Private Synthetic Graph Generation and Fused Gromov-Wasserstein Distance

Leoni Carla Wirth
Department of Statistics
University of Oxford, UK

Gholamali Aminian
The Alan Turing Institute
London, UK

Gesine Reinert
Department of Statistics
University of Oxford
and The Alan Turing Institute, UK

Abstract

Networks are popular representations of complex data. In particular, differentially private synthetic networks are much in demand. Here, instead of starting from a network, we start with the complex data set itself and construct both a network representation and a corresponding synthetic network generator. We build a network model directly based on the underlying complex system data, capturing its structure and attributes. Using a random connection model, we devise an effective algorithmic approach for generating attributed synthetic networks which is ϵ-differentially private at the vertex level, while preserving utility. We provide theoretical guarantees for the accuracy of the private synthetic networks using the fused Gromov-Wasserstein distance.

1 INTRODUCTION

For sharing data which contain sensitive features, which is often the case for example in financial transaction or in health-related data, a recommended procedure is to create a derived synthetic dataset which is semantically and statistically similar to the real data while protecting privacy. A notion of privacy which is often used in this context is *differential privacy* from Dwork (2006); it is a property of a randomized data generating mechanism (not of a data set) and, at high level, ensures that when two input data sets differ slightly, then the distributions of the synthetically generated data based on each of the two input data sets are also close.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

To achieve differential privacy, many strategies have been suggested, but obtaining theoretical guarantees for these methods remains a challenge, see Zhang et al. (2025) for an overview. In He et al. (2023) a synthetic data generation algorithm called PSMM is developed for generating differentially private synthetic data in a bounded metric space by perturbing the feature space and adding random noise to obtain a possibly signed measure; then the algorithm finds a probability measure that minimizes the bounded Lipschitz distance to the perturbed measure. In He et al. (2023) it is proven that this algorithm has near-optimal utility guarantees.

Often shared data are further processed, perhaps by constructing a related object such as a network (or graph; we use the notions interchangeably). Networks are popular representations of complex data. Synthetic networks are used to detect anomalies and patterns, to assess statistical significance, to augment data when the underlying data set is highly imbalanced, and they are also used for method and algorithm development.

A multitude of synthetic network generators are available, including parametric methods, as in Batagelj and Brandes (2005), empirical approaches such as Reinert and Xu (2024), and deep learning approaches, see for example Qian et al. (2023). However these generating mechanisms usually do not come with privacy guarantees. Here we are particularly interested in a construction mechanisms of synthetic networks for which a differentially privacy guarantee can be supplied.

The input for synthetic network generators is usually taken to be an observed network. However the observed network is itself only a representation of a complex data set. A key novelty of our approach is that we directly take the complex data as input and provide a construction method which is differentially private in the node attributes to obtain synthetic networks which resemble networks obtained from the observed dataset. We then answer two questions:

1. How different is the probability distribution of our

synthetic network model with private data input compared to that of our synthetic network model with the true input data?
2. For a specific data input, how can we jointly construct a network realization obtained from the true data via the network model and a network realization obtained from the differentially private network model such that the two realizations are, in expectation, close?

For achieving differential privacy, we incorporate the mechanism of He et al. (2023) as a central component, augmented with nontrivial steps to produce a privatized network representation rather than privatized samples of the complex data input. On top of this, we extend the mechanism of He et al. (2023) in the following three directions. First, He et al. (2023) use the bounded Lipschitz metric for the synthetic data generation. We replace the bounded Lipschitz metric, which is often not easy to compute, by the easier to calculate total variation distance. We call the resulting algorithm for differentially private synthetic data generation TV-PSMM. Second, in He et al. (2023) the expected 1-Wasserstein distance between the generated distributions of features is employed to measure the utility of the PSMM algorithm. As our outputs are attributed graphs, we use the expected fused Gromov-Wasserstein distance instead, which generalizes the expected 1-Wasserstein distance to (attributed) networks and thus provides a tool to answer the second question. We also introduce a new metric between synthetic generators of networks with vertex attributes that is based on the fused Gromov-Wasserstein metric to answer our first question. The suitability of the fused Gromov-Wasserstein metric as a utility measure is demonstrated by its applications to machine learning problems, see for example Brogat-Motte et al. (2022). Third, while He et al. (2023) employ discrete Laplacian perturbations, we consider more general discrete noise distributions.

We stress that these extensions of the method in He et al. (2023) are not the key contributions of this paper. Instead, the central advance of this work is the transition to network representations. More specifically, our key contributions are

1. constructing differentially private networks jointly with a network presentation of complex data, instead of first generating a network from the data and then creating a synthetic network generator based on it;
2. providing extensive theoretical guarantees both for the network generator and for the actually created networks.

The strengthening of the PSMM method from He et al. (2023) by using total variation distance and general noise distributions is a further contribution.

The paper is structured as follows: Section 2 introduces differential privacy as well as the fused Gromov-Wasserstein and total variation distances, while Section 3 introduces the TV-PSMM algorithm that can be applied to obtain privatized data (which are later used as vertex attributes). We present our synthetic network model and discuss our private synthetic graph generation (PSGG) algorithm which outputs a joint realization of the network models in Section 4. We derive our main theoretical results regarding the effect of the TV-PSMM algorithm and the PSGG algorithm and their rates in Section 5. The conclusion and future works are presented in Section 6. Detailed proofs are deferred to the Appendix, which also contains more related work, and more illustrations as well as a list of notations. The code can be found under `https://github.com/LCWirth/PrivateNetworkGen.git`. Figure 1 gives an overview of the core ideas of the paper.

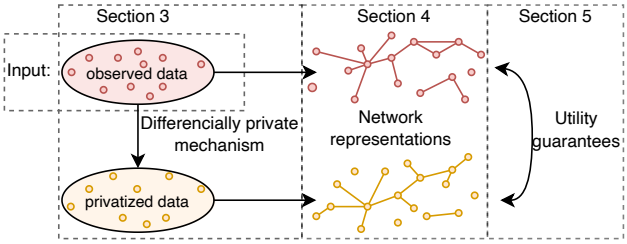

Figure 1: Outline of the paper

2 PRELIMINARIES

Differential Privacy: Inspired by Dwork (2006), we focus on ϵ-differential privacy. A randomized algorithm $\mathcal{M}$ provides *ϵ-differential privacy* for some $\epsilon > 0$ if for any inputs A, A' that differ on only one data point, it satisfies for any measurable set $S \in \text{range}(\mathcal{M})$,

$$\frac{\mathbb{P}(\mathcal{M}(A) \in S)}{\mathbb{P}(\mathcal{M}(A') \in S)} \leq e^{\epsilon}. \tag{1}$$

We introduce a method that starts with complex data that is used as attribute information to generate synthetic networks. In contrast to other approaches, all the information is contained at the vertices itself, not at the edges. Hence, we focus on ensuring differential privacy at vertex level. This also enables us to give privacy guarantees on the edge set, see Remark 4.2. In the context of differential privacy at vertex level, the inputs A, A' in the above definition are given by sets of vertex attributes that differ by a single vertex (attribute).

Total variation distance: The total variation norm of a signed measure η on some measurable space $\mathcal{Z}$ can be defined using the Jordan-Hahn decomposition, which allows expressing η as the difference of two positive measures η^+ and η^-, i.e. $\eta = \eta^+ - \eta^-$; we set

$$\|\eta\|_{TV} := |\eta|(\mathcal{Z}) := \eta^+(\mathcal{Z}) + \eta^-(\mathcal{Z}).$$

In the special case where the signed measure η has a discrete support $\mathcal{Z} = \{z_1, \ldots, z_n\}$, we have for every $Z \subset \mathcal{Z}$ that $\eta(Z) = \sum_{z \in Z} \eta(\{z\})$. This yields the total variation norm $\|\eta\|_{TV} = \sum_{z \in Z} |\eta(\{z\})|$. The total variation norm induces the total variation distance d_{TV} that for two signed measures η_1, η_2 on $\mathcal{Z} = \{z_1, \ldots, z_n\}$ is given by the easy to evaluate formula

$$d_{TV}(\eta_1, \eta_2) := \|\eta_1 - \eta_2\|_{TV} = \sum_{z \in Z} |\eta_1(\{z\}) - \eta_2(\{z\})|. \tag{2}$$

2.1 Fused Gromov-Wasserstein distance

The fused Gromov-Wasserstein (FGW) distance introduced in Vayer et al. (2019, 2020) measures differences between structured objects with features. It combines a Gromov-Wasserstein distance for the structural comparison with a Wasserstein metric for the feature comparison and thus generalizes both metrics.

We focus on the particular discrete case where the structured objects (with features) are given by attributed graphs, see Vayer et al. (2020, Section 4). The underlying idea is the representation of an attributed graph as a measure $\mu = \sum_{i=1}^n h_i \delta_{(x_i, a_i)}$, where $(h_i)_{i \in [n]} \subset \mathbb{R}_+$ such that $\sum_{i=1}^n h_i = 1$. Here the $a_i \in \Omega$ encode the vertex attribute information while the information on the graph structure is captured by the x_i, which are assumed to lie in a complete metric space (X, d_X) that represents the graph structure. For two structured objects $\mu = \sum_{i=1}^n h_i \delta_{(x_i, a_i)}$ and $\eta = \sum_{j=1}^m g_j \delta_{(y_j, b_j)}$, with $y_j \in Y$, and (Y, d_Y) a complete metric space, the fused Gromov-Wasserstein (FGW) distance (or metric) with parameters $p = q = 1$ and $\alpha \in [0, 1]$ is defined by

$$d_{FGW}(\mu, \eta) = d_{FGW,\alpha}(\mu, \eta) = \min_{\pi \in \Pi(\mu, \eta)} E_{\alpha,1,1}(\pi), \tag{3}$$

where with $\pi_{i,j} = \pi((x_i, a_i), (y_j, b_j))$,

$$E_{\alpha,1,1}(\pi) := \sum_{i,j,k,l} \big((1-\alpha) d(a_i, b_j) + \alpha |d_X(x_i, x_k) - d_Y(y_j, y_l)|\big) \pi_{k,l} \pi_{i,j}$$

and $\Pi(\mu, \eta) := \{\pi \in \mathbb{R}_+^{n \times m} \mid \sum_i \pi_{ij} = b_j; \sum_j \pi_{ij} = a_i\}$ is the set of *couplings* between μ and η; a coupling between two probability measures is a construction of the measures on a common probability space. Further details on the fused Gromov-Wasserstein metric can be found in Appendix B.1.

The FGW distance can also be used to measure the distance between random attributed graph distributions and thus between synthetic network models with private data input and true input data, respectively. More precisely, we can consider the 1-Wasserstein metric with respect to the FGW distance to obtain a measure of distance between any two graph distributions P^{M} and P^{H}. By duality of the 1-Wasserstein distance this is equivalent to defining

$$d_{\mathcal{F}}(P^{\mathrm{M}}, P^{\mathrm{H}}) := \sup_{f \in \mathcal{F}} \big|\mathbb{E}(f(\mathrm{M})) - \mathbb{E}(f(\mathrm{H}))\big|, \tag{4}$$

for $\mathcal{F}$ the set of test functions $f : \mathbb{G} \to \mathbb{R}$ that are Lipschitz with respect to the FGW distance. This class of test functions contains the comparison to a reference graph ξ, i.e. the functions $f_\xi(\mu) = d_{FGW}(\xi, \mu)$ are Lipschitz with respect to the FGW distance. This is a direct consequence of the reverse triangle inequality.

From here onwards we assume that there is a $C > 0$, such that

$$|d_X(x_i, x_k) - d_Y(y_j, y_l)| \leq C \text{ for any } i, j, k, l. \tag{5}$$

3 AN ϵ-DIFFERENTIALLY PRIVATE SYNTHETIC DATA ALGORITHM FOR VERTEX ATTRIBUTES

In the following, we describe a differentially private mechanism that generates private synthetic data, which will serve as input of our network model. This mechanism, called the total variation private signed measure mechanism (TV-PSMM), is based on the private signed measure mechanism (PSMM) of He et al. (2023). All proof details are deferred to Appendix D .

