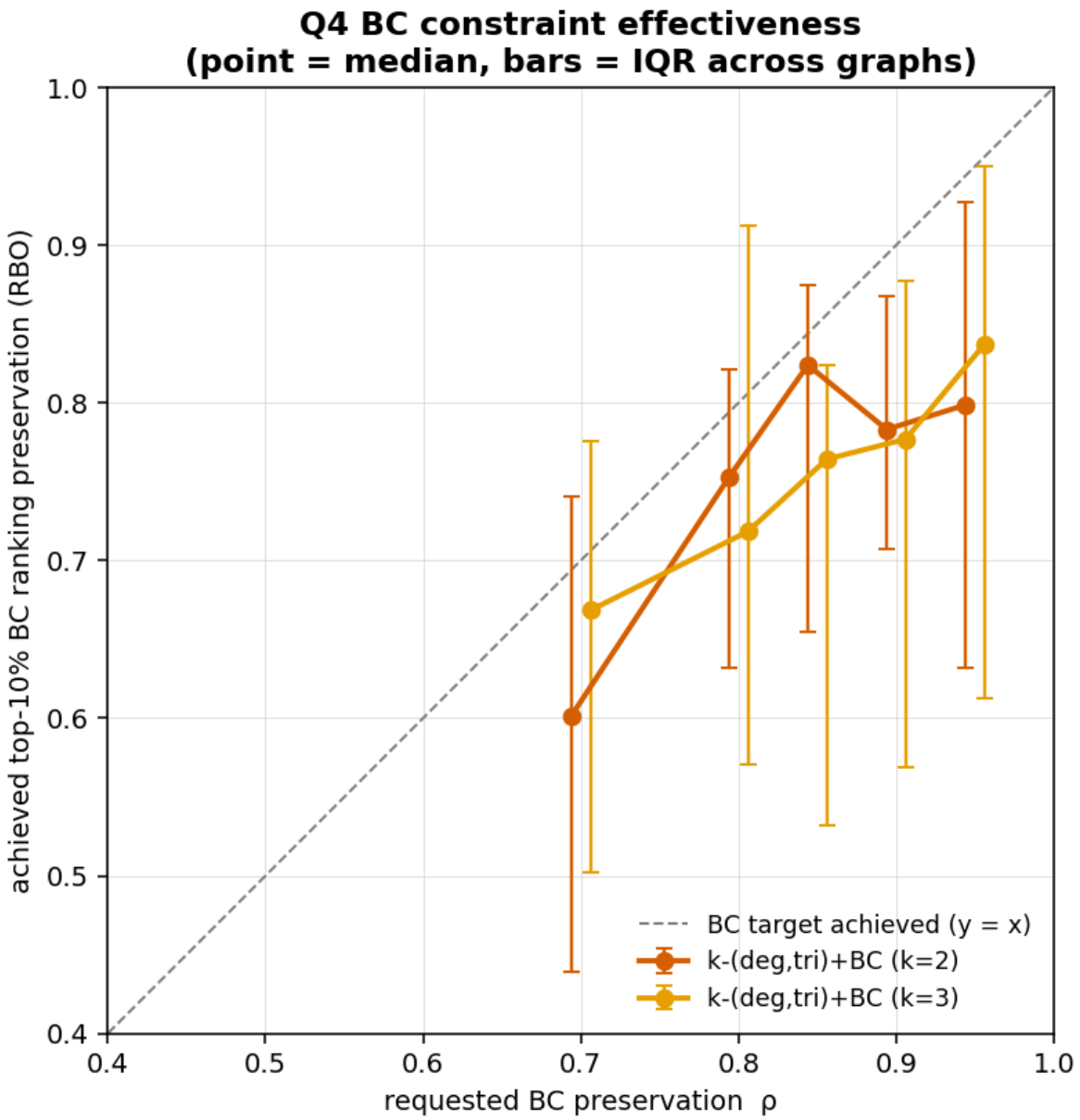

Figure 8.6: BC constraint effectiveness for k-(deg,tri)+BC modes ($k \in \{2, 3\}$): requested BC preservation budget ρ vs. achieved top-10% BC ranking preservation (RBO), measured as the median across graphs with interquartile range. The dashed diagonal indicates preservation of target BC ($y = x$).

9

Conclusion

Publishing social network data enables research in multiple disciplines, such as epidemiology, finance, transport, social sciences, etc., but it also exposes individuals to re-identification attacks that exploit structural properties and connections in the network. Existing approaches to graph anonymization typically treat privacy and utility as separate concerns: they modify the graph to satisfy an anonymity criterion and then measure the damage to network properties after the fact, with no mechanism to control the links changed during the anonymization process itself.

In this thesis, we addressed this gap by formulating graph anonymization as a constraint optimization problem in which utility preservation is modeled as a constraint for anonymization. We modeled two structural anonymity measures k-deg and k-(deg, Tri) and introduced a betweenness-centrality preservation constraint that penalizes modifications to edges between structurally important nodes. The framework supports two solving modes: minimizing the number of edge modifications for a fixed anonymity target and maximizing the number of anonymous nodes under a fixed centrality budget. All configurations are solved by a single call to the Pumpkin constraint solver's linear SAT-UNSAT procedure, which provides optimality certificates when the search completes within the time limit.

We evaluated the framework on 30 real-world networks across four configurations. Our results show that both anonymity measures are practically solvable at the network sizes we tested, with the value of k, starting anonymity, strictness of the anonymity, and density of the graph impacting the feasibility. The two solving modes offer different trade-offs: minimizing modifications implicitly preserves centrality well but does not guarantee high anonymity, while maximizing anonymity under a BC budget achieves high anonymity at the cost of reduced centrality preservation. The betweenness centrality constraint achieves satisfactory results 67% of the time when maximizing anonymity is prioritized. Tighter budgets produce better preservation of utility, confirming that the constraint is responsive and that actively incorporating utility into the anonymization model is a promising direction.

To meaningfully utilize and expand on the work of this thesis, some more prospects could be considered in the future:

Scalability: Applying the framework to larger networks would require a more aggressive model reduction, for example, using the tighter per-group degree-gap pruning discussed in Section 6.3.1, or implementing propagators for anonymization.

Centrality measures: As mentioned previously Section 8.2, our centrality constraint can be extended for other centrality measures apart from betweenness centrality. A good place to start would be datasets from [21] which provide a comprehensive overview of the effect of manipulations on network centrality measures.

Enhancing sandbox environment: The constraint programming approach adopted in this work sets up the groundwork to make anonymization frameworks accessible to scientists of different backgrounds. This can further be expanded by adding more anonymity and utility constraints, starting with stricter or hybrid variants of k-anonymity. We can further solicit input from researchers involved in network studies to understand the privacy requirements for the user data and the properties that are commonly distorted in anonymization. Involving social scientists in this process adds domain knowledge of the properties of the network that need to be preserved or anonymized.