Given the true data $\mathcal{X} \subset \Omega$ in a metric space (Ω, d) and a partition $(\Omega_1, \ldots, \Omega_m)$ of Ω, the PSMM first generates a new dataset $\mathcal{Y}$ by sampling a representative y_i in each cell Ω_i. By adding i.i.d. discrete Laplacian noise $\lambda_1, \ldots, \lambda_m$ to the counts of true data points n_i in each cell Ω_i, a signed measure ν that is supported on $\mathcal{Y}$ is obtained. Finally, the private synthetic dataset is created by sampling m' realizations with respect to the probability measure $\hat{\nu}$ that is the closest probability measure to μ with respect to the bounded Lipschitz distance. Note that the bounded Lipschitz distance can be seen as a generalization of the 1-Wasserstein distance to signed measures, see He et al. (2023, Section 2).

We modify the algorithm of He et al. (2023) by allowing more general noise $\lambda_1, \ldots, \lambda_m \overset{i.i.d.}{\sim} P_\Lambda$ for P_Λ the distribution of some scalar random variable Λ. Additionally,

in the linear programming step, instead of using the bounded Wasserstein distance, which is often not easy to evaluate, we choose the closest probability measure $\hat{\nu}$ with respect to the total variation distance. Furthermore, we output this optimal probability measure $\hat{\nu}$ instead of a private synthetic dataset sampled from $\hat{\nu}$. For completeness, we state the TV-PSMM algorithm before we give a new linear program in Algorithm 2 to find the optimal probability measure $\hat{\nu}$.

Algorithm 1 Total Variation Private Signed Measure Mechanism (TV-PSMM)

Input: True data $\mathcal{X} = (x_1, \dots, x_n) \in \Omega^n$, partition $(\Omega_1, \dots, \Omega_m)$ of Ω, privacy parameter $\epsilon > 0$, noise distribution P_Λ.

Compute the true counts Compute the true count in each set of the partition, $n_i = \#\{x_j \in \Omega_i : j \in [n]\}$.

Create a new dataset For each $i \in [n]$, uniformly choose an element $y_i \in \Omega_i$ independently of $\mathcal{X}$, and let $\mathcal{Y}$ be the collection of n_i copies of y_i.

Add noise Choose i.i.d. $\lambda_i \sim P_\Lambda$; perturb the empirical measure $\mu_\mathcal{Y}$ of $\mathcal{Y}$ to obtain a signed measure ν given by

$$\nu(\{y_i\}) := (n_i + \lambda_i)/n.$$

Linear programming Using Algorithm 2, find $\hat{\nu}$, a probability measure that is closest to ν with respect to the total variation distance.

Output: The probability measure $\hat{\nu}$ on $\mathcal{Y}$ describing the distribution of synthetic data.

Algorithm 2 Linear Programming

Input: A discrete signed measure ν supported on $\mathcal{Y} = \{y_1, \dots, y_m\}$.

Solve the linear program Solve the linear programming problem with $2m$ variables and $3m+1$ constraints:

$$\min \sum_{i=1}^{m} u_i \text{ s.t. } \forall i \le m: \; u_i \ge \nu(\{y_i\}) - \tau_i, \quad u_i \ge \tau_i - \nu(\{y_i\}), \quad \tau_i \ge 0; \; \sum_{i=1}^{m} \tau_i = 1.$$

Output: A probability measure $\hat{\nu}$ with $\hat{\nu}(\{y_i\}) = \tau_i$.

Following the proof of Proposition 5 in He et al. (2023), the following result is obtained directly.

Proposition 3.1. *If for all k in the support of P_Λ and for $a \in \{-1, 1\}$, we have $\frac{P_\Lambda(k+a)}{P_\Lambda(k)} \le e^\epsilon$, then the Algorithm 1 is ϵ-differential private.*

Example 3.2. *The assumption of Proposition 3.1 is satisfied for*

- $P_\Lambda = \mathrm{Lap}_\mathbb{Z}(1/\epsilon)$*, the discrete Laplace distribution with variance* $2/\epsilon^2$
- $P_\Lambda = \mathcal{N}_{\mathbb{Z}\cap[-b,b]}(0, 1/\epsilon^2)$*, the centered discrete Gaussian distribution (Canonne et al., 2020) truncated to a finite interval* $[-b, b]$ *for some* $b > 0$*, see Appendix F for details*
- $P_\Lambda(k) = |k|^\epsilon Z(A, \epsilon)$ *for* $1 \le |k| \le A$*, with* $Z(A, \epsilon)$ *the normalizing constant, and* $A \in \mathbb{N}$.

A detailed inspection of Algorithm 2 shows the following result; the proof is in Appendix D.

Proposition 3.3. *Algorithm 2 outputs a probability measure $\hat{\nu}$ on $\mathcal{Y}$ that is closest to the given discrete signed measure ν on $\mathcal{Y}$ with respect to the total variation distance.*

4 PRIVATE SYNTHETIC NETWORK GENERATION

Next, we provide a precise definition of the (attributed) network models and their associated distributions. These models are based on the random connection model, which has been widely used to analyze complex data, see for example Krioukov et al. (2010) and Penrose (2016). A more detailed discussion can be found in Appendix D. We also present an algorithm that jointly generates a network constructed from the true data and a network with private data input. The private data input is thereby obtained in an initial step of the network generator by applying TV-PSMM introduced in Section 3. Finally, we show that the resulting networks have the desired distribution.

4.1 Synthetic network model

Let $\mathcal{X} = \{x_1, \dots, x_n\} \subset \Omega$ be a input data set of attributes and $\kappa : \Omega \times \Omega \to [0, 1]$ a (given) symmetric edge connection function; examples include Chung-Lu models and graphon models (see Matias and Robin (2014b) for a survey), see also Appendix D.2.

We start by modeling a (synthetic) network constructed from the true input data and having an expected size $a > 0$. In a first step, we draw a Poisson distributed number of vertex attributes, $N \sim \mathrm{Poi}(a)$, uniformly from the true input data set $\mathcal{X}$. We then add a uniform identifier to each vertex to obtain attributed vertices (X_i, U_i) that consist of a (possibly) non-unique attribute X_i and a unique identifier U_i. Finally, we connect any two distinct attributed

vertices $(X_i, U_i), (X_j, U_j)$, $i \neq j$, with probability $\kappa(X_i, X_j)$, independently of other edges.

We derive a mathematical description of this (synthetic) network (H, T) using point processes. This enables us to represent the vertex process by $\mathrm{H} = \sum_{i=1}^{N} \delta_{(X_i, U_i)}$ for $N \sim \mathrm{Poi}(a)$ and uniformly chosen $X_i \sim \mathcal{U}(\mathcal{X})$ and $U_i \sim \mathcal{U}([0,1])$, $i \in [N]$. Similarly we represent the edge process as $\mathrm{T} = \sum_{\substack{i,j=1 \\ i \neq j}}^{N} K_{ij}^{\mathrm{T}} \delta_{\{U_i, U_j\}}$ for (Bernoulli-distributed) $K_{ij}^{\mathrm{T}} \overset{i.i.d.}{\sim} \mathrm{Ber}(\kappa(X_i, X_j))$, $i, j \in [N]$. Furthermore, we denote the distribution of (H, T) by $P^{\mu_{\mathcal{X}}^a} \otimes Q^{\kappa}$ and refer to the Appendix D.1 for technical details.

We construct a (synthetic) network (Ξ, Σ) with private data input on $\mathcal{Y} \times [0,1]$ with $\mathcal{Y} = \{y_1, \ldots, y_m\} \subset \Omega$ analogously, as follows. The vertices are given by a point process $\Xi = \sum_{i=1}^{M} \delta_{(Y_i, V_i)}$ where $M \sim \mathrm{Poi}(b)$ for some $b > 0$, $V_i \sim \mathcal{U}([0,1])$ and $Y_i \in \mathcal{Y}$ is chosen with respect to $\hat{\nu}$. Furthermore, we define the edge process by $\Sigma = \sum_{\substack{i,j=1 \\ i \neq j}}^{M} K_{ij}^{\Sigma} \delta_{\{V_i, V_j\}}$ for $K_{ij}^{\Sigma} \overset{i.i.d.}{\sim} \mathrm{Ber}(\kappa(Y_i, Y_j))$. This yields a random graph $(\Xi, \Sigma) \sim P^{\mu_{\mathcal{Y}}^b} \otimes Q^{\kappa}$.

In Section C.1 of the Appendix, extensive simulations from this model are shown.

4.2 The private synthetic network generator

We introduce a method to jointly draw realizations of the network model with true data input and the network model with private data input described in Section 4.1; the procedure is described in Algorithm 3. The computational complexity of Algorithm 3 grows polynomially with the partition size m for TV-PSMM, and quadratically with the graph size $\max\{a, b\}$ for the graph sampling, see Table 1. We refer to Appendix D.3 and Appendix C for a detailed complexity analysis and simulation results, respectively.

The following result guarantees that Algorithm 3 generates suitable graph samples.

Theorem 4.1. *The graphs (Ξ, Σ) and (H, T) obtained by Algorithm 3 have distributions $P^{\mu_{\mathcal{Y}}^b} \otimes Q^{\kappa}$ and $P^{\mu_{\mathcal{X}}^a} \otimes Q^{\kappa}$, respectively.*

Sketch of the proof of Theorem 4.1 We show that the vertex counts $(M_1, \ldots, M_m)$ and $(N_1, \ldots, N_m)$ constructed in Algorithm 3 have multinominal distributions and thus the correct marginal distributions. This implies that the vertex processes constructed in Algorithm 3 define a coupling of the vertex measures of the graph models introduced in Section 4.1. Furthermore, the construction of the edge processes corresponds to a maximal coupling of the corresponding Bernoulli random variables with suitable parameters. Hence, the graphs constructed in Algorithm 3 have the claimed

Algorithm 3 Private Synthetic Graph Generation (PSGG)

Input: True data $\mathcal{X} = (x_1, \ldots, x_n) \in \Omega^n$, partition $(\Omega_1, \ldots, \Omega_m)$ of Ω, privacy parameter $\epsilon > 0$, noise distribution P_Λ, expected network sizes a and b, edge probability function κ.

TV-PSMM Apply Algorithm 1 to obtain the probability measure $\hat{\nu}$ on $\mathcal{Y} = \{y_1, \ldots, y_m\}$ describing the distribution of private synthetic data.

Sample network size Sample $K_N \sim \mathrm{Poi}(a - (a \wedge b))$, $K_M \sim \mathrm{Poi}(b - (a \wedge b))$ and $L \sim \mathrm{Poi}(a \wedge b)$ and set $N = L + K_N$ and $M = L + K_M$.

Create common vertex counts For $j \in [L]$ sample $(Z_1^j, \ldots, Z_m^j) \in \{0,1\}^m$, according to

$$\mathbb{P}((Z_1^j, \ldots, Z_m^j) = (l_1, \ldots, l_m)) = \begin{cases} \frac{n_k}{n} \wedge \hat{\nu}_k & \text{if } l_k = 1 \\ 1 - \sum_{k \in [m]} \frac{n_k}{n} \wedge \hat{\nu}_k & \text{if } \sum_{k \in [m]} l_k = 0 \end{cases}$$

for $l_1, \ldots, l_m \in \{0,1\}$ such that $\sum_{k \in [m]} l_k \leq 1$. Set $Z = \sum_{j \in [L]} \sum_{k \in [m]} Z_k^j$.

Create non-common vertex counts Sample $(M_1, \ldots, M_m) \sim \mathrm{Multin}\big(M - Z, (\hat{\nu}_1, \ldots, \hat{\nu}_m)\big)$, $(N_1, \ldots, N_m) \sim \mathrm{Multin}\left(N - Z, \left(\frac{n_1}{n}, \ldots, \frac{n_m}{n}\right)\right)$.

Create vertices Sample

point attributes $X_{kj} \sim \mathcal{U}(\mathcal{X} \cap \Omega_k)$

i.i.d. identifiers $V_{kj} \sim \mathcal{U}[0,1]$

$U_{kj} \sim \mathcal{U}[0,1]$.

Set $\Xi = \sum_{k \in [m]} \sum_{j \in [Z_k + M_k]} \delta_{(y_k, V_{kj})}$ and

$\mathrm{H} = \sum_{k \in [m]} \sum_{j \in [Z_k + N_k]} \delta_{(X_{kj}, U_{kj})}$.