Bibliography

[1] E. D. Arsene, R. G. de Jong, F. W. Takes, and A. L. D. Latour. Simulated annealing for network anonymization, 2025. Unpublished manuscript.

[2] Asma Azizi, Cesar Montalvo, Baltazar Espinoza, Yun Kang, and Carlos Castillo-Chavez. Epidemics on networks: Reducing disease transmission using health emergency declarations and peer communication. *Infectious Disease Modelling*, 5:12–22, 2020.

[3] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th International Conference on World Wide Web*, page 181–190. Association for Computing Machinery, 2007.

[4] Samuel Bonello, Rachel G. de Jong, Thomas H. W. Bäck, and Frank W. Takes. Utility-aware social network anonymization using genetic algorithms. *arXiv preprint*, 2025.

[5] 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*, page 671–675, New York, NY, USA, 2013. Association for Computing Machinery.

[6] 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*, page 459–470, 2010.

[7] Rachel G. de Jong, Mark P. J. van der Loo, and Frank W. Takes. Algorithms for efficiently computing structural anonymity in complex networks. *ACM Journal of Experimental Algorithmics*, 28:1–22, 2023.

[8] Rachel G. de Jong, Mark P. J. van der Loo, and Frank W. Takes. A systematic comparison of measures for publishing k-anonymous social network data, 2025.

[9] Rachel G. de Jong, Mark P. J. van der Loo, and Frank W. Takes. The anonymization problem in social networks, 2026.

[10] Delft High Performance Computing Centre. DHPC Hardware DelftBlue Documentation. `https://doc.dhpc.tudelft.nl/delftblue/DHPC-hardware/`, 2025. Accessed: July 2026.

[11] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[12] François Fages and Sylvain Soliman. Reifying global constraints. Research Report RR-8084, INRIA, 2012.

[13] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[14] Yuanjing Hao, Xuemin Wang, Liang Chang, Long Li, and Mingmeng Zhang. A dynamic social network graph anonymity scheme with community structure protection. *Computers, Materials & Continua*, 82(2):3131–3159, 2025.

[15] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. *VLDB*, 03 2007.

[16] Shouling Ji, Weiqing Li, Prateek Mittal, Xin Hu, and Raheem Beyah. SecGraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization. In *24th USENIX Security Symposium (USENIX Security 15)*, 2015.

[17] Honglu Jiang, Jian Pei, Dongxiao Yu, Jiguo Yu, Bei Gong, and Xiuzhen Cheng. Applications of differential privacy in social network analysis: A survey, 2021.

[18] H. Kaur, N. Hooda, and H. Singh. k-anonymization of social network data using neural network and svm: K-neurosvm. *Journal of Information Security and Applications*, 72:103382, 2023.

[19] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, 2008.

[20] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, page 173–187. IEEE Computer Society, 2009.

[21] Qikai Niu, An Zeng, Ying Fan, and Zengru Di. Robustness of centrality measures against network manipulation. *Physica A: Statistical Mechanics and its Applications*, 438:124–131, 2015.

[22] Rodrigo Marcel Araujo Oliveira, Angelo Marcio Oliveira Sant'Anna, and Paulo Henrique Ferreira. Complex networks-based anomaly detection for financial transactions in anti-money laundering. *Forensic Science International: Digital Investigation*, 55:302005, 2025.

[23] Daniele Romanini, Sune Lehmann, and Mikko Kivelä. Privacy and uniqueness of neighborhoods in social networks, 2020.

[24] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier, 2006.

[25] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[26] P. Samarati. Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, pages 1010–1027, 2001.

[27] Piotr Sapiezynski, Arkadiusz Stopczynski, David Dreyer Lassen, and Sune Lehmann. Interaction data from the Copenhagen Networks Study. *Scientific Data*, 6:315, 2019.

[28] Sergei Solonets, Victor Drobny, Victor Rivera, and JooYoung Lee. Introducing adegree: Anonymisation of social networks through constraint programming. In *Mobility Analytics for Spatio-Temporal and Social Data*, 2018.

[29] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, pages 571–588, 2002.

[30] William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.*, 28(4), 2010.

[31] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*, 2019.

[32] Navid Yazdanjue, Hossein Yazdanjouei, Hassan Gharoun, Mohammad Sadegh Khorshidi, Morteza Rakhshaninejad, and Amir H. Gandomi. A comprehensive bibliometric analysis on social network anonymization: Current approaches and future directions, 2023.

[33] Sen Zhang, Wei Li, and Hao Chen. Utility analysis on privacy-preservation algorithms for online social networks: an empirical study. *Journal of Network Science*, 2021.

[34] 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, 2008.

[35] Lei Zou, Lei Chen, and M. Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. *PVLDB*, 2:946–957, 01 2009.

A

Additional Results

This appendix collects the full per-mode and per-density-tier breakdowns of instance outcomes summarized in Chapter 8. Figures A.1 and A.2 report outcomes by anonymity target and BC budget, and the remaining figures break these down by graph density tier for each mode.

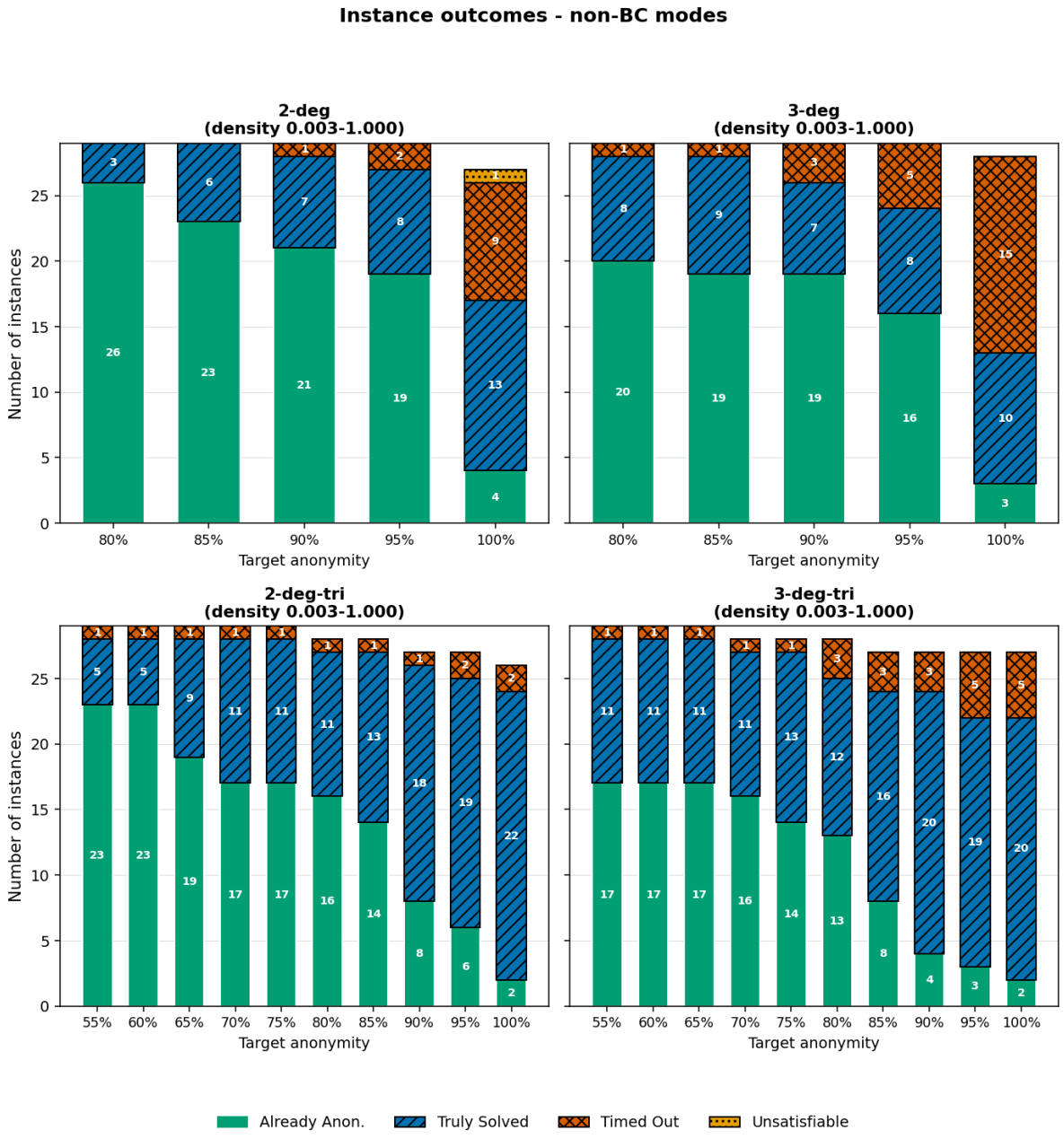

Figure A.1: Visualization of instance outcomes by anonymity target for non-BC modes (2-deg, 3-deg, 2-deg-tri, 3-deg-tri), showing solved, already-anonymous, timed-out, and unsatisfiable instances at each target level.

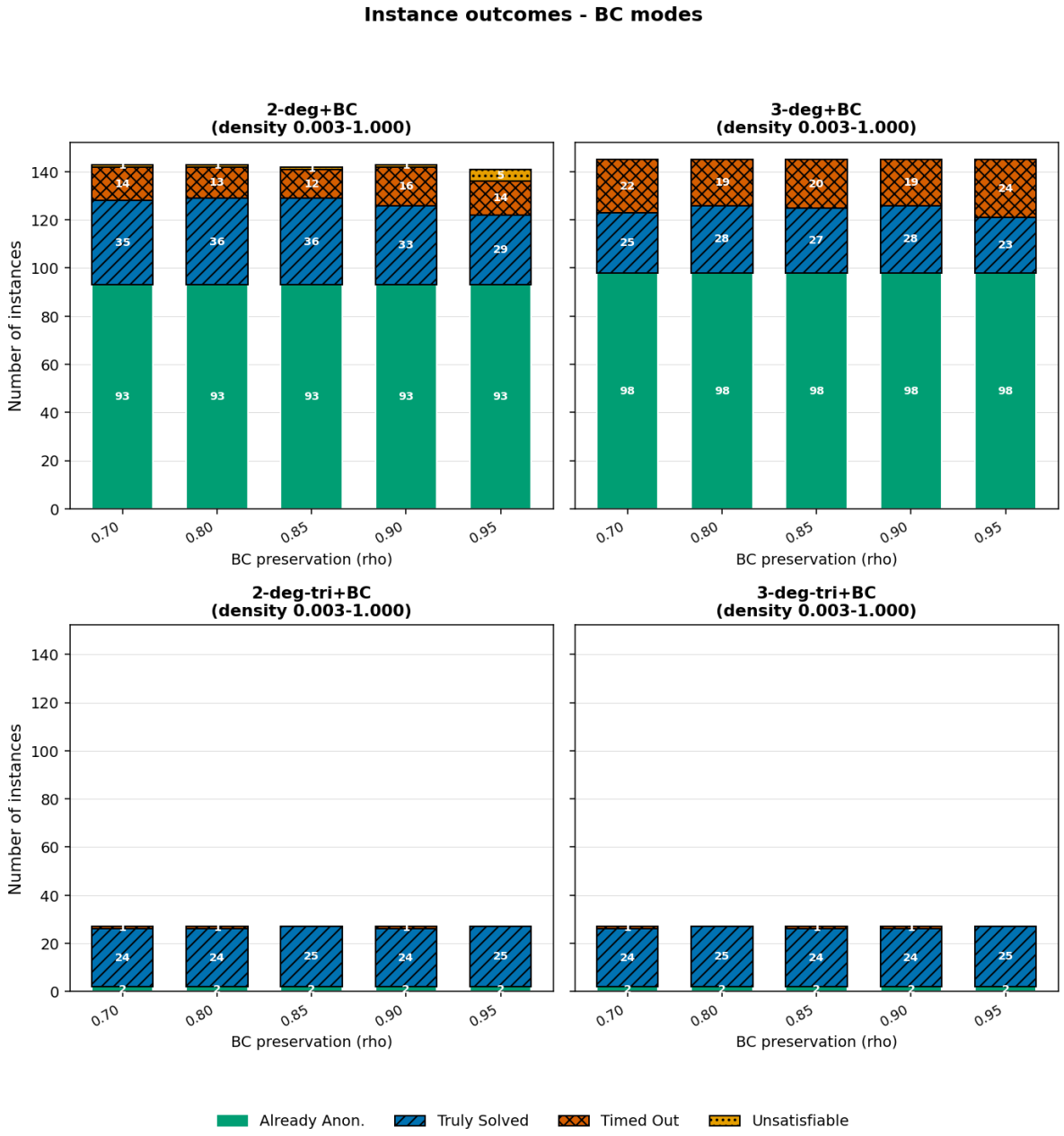

Figure A.2: Visualization of instance outcomes by BC preservation target ρ for BC-constrained modes (2-deg+BC, 3-deg+BC, 2-deg-tri+BC, 3-deg-tri+BC), showing solved, already-anonymous, timed-out, and unsatisfiable instances at each ρ level.