Create edges For $k, l \in [m]$ and $i \in [Z_k]$, $j \in [Z_l]$ sample K_{kilj}^{T} and K_{kilj}^{Σ} according to

$$\mathbb{P}((K_{kilj}^{\mathrm{T}}, K_{kilj}^{\Sigma}) = (e_1, e_2)) = \begin{cases} \kappa_{kilj}^X \wedge \kappa_{kl}^y & \text{if } (e_1, e_2) = (1,1) \\ \kappa_{kilj}^X - \kappa_{kilj}^X \wedge \kappa_{kl}^y & \text{if } (e_1, e_2) = (1,0) \\ \kappa_{kl}^y - \kappa_{kilj}^X \wedge \kappa_{kl}^y & \text{if } (e_1, e_2) = (0,1) \\ 1 - \kappa_{kilj}^X \vee \kappa_{kl}^y & \text{if } (e_1, e_2) = (0,0) \end{cases}$$

where $\kappa_{kilj}^X = \kappa(X_{ki}, X_{lj})$ and $\kappa_{kl}^y = \kappa(y_k, y_l)$. For $i > Z_k$ or $j > Z_l$ sample $K_{kilj}^{\Sigma} \sim \mathrm{Ber}(\kappa(y_k, y_l))$ and $K_{kilj}^{\mathrm{T}} \sim \mathrm{Ber}(\kappa(X_{ki}, X_{lj}))$.

Set $\Sigma = \sum_{k,l \in [m]} \sum_{i=1}^{M_k} \sum_{j=1}^{M_l} K_{kilj}^{\Sigma} \delta_{\{V_{ki}, V_{lj}\}}$ and $\mathrm{T} = \sum_{k,l \in [m]} \sum_{i=1}^{N_k} \sum_{j=1}^{N_l} K_{kilj}^{\mathrm{T}} \delta_{\{U_{ki}, U_{lj}\}}$.

Output: Private synthetic network (Ξ, Σ), and network (H, T) constructed with true data input.

Table 1: Computational complexity of the components of the graph generator.

Component	Complexity	Remarks
Solving linear program	$\mathcal{O}(m^{2+\delta})$	Faster than solving the LP in PSMM ($\mathcal{O}(m^{2(2+\delta)})$)
Sampling the graph	$\mathcal{O}(a^2+b^2)$	Depends on graph sizes a, b
Other components	Lower order	Negligible compared to above

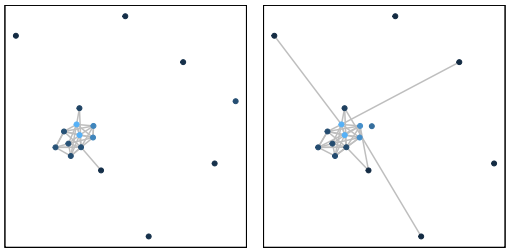

Figure 2: Network representation of Florentine family data collected by Padgett and Ansell (1993) using the wealth attribute, obtained by a joint realization. Left: Realization of a network obtained from the observed data. Right: Realization of a network obtained from the data in a differential private way with privacy parameter $\epsilon = 2$, expected network size $a = b = 16$, and number of cells in the partition $m = 10$.

marginal distributions. The full proof is given in Appendix D.

Remark 4.2. *PSGG in Algorithm 3 first applies a privacy mechanism to the vertices, and then constructs the edges based on the privatized vertex representations. Thus, the edge generation mechanisms is also private, and PSGG is a private mechanism.*

Figure 2 gives an example of this network generation yielding a network representation of the often used Florentine families data from Padgett and Ansell (1993) based *not* on marriage relations or business relations but on wealth, normalized by the maximum wealth, using $\kappa(x,y) = \min(2xy, 1)$. This gives a random graph model of Chung-Lu type, where vertices with high weights/attributes (light color) are more likely to form edges than vertices with small weights (dark color).

In Appendix C.2 a more involved example is found, based on the 2011 census in England & Wales.

5 THEORETICAL GUARANTEES ON UTILITY

Next we give theoretical results for the PSGG algorithm and conditions under which the distribution of the network (Ξ, Σ) with private data input $\hat{\nu}$ is close to the distribution of the network (H, T) with true data input $\mathcal{X}$. While Algorithm 3 is generally applicable, we assume here that the edge connection function is Lipschitz continuous in order to establish theoretical guarantees and refer to Appendix D.2 for a discussion of this assumption. The first part addresses the utility of Algorithm 3 while the second part analyzes the difference in distribution. All proofs details are deferred to Appendix E.

5.1 Utility of Algorithm 3

To provide an accuracy guarantee we bound the expected distance between the random network (H, T) and the random network (Ξ, Σ) jointly constructed by Algorithm 3 with respect to the FGW distance, see Section 2.1. That is, we study $\mathbb{E}(d_{FGW}(\eta^{(\mathrm{H},\mathrm{T})}, \eta^{(\Xi,\Sigma)}))$, where $\eta^{(\mathrm{H},\mathrm{T})}$ and $\eta^{(\Xi,\Sigma)}$ are random measures that capture the (attributed) network information of (H, T) and (Ξ, Σ), respectively. We recall the notion of diameter; $diam(\Omega) = \sup_{x,y\in\Omega} d(x,y)$.

Theorem 5.1. *Assume that κ is Lipschitz continuous with Lipschitz constant L_κ in both components and let (H, T) and (Ξ, Σ) be the networks with true and private data input of expected size $a > 0$ and $b > 0$, respectively, constructed by Algorithm 3. Then, using the FGW distance with parameter $\alpha \in [0,1]$,*

$$\begin{aligned}&\mathbb{E}\big(d_{FGW}(\eta^{(\mathrm{H},\mathrm{T})}, \eta^{(\Xi,\Sigma)})\big)\\ &\le \tilde{C}_1 \max_{k\in[m]} \mathrm{diam}(\Omega_k) \frac{1}{1+\frac{a\wedge b}{|a-b|}\mathbb{1}_{\{|a-b|>0\}}}\\ &\quad + 4\tilde{C}_2 \frac{m}{n}\mathbb{E}(|\Lambda|) + \tilde{C}_2\left(1 - \frac{1}{\big(1+\frac{a\wedge b}{|a-b|}\big)^2}\right)\mathbb{1}_{\{|a-b|>0\}},\end{aligned}$$

where $\tilde{C}_1 = 1-\alpha+\alpha(2\,C\,L_\kappa)$ and

$$\tilde{C}_2 = (1-\alpha)\mathrm{diam}(\Omega) + \alpha\min\{C, 2\,C\,L_\kappa \mathrm{diam}(\Omega)\},$$

with $C > 0$ given in (5).

In what follows we sketch the idea of the proof and refer to App. E for details. Note that the joint construction of the networks in Algorithm 3 allows us to significantly reduce the costs of the estimates.

Sketch of the proof of Theorem 5.1: For two pairs of vertices (X_i, Y_j) we bound the attribute cost by $\mathrm{diam}(\Omega)$ and the edge cost by $\min\{C, 2\,C\,L_\kappa \mathrm{diam}(\Omega)\}$ using the Lipschitz property of κ for the second term. If (X_i, Y_j) are in the same cell, this bound can be improved to $\max_{k\in[m]} \mathrm{diam}(\Omega_k)$ for the vertex part and

$2\,C\,L_\kappa \max_{k\in[m]} \mathrm{diam}(\Omega_k)$ for the edge part; details are found in Appendix E. We then obtain the result by conditioning on the (minimal) number Z of vertices (X_i, Y_i) that are in the same cell.

We consider a special case of Theorem 5.1 to give a more thorough discussion of the obtained bound; we set $\Omega = [0,1]^d$ to be the d-dimensional unit cube, we let $a = b$, and we choose the sample size $m = \lceil f_n(\epsilon)n \rceil$ for a function $f_n : \mathbb{R}_+ \to \mathbb{R}_+$. Under these assumptions Theorem 5.1 yields the following result.

Corollary 5.2. *Assume that κ is Lipschitz continuous with Lipschitz constant L_κ in both components and let $m = \lceil f_n(\epsilon)n \rceil$. Let (H, T) and (Ξ, Σ) be networks generated by Algorithm 3 using discrete noise Λ and a partition $(\Omega_k)_{k=1}^m$ satisfying $\mathrm{diam}(\Omega_k) \asymp m^{-1/d}$, with same expected size $a = b$ with attributes in $\Omega = [0,1]^d$. Then Theorem 5.1 simplifies to*

$$\mathbb{E}\big(d_{FGW}(\eta^{(\mathrm{H},\mathrm{T})}, \eta^{(\Xi,\Sigma)})\big) \le \frac{\tilde{C}_1}{m^{1/d}} + 2\tilde{C}_2 f_n(\epsilon)\mathbb{E}|\Lambda|. \tag{6}$$

In particular if $\Lambda \sim Lap_{\mathbb{Z}}(1/\epsilon)$ then

$$\mathbb{E}\big(d_{FGW}(\eta^{(\mathrm{H},\mathrm{T})}, \eta^{(\Xi,\Sigma)})\big) \le \frac{\tilde{C}_1}{m^{1/d}} + 4\tilde{C}_2 f_n(\epsilon)\frac{e^{-\epsilon}}{1-e^{-2\epsilon}}.$$

The last part of the corollary follows from the fact that discrete Laplacian noise $\Lambda \sim \mathrm{Lap}_{\mathbb{Z}}(1/\epsilon)$ has $\mathbb{E}|\Lambda| = 2\frac{e^{-\epsilon}}{1-e^{-2\epsilon}}$, see Inusah and Kozubowski (2006). A partition of the unit cube $\Omega = [0,1]^d$ that satisfies the diameter assumption in Corollary 5.2 can be obtained by dividing the space into m bins of equal length $m^{-1/d}$. A reasonable choice of the function f_n is essential to obtain good convergence rates; see Section 5.3. As another discrete noise example, the truncated discrete Gaussian noise is discussed in Appendix F.

5.2 Bounds on the difference between the graph distributions

Next we analyze and measure the differences between the distribution $(P^{\mu_{\mathcal{X}}^a} \otimes Q^\kappa)$ of the random network (H, T) with true input data and the distribution $(P^{\mu_{\mathcal{Y}}^b} \otimes Q^\kappa)$ of the random network (Ξ, Σ) with private data input. To measure the distance of random network distributions, we use the Wasserstein distance $d_{\mathcal{F}}$ from (4), with respect to the FGW distance, as an underlying metric, see Section 2.1. As a direct consequence of the Lipschitz property and Theorem 4.1, the results obtained in Section 5.1 can be used to bound this distance.

We derive upper bounds for these distances of random network distributions by adapting general bounds from Schuhmacher and Wirth (2024). In general, it is possible to consider different classes of test functions and thus obtain different types of so-called *integral probability metrics*, see Müller (1997). Our approach will then work analogously; however, the obtained upper bounds and rates will strongly depend on the choice of test functions.

Theorem 5.3. *Assume that κ is Lipschitz continuous with Lipschitz constant L_κ in both components and let $(P^{\mu_{\mathcal{X}}^a} \otimes Q^\kappa)$ and $(P^{\mu_{\mathcal{Y}}^b} \otimes Q^\kappa)$ be distributions of the network models with true and private data input of expected size $a > 0$ and $b > 0$, respectively, constructed as in Section 4.1 by using a probability measure $\hat{\nu}$ created by Algorithm 1 with discrete random noise Λ. Then, using the FGW distance with parameter $\alpha \in [0,1]$,*

$$\begin{aligned} &d_{\mathcal{F}}(P^{\mu_{\mathcal{Y}}^b} \otimes Q^\kappa, P^{\mu_{\mathcal{X}}^a} \otimes Q^\kappa) \\ &\le \left(1 + \frac{1}{1 + \frac{a\wedge b}{|a-b|}\mathbb{1}_{\{|a-b|>0\}}}\right)(1-\alpha) \max_{k\in[m]} \mathrm{diam}(\Omega_k) \\ &\quad + \tilde{C}_2\left(1 - \frac{1}{\big(1 + \frac{a\wedge b}{|a-b|}\big)^2}\right)\mathbb{1}_{\{|a-b|>0\}} \\ &\quad + 2c_V(a\wedge b)\frac{a\wedge b}{n}\mathrm{Leb}^d(\Omega)\mathbb{E}|\Lambda| \\ &\quad + c_E(a\wedge b)\,2L_\kappa \max_{k\in[m]} \mathrm{diam}(\Omega_k)^3 (a\wedge b)^2, \end{aligned} \tag{7}$$

where

$$c_V(c) := \min\Big\{1, \frac{1}{c}\big(1 + (1-e^{-c})\log^+ c\big)\Big\}C_\alpha,$$
$$c_E(c) := \min\Big\{1, \frac{2-e^{-c}}{c} - \frac{1}{c^2}\Big(\frac{3}{2} - e^{-c}\Big)\Big\}\alpha C$$

for $C_\alpha := (1-\alpha)\mathrm{diam}(\Omega) + \alpha C$ and $\tilde{C}_2 = (1-\alpha)\mathrm{diam}(\Omega) + \alpha \min\{C, 2CL_\kappa \mathrm{diam}(\Omega)\}$ with $C > 0$ given in (5).