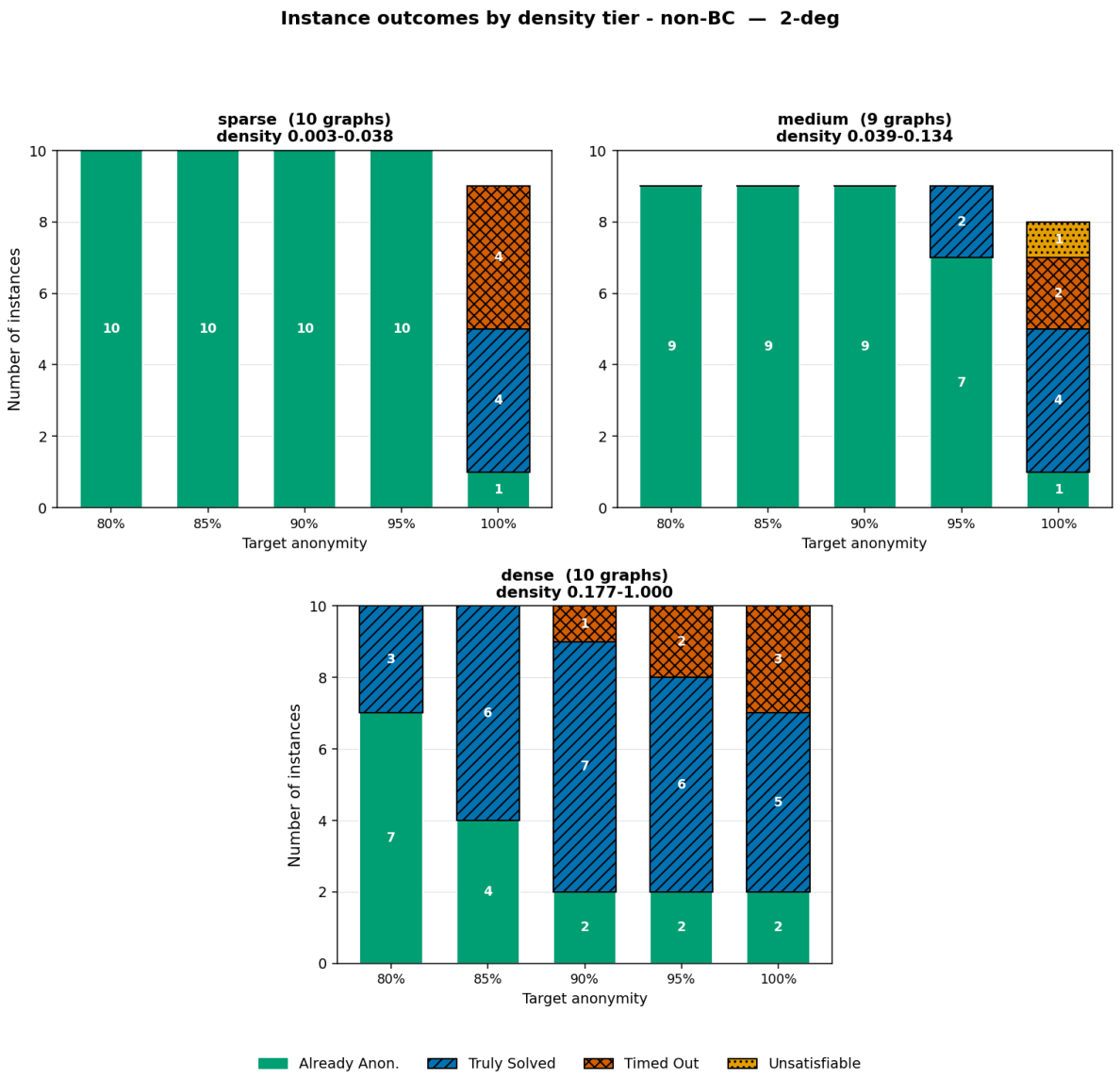

Figure A.3: Instance outcomes by graph density tier for the 2-deg mode, showing already-anonymous, solved, timed-out, and unsatisfiable instances across sparse, medium, and dense graphs at each anonymity target.

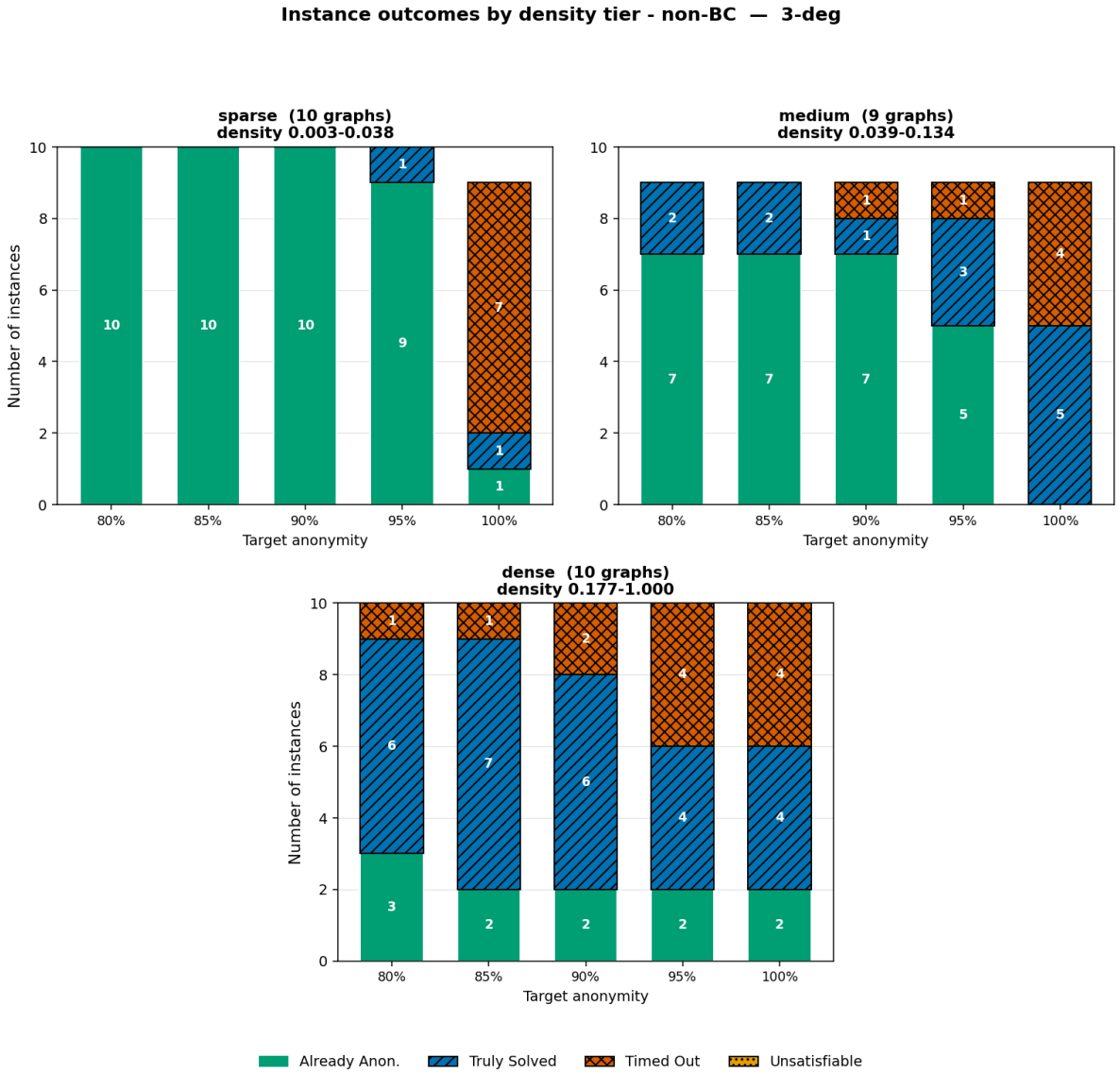

Figure A.4: Instance outcomes by graph density tier for the 3-deg mode.

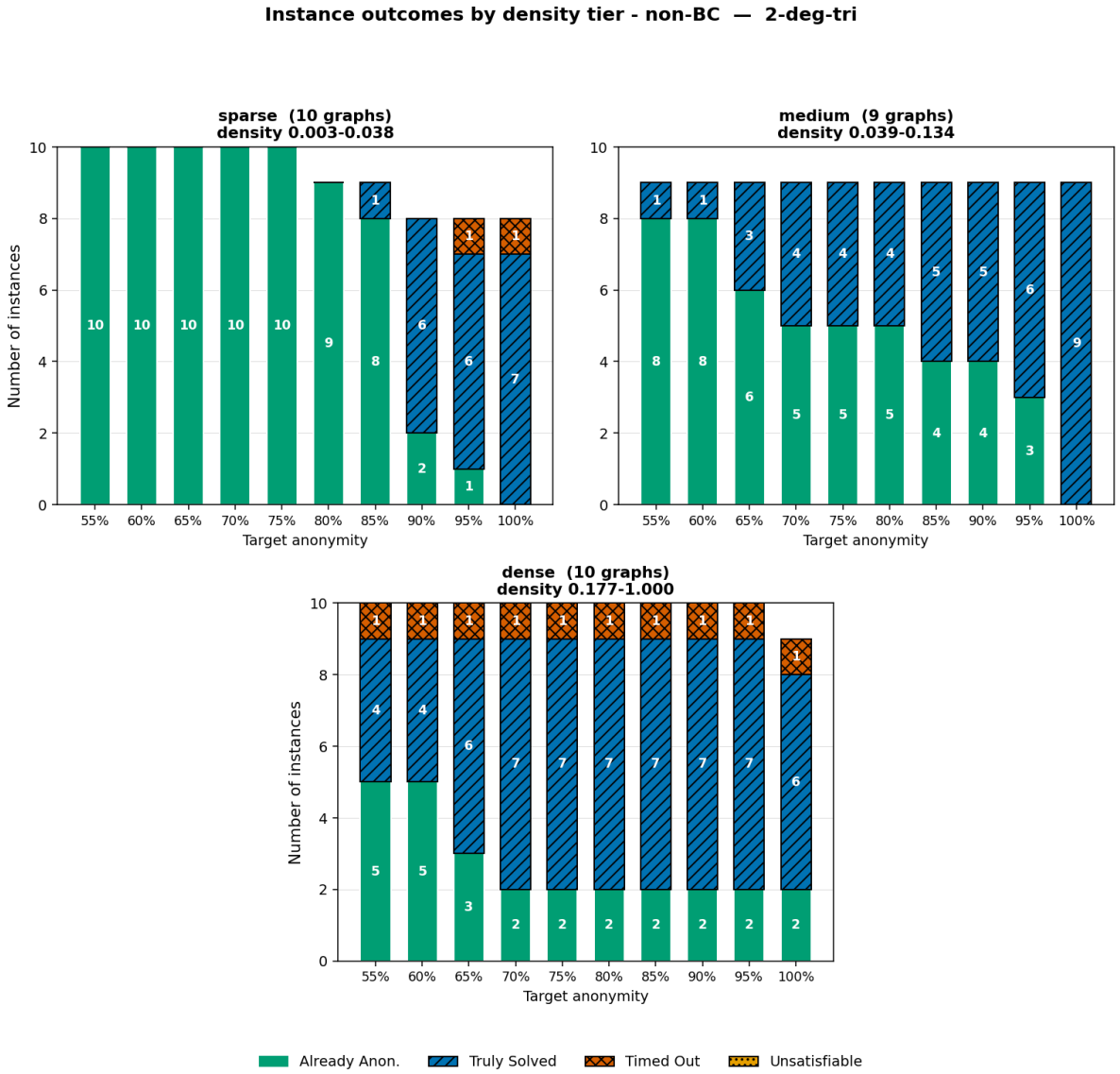

Figure A.5: Instance outcomes by graph density tier for the 2-(deg,tri) mode.