We give a sketch of the proof for Theorem 5.3 and refer to App E for details.

Sketch of the proof of Theorem 5.3. The proof is based on results of Schuhmacher and Wirth (2024, Theorem 4.7), which provides upper bounds on the distance of two random (attributed) network distributions. The bounds in Schuhmacher and Wirth (2024, Theorem 4.7) are of total variation type and thus will not vanish in a direct comparison (as the attribute measures have different support); they are also sensitive to the number of vertices. We approach these issues by creating intermediate graphs that are supported on the whole space Ω and have the same size $N \wedge M (= \min(N, M))$. The difference between the original graphs and their corresponding intermediate graphs can be analyzed using a coupling, while a comparison of the intermediate graphs can be carried out by an application of Schuhmacher and Wirth (2024,

Theorem 4.7). Finally, we assemble all results to obtain the claimed upper bound.

The bound obtained in Theorem 5.3 simplifies under additional assumptions on the attribute space and its partition as well as the noise distribution similar to those in Corollary 5.2.

Corollary 5.4. *Set the expected network sizes to be $a = b$ and $\Omega = [0,1]^d$. Assume that κ is Lipschitz continuous with Lipschitz constant L_κ in both components. Consider networks being distributed according to Section 4.1 w.r.t. a probability measure $\hat{\nu}$ created by Algorithm 1 using discrete noise Λ and a partition $(\Omega_k)_{k=1}^m$ satisfying $\operatorname{diam}(\Omega_k) \asymp m^{-1/d}$. Then Theorem 5.3 simplifies to*

$$d_{\mathcal{F}}(P^{\mu_{\mathcal{Y}}^b} \otimes Q^\kappa, P^{\mu_{\mathcal{X}}^a} \otimes Q^\kappa) \leq 2\frac{1-\alpha}{m^{1/d}} + C_\alpha \frac{(1+\log^+ a)}{n}\mathbb{E}|\Lambda| + 4\alpha C L_\kappa \frac{a}{m^{3/d}}. \tag{8}$$

Similarly to the results in Section 5.1, we can obtain a partition fulfilling the diameter assumption in Corollary 5.4 by dividing the unit cube $\Omega = [0,1]$ into m equally sized bins. Again, a reasonable choice of the (expected) size $a = a_n$ of the random network and the partition size $m = \lceil f_n(\epsilon)n \rceil$ are essential to obtain good convergence rates, as we discuss in the next section.

5.3 Choice of parameters

In practice, we often are given a dataset $\mathcal{X}$ of size n and a privacy level ϵ and have to make suitable choices for the expected network size a and the partition size $m = \lceil f_n(\epsilon)n \rceil$. Here we discuss a suitable choice of these parameters using the bounds obtained by Theorem 5.1 and Theorem 5.3. To obtain the simplified bounds stated in Corollary 5.2 and Corollary 5.4, we restrict to networks having the same expected size $a = b$ with attributes on the unit cube $\Omega = [0,1]^d$ and assume that the networks are generated by Algorithm 3 using discrete Laplacian noise $\Lambda \sim \mathrm{Lap}_{\mathbb{Z}}(1/\epsilon)$. Furthermore, we consider a partition $(\Omega_k)_{k=1}^m$ that satisfies the diameter assumption $\operatorname{diam}(\Omega_k) \asymp m^{-1/d}$ and has size $m = \lceil f_n(\epsilon)n \rceil$, where the number of bins m depends on the privacy parameter ϵ through some function f_n. Note that for any $\epsilon \geq 0$ we have $\frac{\epsilon e^{-\epsilon}}{1-e^{-2\epsilon}} \leq 1/2$. Then Corollary 5.2 yields

$$\begin{aligned} d_{\mathcal{F}}(P^{\mu_{\mathcal{Y}}^b} \otimes Q^\kappa, P^{\mu_{\mathcal{X}}^a} \otimes Q^\kappa) &\leq \mathbb{E}\big(d_{FGW}(\eta^{(\mathrm{H,T})}, \eta^{(\Xi,\Sigma)})\big) \\ &\leq \tilde{C}_1 (f_n(\epsilon)n)^{-\frac{1}{d}} + 2\tilde{C}_2 f_n(\epsilon)\epsilon^{-1}, \end{aligned} \tag{9}$$

where the first inequality follows from the definition of $d_{\mathcal{F}}$, see Section 5.2, while Corollary 5.4 yields the uniform bound (8). The speed of convergence of the bounds in (8) and (9) depends on the choice of f_n, with the bound in (8) also depending on the expected network size a. We start with a suitable choice of network size. On the one hand, we would like to choose the expected network size $a = a_n$ as large as possible to have a good sample size for our graphs. On the other hand, (8) yields better bounds the smaller a is. The optimal convergence rate can be obtained setting $a = m^{2/d}$. A detailed derivation of this result can be found in the Appendix E.1.

It remains to find a suitable choice of $m = \lceil f_n(\epsilon)n \rceil$ by deriving an optimal decay of $f_n(\epsilon)$ as $\epsilon \to 0$. Note that in both, (9) and (8) with $a = m^{2/d}$, we obtain a trade-off for the choice of f_n. More precisely, to obtain fast convergence (to zero) in the first terms, we would like $f_n(\epsilon)$ to go to zero slowly. On the other hand, the second terms decrease faster the faster $f_n(\epsilon)$ goes to zero. It is shown in Appendix E.1 that an optimal decay can be obtained by setting $f_n \asymp \epsilon^{\frac{d}{d+1}} n^{-\frac{1}{d+1}}$ and thus $m = \lceil f_n(\epsilon)n \rceil \asymp \epsilon^{\frac{d}{d+1}} n^{\frac{d}{d+1}}$.

For this optimal parameter choice $m \asymp \epsilon^{\frac{d}{d+1}} n^{\frac{d}{d+1}}$ and under the above assumptions, (9) and thus Corollary 5.2 simplify to

$$\begin{aligned} d_{\mathcal{F}}(P^{\mu_{\mathcal{Y}}^b} \otimes Q^\kappa, P^{\mu_{\mathcal{X}}^a} \otimes Q^\kappa) &\leq \mathbb{E}\big(d_{FGW}(\eta^{(\mathrm{H,T})}, \eta^{(\Xi,\Sigma)})\big) \\ &\leq (\tilde{C}_1 + 2\tilde{C}_2)\big(\epsilon\, n\big)^{-\frac{1}{d+1}}. \end{aligned}$$

Similarly, setting $m \asymp \epsilon^{\frac{d}{d+1}} n^{\frac{d}{d+1}}$ and $a = m^{2/d}$ Corollary 5.4 turns to

$$\begin{aligned} &d_{\mathcal{F}}(P^{\mu_{\mathcal{Y}}^b} \otimes Q^\kappa, P^{\mu_{\mathcal{X}}^a} \otimes Q^\kappa) \\ &\leq \big(2(1-\alpha) + 4\alpha C L_\kappa\big)(\epsilon n)^{-\frac{1}{d+1}} \\ &\quad + C_\alpha \left(1 + \frac{2}{d+1}\log^+(\epsilon n)\right)(\epsilon n)^{-1}. \end{aligned}$$

Thus, with the above choice of f_n and a we obtain the same speed of convergence of order $n^{-\frac{1}{d+1}}$ for both approaches. This is similar to the rate of convergence derived in (He et al., 2023, Corollary 8) under very similar assumptions, when considering only the vertex attributes (privatized by PSMM). As we optimized f_n w.r.t. the results obtained in Section 5.1, for a different choice of f_n the results stated in Section 5.2 may yield improved rates.

Table 2 gives explicit upper bounds (see Appendix E for other parameter settings). Corollary 5.2 gives smaller bounds than Corollary 5.4 when ϵ is large, while none of the bounds is uniformly better than the other. Both bounds decrease when n increases.

Table 2: Explicit bounds obtained in Corollary 5.2 and Corollary 5.4 assuming discrete Laplacian noise and $\Omega=[0,1]^d$, $d=2$, $\alpha=\frac{1}{2}$ as well as $C=L_\kappa=1$. Furthermore, $m=(\epsilon n)^{\frac{d}{d+1}}$ and $a=m^{2/d}$ are optimal, see Section 5.3.

		n					
		100		1000		10.000	
		Corollary 5.2	Corollary 5.4	Corollary 5.2	Corollary 5.4	Corollary 5.2	Corollary 5.4
	1	0.751	0.681	0.349	0.304	0.162	0.140
ϵ	0.1	1.599	1.602	0.751	0.681	0.349	0.304
	0.01	3.061	3.721	1.599	1.602	0.751	0.681

6 CONCLUSION AND FUTURE WORK

In this work, we proposed PSGG, an algorithm that jointly generates a network representation as well as a private synthetic network from a given dataset. We provided a theoretical analysis of our algorithm studying the accuracy w.r.t. the fused Gromov Wasserstein distance and comparing the corresponding network distributions. Under additional assumptions, we derived optimal parameter choices and deduced that in this case our bounds are $\mathcal{O}(n^{-1/(d+1)})$.

We intend to extend our results to include (ϵ, δ)- differential privacy, see Dwork (2006), thus allowing for other discrete noise distributions, e.g., a discrete Gaussian distribution (on whole $\mathbb{Z}$), as in Canonne et al. (2020).

Another interesting further direction would be the study of networks with weighted edges. Here challenges lie in a possible dependence of the weight on the existence of an edge. Moreover, it would require a new notion of accuracy for attributed networks with weighted edges.

Furthermore, it would be worthwhile to study network embeddings. This would generalize our PSGG to the framework of graph-to-graph generation by treating embeddings as input data. Complication regarding privacy guarantees arise if the embeddings are created using information from all vertices and edges.

It would be interesting to evaluate our graph generator against privacy attacks. However, this would require a new type of privacy attacks as the sensitive information in our setting is solely contained at the vertices. As a result, edge-based privacy attacks (such as those discussed in Mislove et al. (2010); Jiang et al. (2021)) are not suitable.

It would be valuable to combine the node privacy studied in this paper with a suitable notion of edge privacy (assuming additional edge information). This could be achieved, for example, by adapting the rule how edges are being created by including a resampling mechanism on the edges. However, to our knowledge a suitable notion of such a joint privacy measure has not yet been investigated.

Another promising direction is the combined study of (differential) privacy and fairness constraints for graph generation. A strong privacy mechanism can cause minority information to vanish in the noise, potentially leading to severe fairness issues. Understanding this tension between privacy and fairness is therefore an important direction for future research.

Finally, a word of caution for applications: differential privacy guarantees do not guarantee complete privacy; they control the risk of privacy leakage.

Acknowledgements. We would like to thank the referees for their insightful comments and suggestions. Leoni Carla Wirth is funded by Deutsche Forschungsgemeinschaft (DFG) through GRK 2088 and by the EPSRC grant EP/T018445/1. Gesine Reinert is funded in part by EPSRC grants EP/T018445/1, EP/V056883/1, EP/Y028872/1, and EP/X002195/1. Gholamali Aminian acknowledges the support of the UKRI Prosperity Partnership Scheme (FAIR) under EPSRC Grant EP/V056883/1 and the Alan Turing Institute. For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

M. Abroshan, A. Elliott, and M. M. Khalili. Imposing fairness constraints in synthetic data generation. In *International Conference on Artificial Intelligence and Statistics*, pages 2269–2277. PMLR, 2024.

A. Athreya, D. E. Fishkind, M. Tang, C. E. Priebe, Y. Park, J. T. Vogelstein, K. Levin, V. Lyzinski, Y. Qin, and D. L. Sussman. Statistical inference on random dot product graphs: a survey. *Journal of Machine Learning Research*, 18(226):1–92, 2018.

V. Batagelj and U. Brandes. Efficient generation of large random networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 71(3):036113, 2005.

L. Brogat-Motte, R. Flamary, C. Brouard, J. Rousu, and F. d'Alché Buc. Learning to predict graphs with fused Gromov-Wasserstein barycenters. In *International Conference on Machine Learning*, pages 2321–2335. PMLR, 2022.

C. L. Canonne, G. Kamath, and T. Steinke. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems*, 33:15676–15688, 2020.

W.-Y. Day, N. Li, and M. Lyu. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*, pages 123–138, 2016.