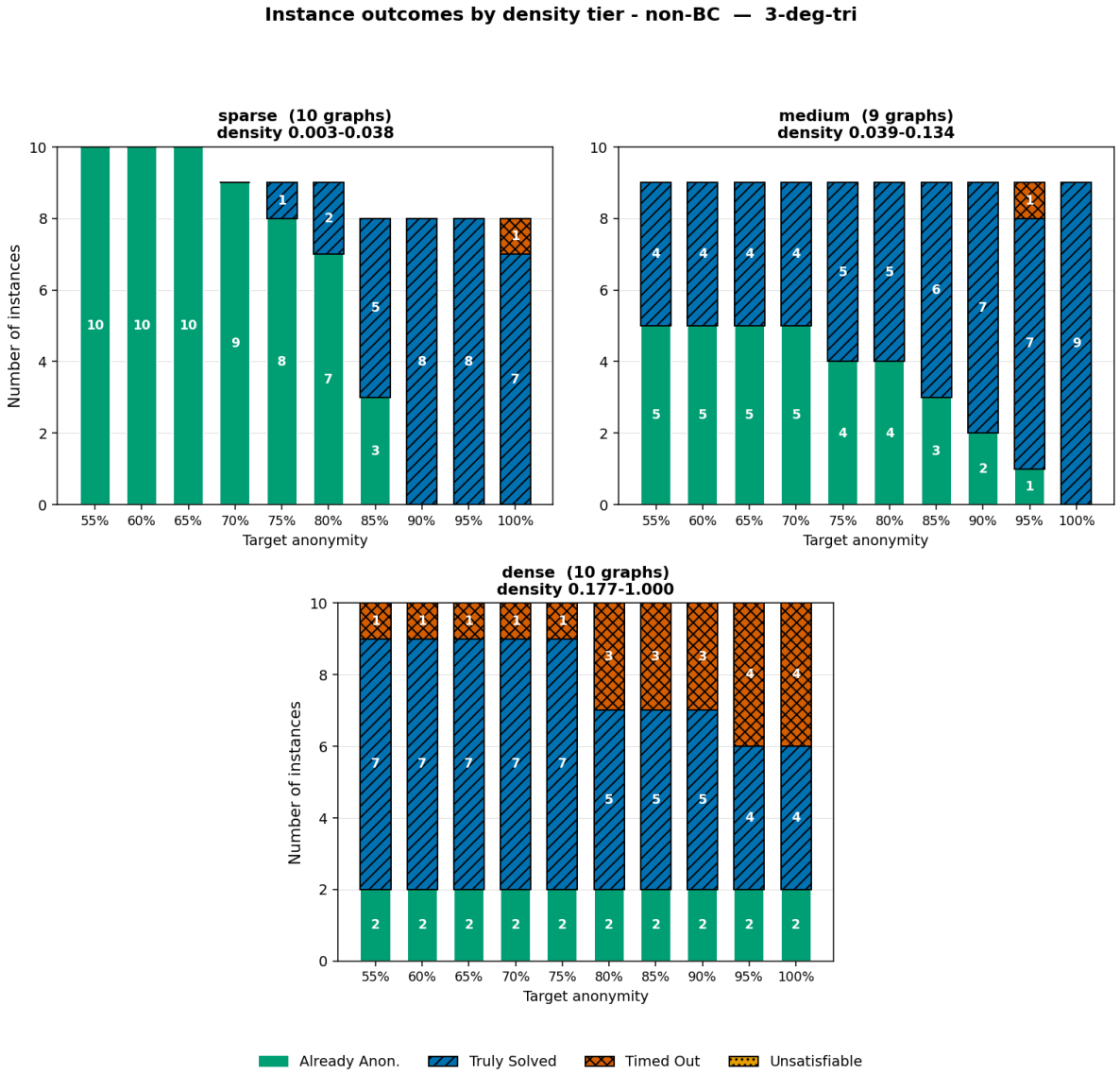

Figure A.6: Instance outcomes by graph density tier for the 3-(deg,tri) mode.

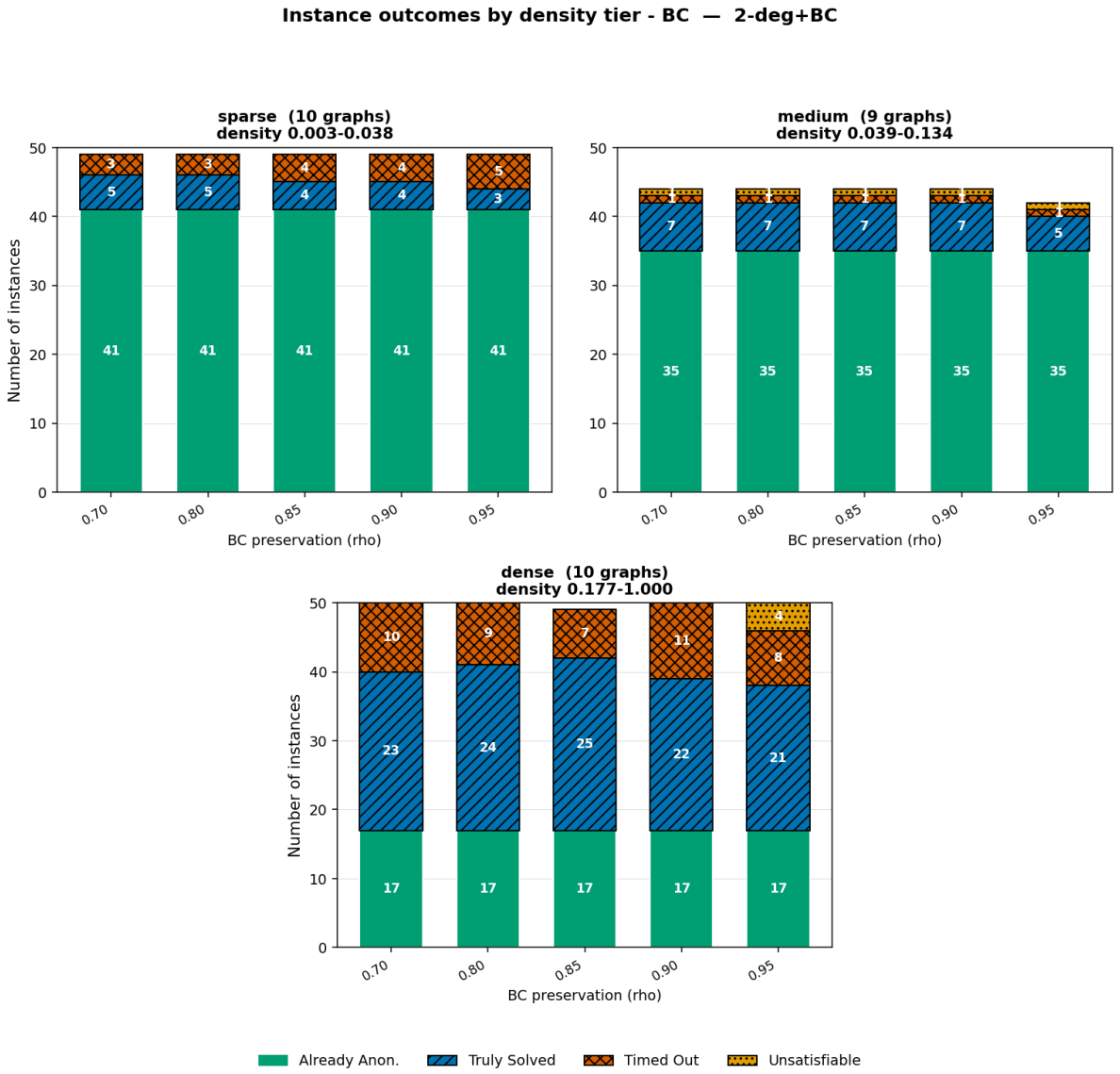

Figure A.7: Instance outcomes by graph density tier for the 2-deg+BC mode, showing already-anonymous, solved, timed-out, and unsatisfiable instances across sparse, medium, and dense graphs at each betweenness-centrality preservation budget ρ.

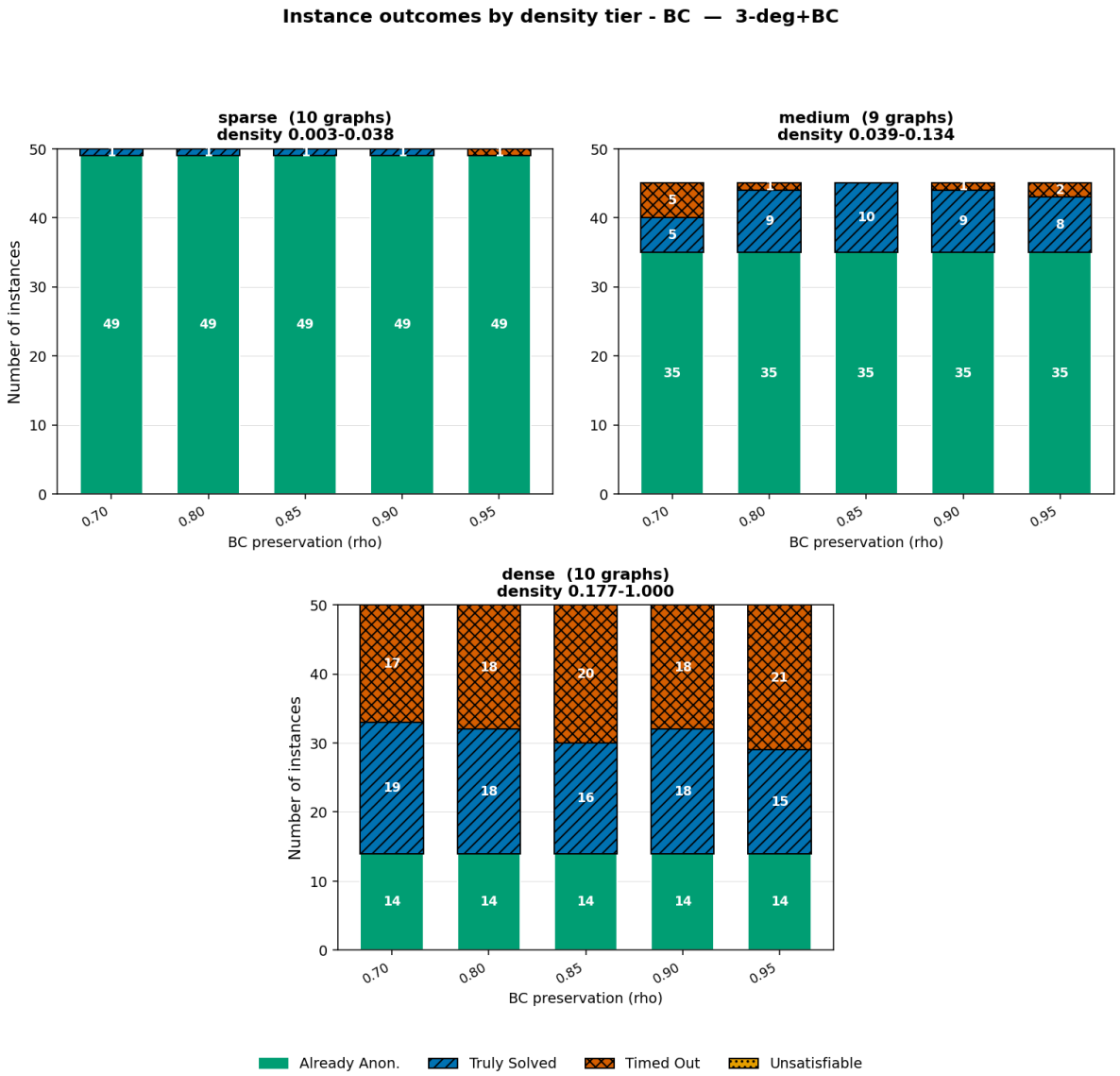

Figure A.8: Instance outcomes by graph density tier for the 3-deg+BC mode.

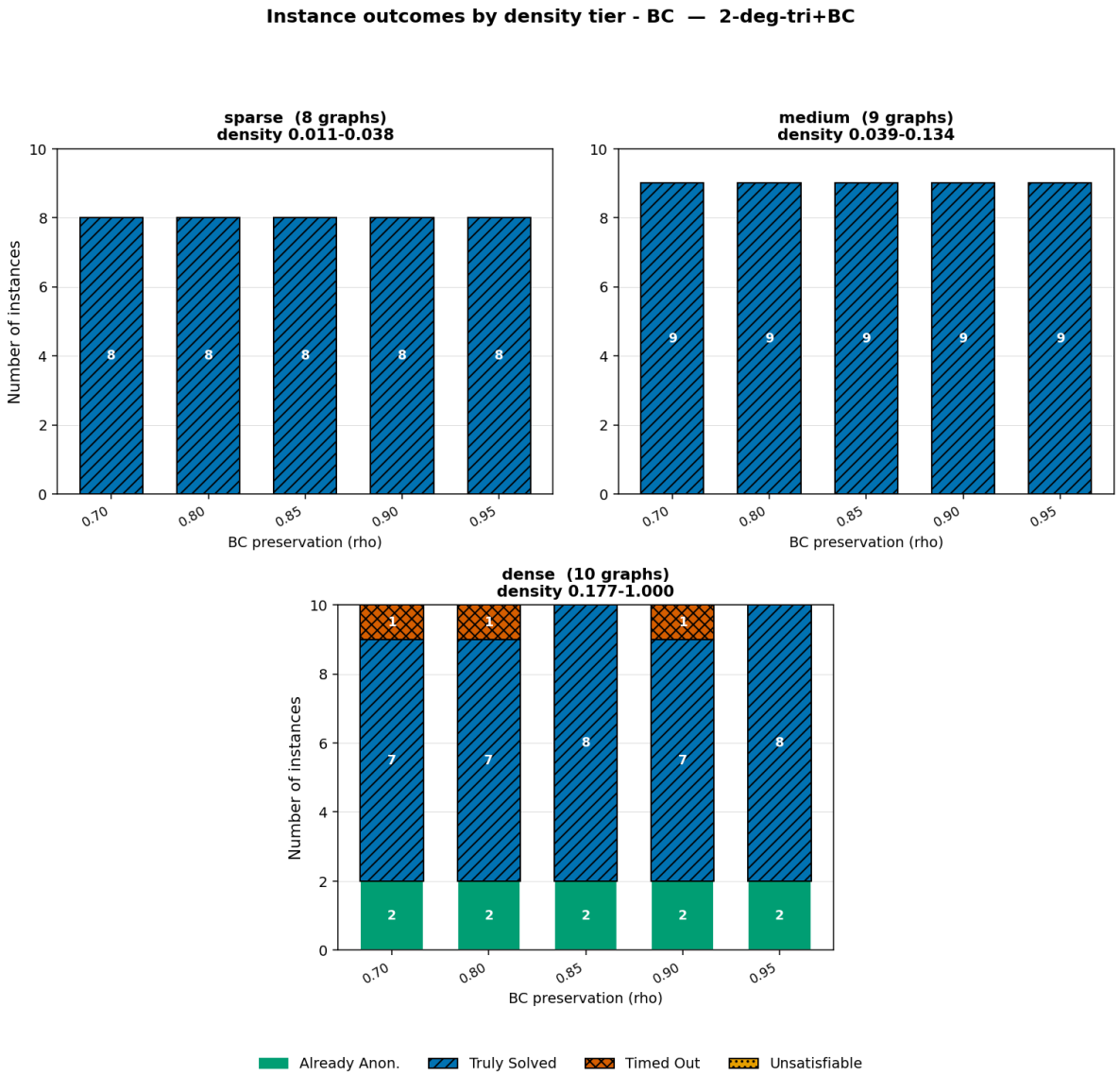

Figure A.9: Instance outcomes by graph density tier for the 2-(deg,tri)+BC mode.