Q. Duchemin and Y. De Castro. Random geometric graph: Some recent developments and perspectives. *High Dimensional Probability IX: The Ethereal Volume*, pages 347–392, 2023.

C. Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2006.

X. Han, Y. Yang, L. Wang, and J. Wu. Privacy-preserving network embedding against private link inference attacks. *IEEE Transactions on Dependable and Secure Computing*, 21(2):847–859, 2023.

Y. He, R. Vershynin, and Y. Zhu. Algorithmically effective differentially private synthetic data. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 3941–3968. PMLR, 2023.

L. Hou, W. Ni, S. Zhang, N. Fu, and D. Zhang. PPDU: dynamic graph publication with local differential privacy. *Knowledge and Information Systems*, 65(7): 2965–2989, 2023.

S. Inusah and T. J. Kozubowski. A discrete analogue of the Laplace distribution. *Journal of Statistical Planning and Inference*, 136(3):1090–1102, 2006.

X. Jian, Y. Wang, and L. Chen. Publishing graphs under node differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):4164–4177, 2021.

H. Jiang, J. Pei, D. Yu, J. Yu, B. Gong, and X. Cheng. Applications of differential privacy in social network analysis: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):108–127, 2021.

O. Kallenberg. *Foundations of Modern Probability*, volume 2. Springer, 1997.

S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 457–476. Springer, 2013.

O. Kose and Y. Shen. Fairwire: Fair graph generation. *Advances in Neural Information Processing Systems*, 37:124451–124478, 2024.

D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná. Hyperbolic geometry of complex networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 82(3):036106, 2010.

G. Last, F. Nestmann, and M. Schulte. The random connection model and functions of edge-marked poisson processes: second order properties and normal approximation. *Annals of Applied Probability*, pages 128–168, 2021.

Y. Li, M. Purcell, T. Rakotoarivelo, D. Smith, T. Ranbaduge, and K. S. Ng. Private graph data release: A survey. *ACM Computing Surveys*, 55(11):1–39, 2023.

Q. Liu, O. Deho, F. Vadiee, M. Khalil, S. Joksimovic, and G. Siemens. Can synthetic data be fair and private? a comparative study of synthetic data generation and fairness algorithms. In *Proceedings of the 15th International Learning Analytics and Knowledge Conference*, pages 591–600, 2025a.

S. Liu, H. Du, Y. Cao, B. Yan, J. Liu, and M. Yoshikawa. Pgb: Benchmarking differentially private synthetic graph generation algorithms. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pages 1348–1361. IEEE, 2025b.

C. Matias and S. Robin. Modeling heterogeneity in random graphs through latent space models: a selective review. *ESAIM: Proceedings and Surveys*, 47: 55–74, 2014a.

C. Matias and S. Robin. Modeling heterogeneity in random graphs through latent space models: a selective review. *ESAIM: Proceedings and Surveys*, 47: 55–74, 2014b.

A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: inferring user profiles in online social networks. In *Proceedings of the Third*

ACM International Conference on Web Search and Data Mining, pages 251–260, 2010.

A. Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

J. F. Padgett and C. K. Ansell. Robust action and the rise of the Medici, 1400-1434. *The American Journal of Sociology*, pages 1259–1319, 1993.

M. D. Penrose. On a continuum percolation model. *Advances in Applied Probability*, 23(3):536–556, 1991.

M. D. Penrose. Connectivity of soft random geometric graphs. *Annals of Applied Probability*, pages 986–1028, 2016.

Z. Qian, R. Davis, and M. Van Der Schaar. Synthcity: a benchmark framework for diverse use cases of tabular synthetic data. *Advances in Neural Information Processing Systems*, 36:3173–3188, 2023.

Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 425–438, 2017.

G. Reinert and W. Xu. Steingen: Generating fidelitous and diverse graph samples. *arXiv preprint arXiv:2403.18578*, 2024.

P. Rubin-Delanchy, J. Cape, M. Tang, and C. E. Priebe. A statistical interpretation of spectral embedding: the generalised random dot product graph. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(4):1446–1473, 2022.

D. Schuhmacher and L. C. Wirth. Assignment based metrics for attributed graphs. *arXiv preprint arXiv:2308.12165*, 2023.

D. Schuhmacher and L. C. Wirth. Stein's method for spatial random graphs. *arXiv preprint arXiv:2411.02917*, 2024.

J. van den Brand. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 259–278. SIAM, 2020.

R. van der Hofstad, P. van der Hoorn, and N. Maitra. Local limits of spatial inhomogeneous random graphs. *Advances in Applied Probability*, 55(3):793–840, 2023.

T. Vayer, N. Courty, R. Tavenard, L. Chapel, and R. Flamary. Optimal transport for structured data with application on graphs. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6275–6284. PMLR, 2019.

T. Vayer, L. Chapel, R. Flamary, R. Tavenard, and N. Courty. Fused Gromov-Wasserstein distance for structured objects. *Algorithms*, 13(9):212, 2020.

K. Zahirnia, Y. Hu, M. Coates, and O. Schulte. Neural graph generation from graph statistics. *Advances in Neural Information Processing Systems*, 36, 2024.

S. Zhang, H. Hu, Q. Ye, and J. Xu. PrivDPR: Synthetic graph publishing with deep PageRank under differential privacy. *arXiv preprint arXiv:2501.02354*, 2025.

L. Zheng, D. Zhou, H. Tong, J. Xu, Y. Zhu, and J. He. Fairgen: Towards fair graph generation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 2285–2297. IEEE, 2024.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Yes]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A TABLE OF NOTATIONS

All notations are summarized in Table 3, in order of appearance in the text.

Notation	Definition
ϵ	privacy parameter
$\|\eta\|_{TV}$	total variation norm of a signed measure η
$d_{TV}(\eta_1, \eta_2) = \|\eta_1 - \eta_2\|_{TV}$	total variation distance between two signed measures η_1 and η_2
$d_{FGW}(\mu, \eta)$	fused Gromov-Wasserstein distance between two structured objects μ and η
$\Pi(\mu, \eta)$	the set of couplings between μ and η
$d_{\mathcal{F}}(\mu, \eta)$	1-Wasserstein distance between μ and η
(Ω, d)	metric space in which the dataset is embedded
$(\Omega_1, \ldots, \Omega_m)$	partition of Ω into m cells
$\mathcal{X}$	true dataset
$\mathcal{Y}$	privatized data set
P_Λ	distribution of a discrete noise random variable Λ
ν	signed measure
$\hat{\nu}$	probability measure
$[n]$	{1,. . . ,n}
n_i	the true count in cell Ω_i
$\mathrm{Lap}_{\mathbb{Z}}(1/\epsilon)$	the discrete Laplace distribution with variance $2/\epsilon^2$
$\mathcal{N}_{\mathbb{Z}\cap[-b,b]}(0, 1/\epsilon^2)$	the centered discrete Gaussian distribution truncated to $[-b, b]$ with variance $1/\epsilon^2$
κ	symmetric edge connection function
$\mathrm{Poi}(a)$	Poisson distribution with mean a
$\mathcal{U}(A)$	uniform distribution on the set A
$\mathrm{Ber}(p)$	Bernoulli distribution with mean p
$\mathrm{Bin}(n, p)$	binomial distribution with n objects and success probability p
(H, T)	synthetic network based on privatized data
$P^{\mu^b_{\mathcal{Y}}} \otimes Q^\kappa$	distribution of (H, T)
(Ξ, Σ)	synthetic network based on true data
$P^{\mu^a_{\mathcal{X}}} \otimes Q^\kappa$	distribution of (Ξ, Σ)
$a \wedge b$	the minimum between $a \in \mathbb{R}$ and $b \in \mathbb{R}$
$\mathrm{Multin}\big(N, (p_1, \ldots, p_k)\big)$	multinomial distribution with N independent trials and k possible outcomes with $\sum_{i=1}^k p_i = 1$
$\eta^{(\mathrm{H},\mathrm{T})}$	random measure capturing the attributed network information of (H, T)
$\eta^{(\Xi,\Sigma)}$	random measure capturing the attributed network information of (Ξ, Σ)
$diam(\Omega)$	$\sup_{x,y\in\Omega} d(x, y)$
L_κ	Lipschitz constant of κ (in both components)
$\mathrm{Leb}^d(\Omega)$	the d-dimensional Lebesgue measure $\Omega \in \mathbb{R}^d$
$\log^+ a$	the maximum between $\log a$ and 0
$a \asymp m^k$	a is of the same order of magnitude as m^k

Table 3: Summary of notations in the paper

B MORE RELATED WORK

B.1 Fused Gromov-Wasserstein metric

The fused Gromov-Wasserstein metric introduced in Vayer et al. (2019, 2020) measures the dissimilarity of two structured objects with features by combining a Wasserstein metric for a feature comparison and a Gromov-Wasserstein metric for a structural comparison. In this paper we are interested in the special case that the structured objects are attributed networks (Vayer et al., 2020, Section 4). In this setting the features are given by vertex attributes in some metric space (Ω, d), while the structural component captures the graph structure using

a metric measure space. The metric measure space is constructed by endowing the metric space (Ω, d) with a measure μ that encodes the distribution of vertex mass. Here, the metric describes the edge structure in terms of intra-graph distances, for example by a shortest path distance.

Using this metric measure space, we define a representation of an attributed network with (vertex) attributes in some metric space (Ω, d) by a measure $\mu = \sum_{i=1}^{n} h_i \delta_{(x_i, a_i)}$, where $(h_i)_{i\in[n]} \subset \mathbb{R}_+$ such that $\sum_{i=1}^{n} h_i = 1$. Here the $a_i \in \Omega$ encode the vertex attribute information while the structural information is assumed to be captured by a complete finite metric space (X, d_X) which we interpret as graph, with the elements $x_i \in X$ as vertices and edge structure induced by the metric d_X. For example, if the d_X captures the shortest path distances between vertices in X, then the graph contains an edge between two vertices x_i, $x_j \in X$ whenever $d_X(x_i, x_j) = 1$.

For two attributed networks $\mu = \sum_{i=1}^{n} h_i \delta_{(x_i, a_i)}$ and $\eta = \sum_{j=1}^{m} g_j \delta_{(y_j, b_j)}$, with $y_j \in Y$, and (Y, d_Y) a complete metric space, the fused Gromov-Wasserstein (FGW) distance (or metric) with parameters $p, q \geq 1$ and $\alpha \in [0, 1]$ is defined as in (3), by

$$d_{FGW}(\mu, \eta) = \Big(\min_{\pi \in \Pi(\mu,\eta)} E_{\alpha,p,q}(\pi) \Big)^{\frac{1}{q}},$$

where

$$E_{\alpha,p,q}(\pi) := \Big(\underbrace{(1-\alpha) d(a_i, b_j)^q}_{\text{WS comparison attributes}} + \underbrace{\alpha |d_X(x_i, x_k) - d_Y(y_j, y_l)|^q}_{\text{Gromov WS comparison edge structure}} \Big)^p \underbrace{\pi((x_k, a_k), (y_l, b_l))}_{\text{mass transported from } a_k \text{ to } b_l}$$

for $\pi_{i,j} = \pi((x_i, a_i), (y_j, b_j))$, and $\Pi(\mu, \eta) := \{\pi \in \mathbb{R}_+^{n\times m} \mid \sum_i \pi_{ij} = b_j; \sum_j \pi_{ij} = a_i\}$ is the set of *couplings*; a coupling between two probability measures is a construction of the measures on a common probability space. The first term in the definition of $E_{\alpha,p,q}(\pi)$ can be interpreted as Wasserstein comparison of the attributes with respect to an underlying metric $d = d_\Omega$, while the second term can be viewed as a Gromov-Wasserstein comparison of the edge structures. Furthermore, $\pi_{i,j}$ gives the amount of mass that is transported from i to j by the coupling, while α can be seen as trade-off parameter between attribute and graph structure cost. We take $p = q = 1$ to simplify the notation. However, our results hold for general $p, q \geq 1$.

B.2 Differential privacy in graphs

This section briefly reviews related works about differential privacy in graphs.

Vertex and edge differential privacy in graphs were introduced in Kasiviswanathan et al. (2013) using the rewiring distance. Several works focus on differentially private graph publishing (Jian et al., 2021; Day et al., 2016; Hou et al., 2023). Other works propose practical approaches for private synthetic graph generation under different notions of privacy (Qin et al., 2017; Zahirnia et al., 2024). In Li et al. (2023) a survey of differentially private graph generation algorithms is provided, and in Liu et al. (2025b) a comparison of algorithms is carried out.