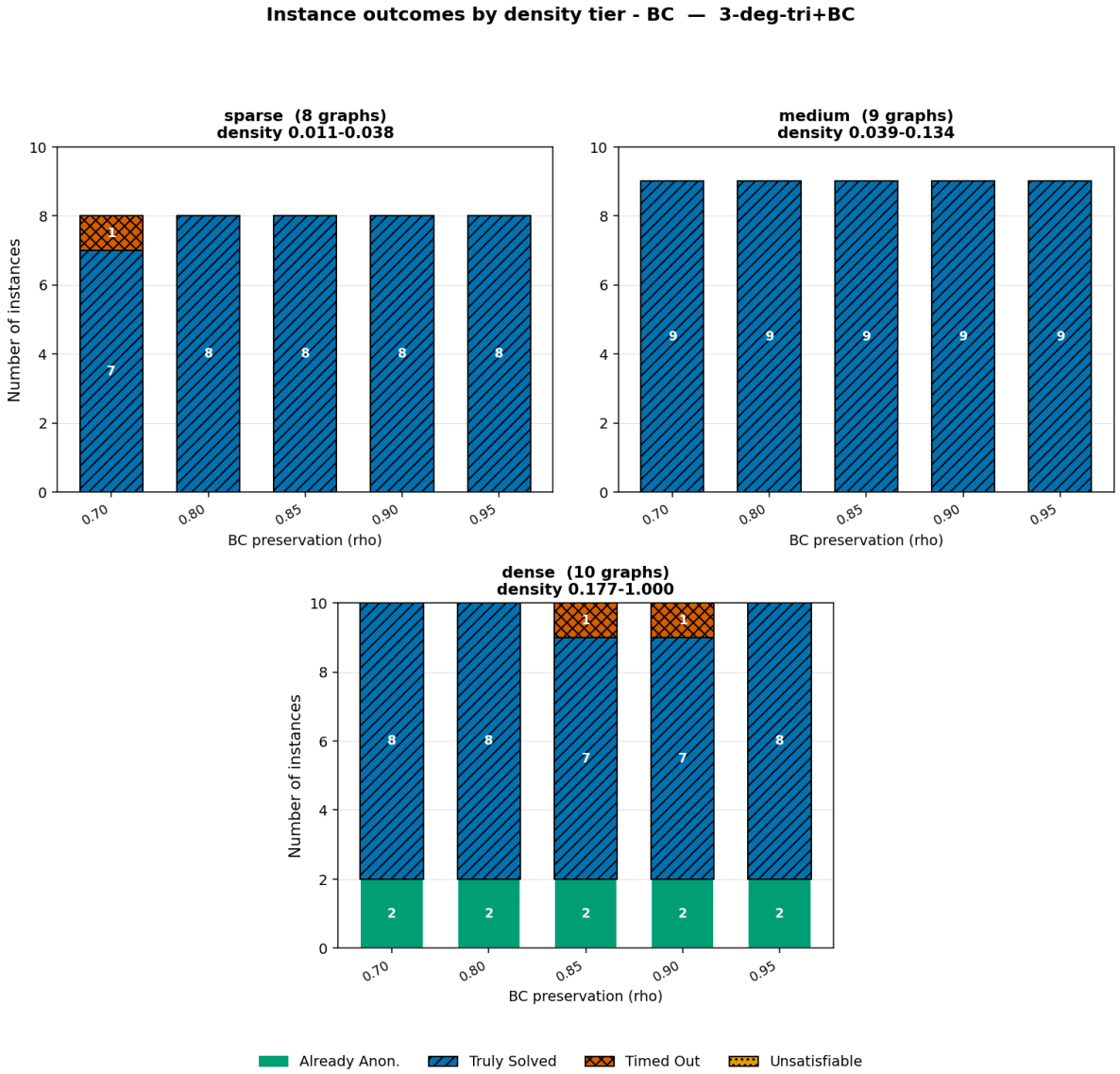

Figure A.10: Instance outcomes by graph density tier for the 3-(deg,tri)+BC mode.

B

Use of Generative AI

I have used generative AI tools in the process of working on my master's thesis within the constraints of TU Delft's guidelines on responsible use of AI. This appendix discloses the scope and use of generative AI tools in this thesis. The tool used was Claude (https://claude.ai). I used Claude Sonnet 5 and Opus 4.8 models in a supporting role to brainstorm and refine my ideas about the project. I crafted the research, design, implementation, analysis, and conclusions myself, and reviewed, verified, and edited all AI-assisted output. I retain full responsibility for the content of this thesis.

B.1 Literature search support.

Apart from finding papers via related links in base papers, the Google Scholar database and specifically searching for known authors, I occasionally used generative AI to search for new papers on a particular topic. For instance, a prompt used would be similar to the following: "Can you help search for papers that used a constraint programming/propagation approach for social network data anonymization, similar to the (attached) ADegree paper?" This was done to ensure that, apart from already covered sources, no new discoverable sources were left out. I independently retrieved, read, and verified every paper found this way and used them in relevant places as citations.

B.2 Coding and data analysis support.

I used generative AI as a coding aid for supporting tasks such as writing boilerplate code, debugging, and improving the readability of my code, as well as for scripts for processing and visualizing results. It did not replace the core contribution of this thesis: I designed and implemented the constraint model for anonymity and utility constraints myself through multiple iterations and experiments. I understood, tested, and verified all code and every result included in this work. AI assistance was used in debugging, library support, code refactoring, refining plotting scripts, adjusting figure layouts, and ensuring consistency between analysis scripts. Example prompts included:
Can you help create a Slurm array job which utilizes the following parameters, file names and partition information?
I want to make this a square plot so that the ideal values bisect the axes. What changes would be required?
Explain the normalization process in the betweenness centrality method of the NetworkX library, with examples?

B.3 Writing.

I used generative AI to improve the clarity and readability of my own drafts and as a way to rephrase the content I had written myself. For instance: *Can you rephrase this paragraph to be less verbose while maintaining the tone of the message with minimal changes to the content?* followed by iterative refinements to the length and content of the text thus generated. I also used it to tighten individual passages, for instance, asking how to

restructure the related-work section so that each paragraph leads with its main idea. I subsequently reviewed and revised all resulting text. I did not use it to generate research ideas, arguments, or interpretations.

B.4 Formatting.

I used generative AI to format tables and results and to debug issues with LaTeX formatting, for instance:
Can you generate boilerplate LaTeX text for inserting a table (or figure) to float at the top of a page with 3 columns and a grid layout?
How to resolve: cref reference format for label type 'algocf' undefined.
How do I resolve: The command "openbox" is already defined.

Note. In compliance with the data-privacy requirements of the guidelines, I did not enter any sensitive, confidential, or personal data into a generative AI tool; all datasets used in this work are publicly available benchmark networks.