In contrast to these existing methods that take an entire network as input, our approach starts with complex data input that is used as attribute information to generate synthetic networks. We provide a theoretical foundation for this process, offering rigorous privacy guarantees for synthetic graph generation. We thereby focus on vertex differential privacy, which is appropriate in our setting as all the information is contained at the vertices itself, not at the edges. Furthermore, as our graph generator first applies a privacy mechanism to the vertices and then jointly constructs the edges of the synthetic network based on true data input and the edges of the network based on private data input, the edges are private as well. There is however currently no easy way to propagate information obtained from attacks on edges into privacy calculations. A detailed study of other forms of differential-privacy is thus an interesting direction for future research.

B.3 Fairness

Exploring connections between (different notions of) fairness and differential privacy is an important research direction. As discussed for example in Abroshan et al. (2024); Liu et al. (2025a); Kose and Shen (2024); Zheng et al. (2024), fairness in synthetic data often entails preserving the statistical properties of minority groups (fair representation), whereas (differential) privacy aims to mask the participation of a single individual in the database. In particular, if the privacy mechanism is so strong that minority information vanishes in the noise, severe fairness problems can arise. Hence there is a tension between privacy and fairness. While there are some synthetic data

generators available that satisfy both fairness and privacy constraints, the empirical study in Liu et al. (2025a) shows that these methods can suffer in utility.

In our proposed method (PSGG), fairness (based on minority group representation in synthetic dataset) could perhaps be integrated into the design of the mechanism itself if the sensitive fairness variable is not subject to privacy constraints. Then the fairness variable could be used to create a weighted sampling scheme; during the creation of the dataset in Algorithm 1, we can replace the standard uniform sampling with a weighted sampling method. This allows us to enforce constraints that prevent the under-representation of minority groups in the generated synthetic network.

However, if the sensitive fairness variable is itself subject to privacy requirements, then different approaches are required. One option is to include data oversampling or even data augmentation before network creation, balancing the data set which will then be subject to the PSGG method.

Another possibility would be to take different noise levels for different types of observations, without revealing sensitive information, which may not be straightforward if the sensitive information is also private. It may also be possible to first create a synthetic data set that satisfies fairness constraints, for example as in Abroshan et al. (2024), and then use PSGG for network generation. However possible privacy leaks in the synthetic data set generation would have to be taken into account when providing theoretical guarantees.

B.4 Network embeddings

Combining network embeddings with our current graph generator would allow to generalize our PSGG to the framework of graph-to-graph generation by treating embeddings as the input data. The mechanism would then construct a new graph using a kernel in a random dot product graph construction. A complication regarding privacy guarantees arises if the embeddings are created using information from all vertices and edges.

Moreover, one could even consider starting with a graph as input. A suitable workflow could then look as follows:

1. Convert the graph (with nodes) into embedding vectors using a privacy method such as in Han et al. (2023).
2. Apply TV-PSMM to these embedding vectors (treating them as the attribute data).
3. Construct a new graph using the dot product of the privatized embeddings.

We would like to clarify the privacy implication here. By privatizing the embeddings, we are effectively protecting the "structural position" of the vertices in the latent space. This workflow fits our vertex-privacy framework, but it is distinct from standard edge-differential privacy. We highlight that to preserve the privacy guarantees, it is essential that the edge kernel either does not depend on the input networks or is privatized in a suitable way. Moreover the privacy budget for the embedding will have to be taken into account when providing an overall privacy guarantee.

C SIMULATION STUDY

In the following, we study the new network generator PSGG described in Algorithm 3 by running simulations on synthetic data (Section C.1) and real-world data (Section C.2). We compare PSGG to a new baseline method which first applies PSMM by He et al. (2023) to the vertex attributes to generate privatized vertex attributes. In a second step, the graph structure of the synthetic networks based on the true data input and the privatized data input, respectively, are generated independently according to the random connection model with edge kernel function κ. In contrast to this baseline method, PSGG stated in Algorithm 3 first applies the TV-PSMM algorithm to the privatized vertex attributes and then uses a joint sampling strategy to create the network representations, again based on the edge kernel function κ. While both methods yield a privacy mechanism, the following simulations show that the independent sampling in the new baseline method destroys the utility.

All simulations were performed on a HP ProLiant server with two Intel Xeon X5660 processors in R 4.2.0 (R Core Team, 2023). The experiments can be reproduced with the code provided under `https://github.com/LCWirth/PrivateNetworkGen.git` using the seed stored in the file name.

C.1 Synthetic data example

We consider a synthetic data example to analyze the practical behavior of the private synthetic network generator PSGG given by Algorithm 3. We study input data in the attribute space $\Omega = [0, 1]$ that is given by the input data set $\mathcal{X} = \{a_1, \dots, a_n\}$ of different size $n \in \{100, 1000, 10000\}$ containing the attributes $a_i = 0$ for $1 \leq i \leq n/2$ and $a_i = 1$ for $n/2 + 1 \leq i \leq n$, and apply the privacy mechanism in Algorithm 1 with decreasing privacy parameter $\epsilon \in \{1, 0.1, 0.01\}$ and thus increasing Laplacian noise $\mathrm{Lap}_{\mathbb{Z}}(1/\epsilon)$. We construct a network representation of this data using the network model introduced in Section 4.1 with $\kappa(x, y) = xy$, which yields a network of Chung-Lu type. To analyze the effect of varying sample size n and privacy parameter ϵ, we fix the expected network size $a = b = 100$ while the partition size $m = \max\{\lceil\sqrt{\epsilon n}\rceil, 5\}$ is chosen according to the optimality discussion in Section 5.3. In what follows we abbreviate this by 'the optimal choice'; it is understood to mean: chosen according to the optimality discussion in Section 5.3. In abuse of the notion, although optimality conditions are only given in terms of orders of magnitude, here we take the respective constants to be 1, and any lower terms are taken to be zero. The corresponding partition $(\Omega_i)_{i=1}^m$ is obtained by dividing the unit interval $\Omega = [0, 1]$ into m bins of equal length m^{-1}. In Figure 3, Figure 4 and Figure 5 the joint realizations of the network model with expected network size $a = b = 100$ based on true data input (of size $n \in \{100, 1000, 10000\}$, respectively, and based on private data input at different levels of privacy $\epsilon \in \{0.01, 0.1, 1\}$) are presented.

Furthermore, Table 4 and Table 6 compare PSGG given in Algorithm 3 with the naive baseline method (referred to as PSMM). For each graph-generation mechanism, utility is assessed by evaluating the similarity between the networks generated from the true data and from the privatized data. This similarity is measured in terms of their average network distance (FGW) and their degree and closeness similarity. Here, the degree and closeness distributions of two networks are each compared by the MMD kernel and we output mean and standard deviation. Additionally, Table 5 and Table 7 give the average runtime for the linear program (contained in TV-PSMM and the PSMM, respectively) and the whole graph generation mechanism.

In Table 4 and Table 5 the sample size n varies, while the expected graph size a and partition size m are chosen to be optimal, see Table 10. It can be observed that for both graph generators an increasing sample size n can balance out an decreasing privacy parameter ϵ with respect to the utility. In Table 6 and Table 7 the privacy parameter ϵ varies with partition size m chosen in an optimal way as discussed in Section 5.3, while the sample size n and the graph size a are kept constant. It can be seen that for both graph generators the utility decreases as the privacy parameter decreases when the partition size is kept constant.

In both cases, we observe that PSGG substantially outperforms the naive baseline method, producing two networks that fulfill privacy constraints while retaining high utility. We remark that the synthetic data example studied here is highly favorable for the naive baseline method as the binary choice of vertex attributes together with the product kernel yield deterministic edges, i.e. edges that exist with probability in $\{0, 1\}$. If the privacy parameter is not to small, a similar observation can be made for the network based on privatized data. In this case of (close to) deterministic edges, the PSGG algorithm based on joint sampling and the baseline approach based on independent sampling yield similar results. However, even in this baseline-favorable setting, PSGG outperforms the baseline, which demonstrates the strength of PSGG. In general (real-data) applications, we expect the performance gap between PSGG and the baseline method to be even larger. This expectation is supported by the simulation results for a real-data application in Section C.2.

Furthermore, as shown in Table 5 and Table 7, TV-PSMM shows improved LP runtime. This runtime improvement grows as the partition size m increases. The more efficient solution of the linear program in TV-PSMM ensures that the overall runtime of PSGG , despite its more involved joint network construction, is comparable to the overall runtime of the baseline method, which relies on a naive independent sampling of the networks.

C.2 Real data example: the 2011 Census in England & Wales

In this section we apply our graph generator to the 2011 Census Microdata Teaching file[1]. The file is freely available and contains 1% of the person records from the 2011 Census in England & Wales. It has around 570.000 entries with 15 attributes; an explanation of the attributes can be found under `https://www.ons.gov.uk/census/2011census/2011censusdata/censusmicrodata/microdatateachingfile/variablelist`. The data is already

[1]Source: Office for National Statistics licensed under the Open Government Licence v.1.0, `https://www.ons.gov.uk/census/2011census/2011censusdata/censusmicrodata/microdatateachingfile`

Table 4: Comparison of PSGG (Algorithm 3) and the baseline method for varying sample size n and optimal partition size $m = (\epsilon n)^{\frac{d}{d+1}}$ and optimal graph size $a = m^{\frac{2}{d}}$. The similarity of the generated networks is evaluated using the average FGW distance, as well as the similarity of their degree and closeness. Here, the degree and closeness distributions of two networks are each compared by the mmd kernel. We output mean and standard deviation; the smaller, the better.

n	ϵ	m	FGW		Degree		SD Degree		Closeness		SD Closeness	
			PSGG	PSMM	PSGG	PSMM	PSGG	PSMM	PSGG	PSMM	PSGG	PSMM
100	1	10	0.028	0.073	0.611	0.689	0.107	0.092	0.740	0.751	0.112	0.103
1000	0.1	10	0.028	0.074	0.611	0.679	0.104	0.097	0.735	0.748	0.110	0.093
10000	0.01	10	0.028	0.071	0.605	0.680	0.114	0.094	0.737	0.748	0.113	0.093

Table 5: Comparison of PSGG (Algorithm 3) and the baseline method for varying sample size n and optimal partition size $m = (\epsilon n)^{\frac{d}{d+1}}$ and optimal graph size $a = m^{\frac{2}{d}}$. We report the average runtime (in seconds) for both the linear program and the full algorithm.

n	ϵ	m	Total runtime		LP runtime	
			PSGG	PSMM	PSGG	PSMM
100	1	10	9.106	3.915	0.002	0.002
1000	0.1	10	19.206	15.527	0.001	0.002
10000	0.01	10	89.331	86.575	0.001	0.001

Table 6: Comparison of PSGG (Algorithm 3) and the baseline method for fixed sample size $n = 1000$, varying privacy parameter ϵ and optimal partition size $m = (\epsilon n)^{\frac{d}{d+1}}$ and fixed graph size $a = 100$. The similarity of the generated networks is evaluated using the average FGW distance, as well as the similarity of their degree and closeness. Here, the degree and closeness distributions of two networks are each compared by the mmd kernel and we output mean and standard deviation; the smaller, the better.

n	ϵ	m	FGW		Degree		SD Degree		Closeness		SD Closeness	
			PSGG	PSMM	PSGG	PSMM	PSGG	PSMM	PSGG	PSMM	PSGG	PSMM
1000	10	100	0.002	0.057	0.192	0.605	0.077	0.127	0.607	0.663	0.202	0.128
1000	1	32	0.009	0.060	0.402	0.612	0.114	0.103	0.702	0.709	0.151	0.103
1000	0.1	10	0.028	0.074	0.611	0.679	0.104	0.097	0.735	0.748	0.110	0.093
1000	0.01	3	0.091	0.159	0.671	0.739	0.087	0.072	0.701	0.760	0.103	0.082

Table 7: Comparison of the PSGG (Algorithm 3) and the baseline method for fixed sample size $n = 1000$, varying privacy parameter ϵ and optimal partition size $m = (\epsilon n)^{\frac{d}{d+1}}$ and fixed graph size $a = 100$. We report the average runtime (in seconds) for both the linear program and the full algorithm.

n	ϵ	m	Total runtime		LP runtime	
			PSGG	PSMM	PSGG	PSMM
1000	10	100	19.864	15.759	0.005	0.225
1000	1	32	19.567	15.257	0.002	0.011
1000	0.1	10	19.206	15.527	0.001	0.002
1000	0.01	3	18.336	15.820	0.001	0.001

classified as non-sensitive. Nevertheless, we apply our graph generator PSGG and the baseline method to compare them on real-world data. We focus on three attributes describing *age*, *health* and *economic activity*. We exclude all entries that contain the answer "no code required" and normalize the remaining attributes. This yields as attribute space $\Omega = [0,1]^3$ that we partition into m bins of equal length $m^{-1/3}$. Furthermore, we choose the partition size m and the expected graph size a, see Section 5.3, while varying the privacy parameter ϵ and the sample size n. In the following we analyze the (social) networks constructed from this complex data using the PSGG graph generator in Algorithm 3 with edge connection function $\kappa(x,y) := \frac{1}{3}(\kappa_{\mathrm{a}}(x_1,y_1) + \kappa_{\mathrm{a}}(x_2,y_2) + \kappa_{\mathrm{ea}}(x_3,y_3))$ for $x = (x_1,x_2,x_3) \in [0,1]^3$ and $y = (y_1,y_2,y_3) \in [0,1]^3$ as well as $\kappa_{\mathrm{a}}(x_1,y_1) := |x_1 - y_1|$, $\kappa_{\mathrm{h}}(x_2,y_2) = 1 - x_2 y_2$ and $\kappa_{\mathrm{ea}}(x_3,y_3) = |x_3 - y_3|$. This connection function reflects the intuition that the probability of connecting two

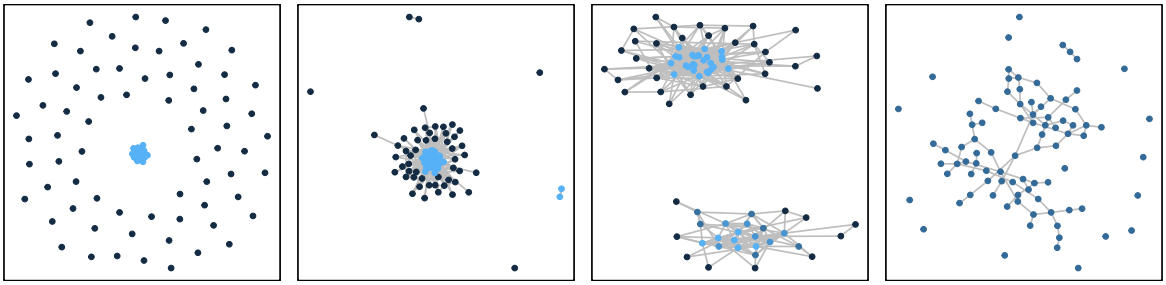

Figure 3: Representation of the input data the input set $\mathcal{X} = \{a_1, \ldots, a_n\}$ of size $n = 100$, containing the attributes $a_i = 0$ for $1 \leq i \leq n/2$ and $a_i = 1$ otherwise, at different levels of privacy obtained by (pairwise) joint realizations of network models of Chung-Lu type with edge probability function $\kappa(x, y) = xy$. Network 1: Realization of a network obtained from the input data $\mathcal{X}$. Network 2-4: Realizations of network obtained from the data in a differential private way with privacy parameter $\epsilon \in \{1, 0.1, 0.01\}$, attribute space $\Omega = [0, 1]$, expected network size $a = b = 100$, and number of cells in the partition $m = \lceil \sqrt{\epsilon n} \rceil$. The vertex color visualizes the attribute, where small attributes correspond to dark colors.

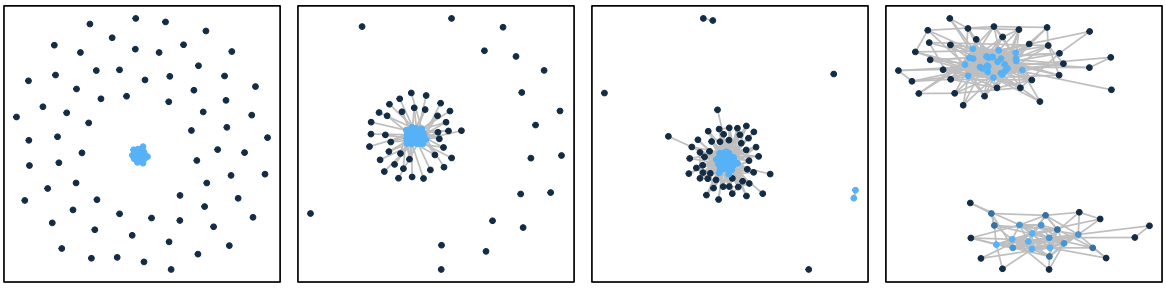

Figure 4: Representation of the input data $\mathcal{X} = \{a_1, \ldots, a_n\}$ of size $n = 1000$, containing the attributes $a_i = 0$ for $1 \leq i \leq n/2$ and $a_i = 1$ otherwise, at different levels of privacy obtained by (pairwise) joint realizations of network models of Chung-Lu type with edge probability function $\kappa(x, y) = xy$. Network 1: Realization of a network obtained from the input data $\mathcal{X}$. Network 2-4: Realizations of network obtained from the data in a differential private way with privacy parameter $\epsilon \in \{1, 0.1, 0.01\}$, attribute space $\Omega = [0, 1]$, expected network size $a = b = 100$, and number of cells in the partition $m = \lceil \sqrt{\epsilon n} \rceil$. The vertex color visualizes the attribute, where small attributes correspond to dark colors.

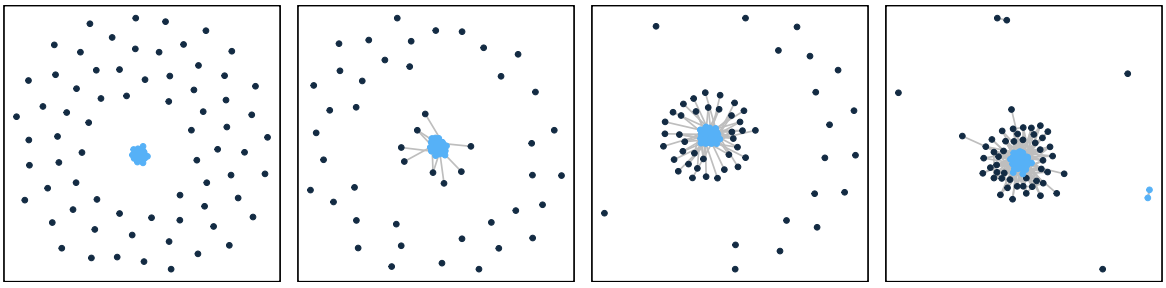

Figure 5: Representation of the input data $\mathcal{X} = \{a_1, \ldots, a_n\}$ of size $n = 10000$, containing the attributes $a_i = 0$ for $1 \leq i \leq n/2$ and $a_i = 1$ otherwise, at different levels of privacy obtained by (pairwise) joint realizations of network models of Chung-Lu type with edge probability function $\kappa(x, y) = xy$. Network 1: Realization of a network obtained from the input data $\mathcal{X}$. Network 2-4: Realizations of network obtained from the data in a differential private way with privacy parameter $\epsilon \in \{1, 0.1, 0.01\}$, attribute space $\Omega = [0, 1]$, expected network size $a = b = 100$, and number of sets in the partition $m = \lceil \sqrt{\epsilon n} \rceil$. The vertex color visualizes the attribute, where small attributes correspond to dark colors.

vertices increases when their ages are more similar (captured by the function κ_{a}), their health scores are lower, indicating better health status (captured by κ_{h}), and their economic activities are more similar (captured by κ_{ea}).

The results of the simulation study are presented in Table 8. For each graph-generation mechanism, utility is assessed by evaluating the similarity between the networks generated from the true data and from the privatized data. This similarity is measured in terms of their average network distance (FGW) and their degree and closeness similarity. Here, the degree and closeness distributions of two networks are each compared by the mmd kernel and we output mean and standard deviation. Additionally, Table 9 gives the average runtime for the linear program (contained in TV-PSMM and PSMM, respectively) and the whole graph generation mechanism. It can be observed that for both graph generators an decreasing privacy parameter ϵ leads to a decreased utility when the sample size n is fixed. As for the synthetic data example in Section C.1, PSGG outperforms PSMM based baseline approach, producing two networks that fulfill privacy constraints while retaining high utility. Furthermore, the runtime improvement of TV-PSMM over PSMM is evident when comparing the respective LP runtimes, see Table 9. This runtime improvement grows as the partition size m increases. The more efficient solution of the linear program in the TV-PSMM ensures that the overall runtime of PSGG , despite its more involved joint network construction, is comparable to the overall runtime of the baseline method, which relies on a naive independent sampling of the networks.

Table 8: Comparison of PSGG and the new baseline method for data input given by the 2011 Census Microdata Teaching file. We study the age, health and economic activity attributes of the 2011 Census Microdata Teaching file. We rescale the attributes to the attribute space $\Omega = [0,1]^3$ and choose a subsample of length $n = 4000$. The privacy mechanism in the baseline method and in PSGG (Algorithm 1) is applied with decreasing privacy parameter $\epsilon \in \{1, 0.1\}$. The edge kernel is chosen to be $\kappa(x,y) := \frac{1}{3}(\kappa_{\mathrm{a}}(x_1,y_1) + \kappa_{\mathrm{a}}(x_2,y_2) + \kappa_{\mathrm{ea}}(x_3,y_3))$ for $x = (x_1,x_2,x_3) \in [0,1]^3$ and $y = (y_1,y_2,y_3) \in [0,1]^3$, where $\kappa_{\mathrm{a}}(x_1,y_1) := |x_1 - y_1|$, $\kappa_{\mathrm{h}}(x_2,y_2) = 1 - x_2y_2$ and $\kappa_{\mathrm{ea}}(x_3,y_3) = |x_3 - y_3|$. The similarity of the generated networks is evaluated using the average FGW distance, as well as the similarity of their degree and closeness. Here, the degree and closeness distributions of two networks are each compared by the mmd kernel and we output mean and standard deviation; the smaller, the better.

n	ϵ	m	FGW		Degree		SD Degree		Closeness		SD Closeness	
			PSGG	PSMM	PSGG	PSMM	PSGG	PSMM	PSGG	PSMM	PSGG	PSMM
4000	1	512	0.025	0.202	0.135	0.286	0.030	0.118	0.117	0.579	0.034	0.221
4000	0.1	64	0.051	0.212	0.185	0.493	0.078	0.216	0.152	0.822	0.087	0.313

Table 9: Comparison of PSGG and the new baseline method for data input given by the 2011 Census Microdata Teaching file. We study the age, health and economic activity attributes of the 2011 Census Microdata Teaching file. We rescale the attributes to the attribute space $\Omega = [0,1]^3$ and choose a subsample of length $n = 4000$. The privacy mechanism in the baseline method and in PSGG (Algorithm 1) is applied with decreasing privacy parameter $\epsilon \in \{1, 0.1\}$. The edge kernel is chosen to be $\kappa(x,y) := \frac{1}{3}(\kappa_{\mathrm{a}}(x_1,y_1) + \kappa_{\mathrm{a}}(x_2,y_2) + \kappa_{\mathrm{ea}}(x_3,y_3))$ for $x = (x_1,x_2,x_3) \in [0,1]^3$ and $y = (y_1,y_2,y_3) \in [0,1]^3$, where $\kappa_{\mathrm{a}}(x_1,y_1) := |x_1 - y_1|$, $\kappa_{\mathrm{h}}(x_2,y_2) = 1 - x_2y_2$ and $\kappa_{\mathrm{ea}}(x_3,y_3) = |x_3 - y_3|$. We report the average runtime (in seconds) for both the linear program and the full algorithm.

n	ϵ	m	Total runtime		LP runtime	
			PSGG	PSMM	PSGG	PSMM
4000	1	512	164.280	190.654	0.161	32.927
4000	0.1	64	43.660	43.903	0.004	0.064

D PROOFS AND DETAILS OF SECTION 4

D.1 Discussion of the synthetic network model

The synthetic network model introduced in Section 4.1 is also known as random connection model (RCM), see Penrose (1991). Its network characteristics are inherited from the edge connection function, which uses the vertex attributes. In particular, the RCM can capture the intuition that similar vertices, i.e. vertices with similar vertex

attributes, are more likely to be connected than those with dissimilar vertex attributes and is thus an important tool to analyze properties of the vertex attributes. The RCM is used in several applications to analyze complex data, see for example Krioukov et al. (2010) and Penrose (2016). Furthermore, the study of the RCM is a very active research area in applied probability, see for example Duchemin and De Castro (2023). For example, in van der Hofstad et al. (2023) the connectivity for these models has been studied, while some theoretical results on the distribution of the number of components are given in Last et al. (2021).

Here we detail the point process representation of the synthetic network model introduced in Section 4.1. We first consider the synthetic network (H, T) constructed from the true data input. The number of vertices is given by a Poisson distributed random variable $N \sim \mathrm{Poi}(a)$, so that the expected size of the synthetic network is a. Each of these N vertices consists of an identifier $U_i \in [0,1]$ and an attribute $X_i \in \mathcal{X}$. The identifier U_i is chosen uniformly from $[0,1]$, while each attribute is chosen uniformly from the true data $\mathcal{X}$. Thus, each vertex (X_i, U_i) is unique almost surely (*a.s.*) as it consists of a (possibly) non-unique attribute X_i and a unique identifier U_i. The identifiers ensure that each (attributed) vertex is distinct and thus are essential for a well-defined edge structure. More precisely, we interpret $(X_i, U_i) \in \mathrm{H}$ as vertex U_i marked with attribute X_i and in particular draw edges between the identifiers U_i with a probability that dependents only on the attributes X_i. The identifier is a technical tool to derive a mathematical well-defined definition of a synthetic network. In particular, it should not impact the closeness of two synthetic networks. To account for this, we measure the distance of two marked points by their attribute distance, i.e. $d_{\Omega\times[0,1]}((x,u),(y,v)) := d_\Omega(x,y)$.

From a mathematical point of view, this construction can be expressed by a point process $\mathrm{H} = \sum_{i=1}^N \delta_{(X_i,U_i)}$. This point process is defined on the space $\mathcal{X}\times[0,1] \subset \Omega\times[0,1]$ and as the vertices (X_i,U_i) are unique, we can identify the point process H by its support $\mathrm{supp}(\mathrm{H}) = \{(X_1,U_1),\dots,(X_N,U_N)\}$. Furthermore, as N is Poisson distributed and the vertices (X_i,U_i) are chosen uniformly and independently of other vertices, H is a Poisson point process. The corresponding attribute intensity measure μ_n is given by $\mu_n = \frac{a}{n}\sum_{i=1}^n \delta_{x_i}$.

In a second step, i.e. after sampling the vertices of the synthetic network, we sample the edges based on the sampled vertex attributes. More precisely, given the vertices H, two attributed vertices $(X_i,U_i),(X_j,U_j) \in \mathrm{H}$ are connected with probability $\kappa(X_i,X_j)$ independently of other edges. In other words, for any possible edge $\{U_i,U_j\}$ we sample a Bernoulli distributed random variable $K_{ij}^{\mathrm{T}} \sim \mathrm{Ber}(\kappa(X_i,X_j))$ to determine whether this edge should be included in the graph. Again, this can be expressed using point processes. We represent the edge process T by a point process $\mathrm{T} = \sum_{\substack{i,j=1\\ i\neq j}}^N K_{ij}^{\mathrm{T}}\delta_{\{U_i,U_j\}}$ for (Bernoulli-distributed) $K_{ij}^{\mathrm{T}} \overset{i.i.d.}{\sim} \mathrm{Ber}(\kappa(X_i,X_j))$, $i,j \in [N]$.

The (synthetic) network (Ξ,Σ) with private data input is constructed in a similar way. Again, the network size is given by a Poisson distributed random variable $M \sim \mathrm{Poi}(b)$ for some $b>0$ and the identifiers V_i are chosen uniformly from $[0,1]$. To obtain a privatized network, the vertex attributes Y_i are now sampled from the set of representatives $\mathcal{Y} = \{y_1,\dots,y_m\} \subset \Omega$ with respect to the privatized probability measure $\hat{\nu}$ obtained from the TV-PSMM algorithm introduced in Section 3. This procedure again defines a Poisson point process $\Xi = \sum_{i=1}^M \delta_{(Y_i,V_i)}$ on the space $\mathcal{Y}\times[0,1]$ with attribute intensity measure $\hat{\nu}$. Given the vertices Ξ, we draw edges dependent on the (now privatized) vertex attributes Y_i. Analogous to the synthetic network based on the true data input, this can be represented by an edge process Σ by $\Sigma = \sum_{\substack{i,j=1\\ i\neq j}}^M K_{ij}^{\Sigma}\delta_{\{V_i,V_j\}}$, where $K_{ij}^{\Sigma} \overset{i.i.d.}{\sim} \mathrm{Ber}(\kappa(Y_i,Y_j))$ determines whether an edge $\{V_i,V_j\}$ is present or not.

D.2 Discussion of the assumptions

Here we comment on the assumptions in our network model; we clarify which assumptions are necessary and also how restrictive they are, and we outline possible relaxations.

- **Bounded attribute space:** The bounded attribute space is a standard assumption to control the utility and, for example, also made in (He et al., 2023). In real-world scenarios a finite number of attributes is observed, which allows to determine the extreme values and adapt the theoretical attribute space accordingly. Furthermore, when revealing the support is undesirable from a privacy perspective, the vertex attributes can be rescaled to a hypercube, thereby preventing sensitive information from being inferred from the support itself. Thus, the bounded attribute space is not a very restrictive assumption.

- **Existence of an edge connection function:** Our method uses the original data points as input, rather

than a graph derived from the data. Hence, we cannot estimate the edge connection function from the data. Instead, we assume that the kernel function is specified based on some "rough" prior knowledge on the data such as a homophily assumption in social networks, or proximity in a latent space. An example of this is given in the real-data example in Appendix C.2, where we assume that a similar age leads to higher probability of connection. In general, the kernel functions can be chosen to capture a wide range of interactions, for example by generating a generalized random dot product graph (see Athreya et al. (2018); Rubin-Delanchy et al. (2022)).

- **Lipschitz continuity:** The Lipschitz continuity of the edge connection function is not required for the network generator itself, but is introduced in Section 5 to derive utility guarantees. This Lipschitz assumption is often quite natural: As long as the data are in a metric space, Lipschitz assumptions make sense. In particular, networks are often constructed so that edge probabilities depend smoothly on points in a latent space; see Matias and Robin (2014a) for a survey. For example, the widely used (generalized) random dot product graphs take as kernel just the (perhaps weighted) product between the latent positions and are Lipschitz whenever the latent space is bounded. Hence the Lipschitz assumption is not unreasonable. Nevertheless, it should be possible to relax the general Lipschitz assumption to a separate Lipschitz assumption within each bin, possibly for the price of looser bounds and more involved proofs.

- **Additional assumptions made in Section 5:** Further assumptions on the diameter, partition size and expected network size that are introduced for parts of Section 5, are not required for the network generator nor for the theoretical guarantees. They are only applied in Corollary 5.2 and 5.4 as well as Section 5.3 to give concrete rates and to allow comparison with other privacy methods which have complex data input (not network input). The assumptions in the corollaries are standard assumptions and are often applied to exemplify the quality of the derived bounds (see for example He et al. (2023)).

D.3 Computational complexity of Algorithm 3

We would first like to clarify that the network generator does not involve the partition of the attribute space nor the computation of the FGW distance. The partition is provided as input to the algorithm together with the complex data. The approach of not including the partition itself is a standard approach (see, for example, (He et al., 2023), as it allows for a wide range of attributes and attribute spaces. Including the partition of the attribute space might restrict this generality. As a consequence, a partitioning of the space does not affect the computational complexity. Furthermore, our network generator does not rely on a computation of the FGW distance. Rather, the FGW distance is used as utility measure to evaluate the output. In practice, the derived theoretical bounds can be applied to completely avoid computation of the FGW distance.

Instead, the computational complexity of Algorithm 3 depends on the partition size m and the expected network sizes a, b. The private network generator consists of the TV-PSMM mechanism to derive privatized attributes and a network construction step to obtain the graphical representation. TV-PSMM requires the solution of a linear program with $2m$ variables and $m+1$ constraints. This linear program can be solved in polynomial time in m with small exponent, see for example van den Brand (2020), which is significantly faster than solving the linear program in PSMM that contains of m^2 variables and $m+1$ constraints (and thus can be solves in polynomial time in m^2), see He et al. (2023). The network sampling involves the generation of $\mathcal{O}(a^2+b^2)$ random variables. The other components of the generator have a lower computational complexity, and, for real-life examples, we expect the expected network size to dominate the rate. The runtime improvement of TV-PSMM over PSMM for solving the corresponding linear program is also observed in the simulation studies in Appendix C.

D.4 Proofs for Sections 3 and 4

For convenience we re-state the results here.

Proposition 3.3. *Algorithm 2 outputs a probability measure $\hat{\nu}$ on $\mathcal{Y}$ that is closest to the given discrete signed measure ν on $\mathcal{Y}$ with respect to the total variation distance.*

Proof. Let ν be a discrete signed probability measure on $\mathcal{Y}$. Then for any probability measure τ on $\mathcal{Y}$ the total variation distance between ν and τ is given by $d_{TV}(\nu,\tau) = \sum_{i=1}^{m} |\nu_i - \tau_i|$, where $\nu_i := \nu(\{y_i\})$ and $\tau_i := \tau(\{y_i\})$, see Section 2. Thus, finding a probability measure that is closest to ν with respect to the total variation distance

is equivalent to solving the program (P1) given by

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} |\nu_i - \tau_i| \\ \text{s.t.} \quad & \sum_{i=1}^{m} \tau_i = 1, \quad \tau_i \geq 0, \quad \forall i \leq m. \end{aligned} \tag{P1}$$

As $|\nu_i - \tau_i| = \max\{\nu_i - \tau_i, \tau_i - \nu_i\}$, minimizing the absolute value $|\nu_i - \tau_i|$ is equivalent to finding the minimal $u_i \geq \max\{\nu_i - \tau_i, \tau_i - \nu_i\}$. In particular, (P1) is equivalent to solving the linear program in Algorithm 2. □

Theorem 4.1. *The graphs* (Ξ, Σ) *and* (H, T) *obtained by Algorithm 3 have distributions* $P^{\mu_{\mathcal{Y}}^b} \otimes Q^{\kappa}$ *and* $P^{\mu_{\mathcal{X}}^a} \otimes Q^{\kappa}$*, respectively.*

Proof. We show that the two attributed random networks constructed in Algorithm 3 define a coupling of the graph models introduced in Section 4.1. We write $\mathbb{P}_\lambda$ and $\mathbb{E}_\lambda$ to denote the conditional probability and expectation given $\lambda = (\lambda_i)_{i=1}^m$. Recall that the probability measure $\hat{\nu}$ depends on λ. We write (H, T) and (Ξ, Σ) for the graphs constructed according to the graph model in Section 4.1 and $(\hat{\mathrm{H}}, \hat{\mathrm{T}})$ as well as $(\hat{\Xi}, \hat{\Sigma})$ for the graphs constructed in Algorithm 3.

Step 1: Coupling of the vertex process
Consider the total number of vertices $N \sim \mathrm{Poi}(a)$ and $M \sim \mathrm{Poi}(b)$ and assume without loss of generality that $a \geq b$. Then $N \stackrel{d}{=} M + K_N$ for $K_N \sim \mathrm{Poi}(a - b)$. In particular, this defines a coupling of N and M such that $N \geq M$.

For $k \in [m]$, let $N_k := \mathrm{H}(\Omega_k)$ be the number of points (X_i, U_i) of H whose attributes X_i are in Ω_k. As the X_i were chosen uniformly from $\mathcal{X}$ we obtain

$$\mathbb{P}(X_i \in \Omega_k) = \frac{\#\{x_j \in \Omega_k : j \in [n]\}}{n} = \frac{n_k}{n}.$$

Given the total number of points N in η, the vector $\mathbb{N} := (N_1, \ldots, N_m)$ of attribute counts in the partition sets $(\Omega_i)_{i\in[m]}$ has a multinomial distribution, $\mathbb{N} \,|\, N \sim \mathrm{Multin}(N, (\frac{n_1}{n}, \ldots, \frac{n_m}{n}))$. Analogously, for $k \in [m]$ let $M_k := \Xi(\Omega_k)$ be the number of points (Y_i, V_i) of Ξ whose attributes Y_i are in Ω_k. Then we have

$$\mathbb{P}_\lambda(Y_i \in \Omega_k) = \mathbb{P}(Y_i = y_k) = \hat{\nu}_k$$

and, given the total number of points M in Ξ, we obtain that the conditional distribution $\mathbb{M} \,|\, M$ of the vector $\mathbb{M} = (M_1, \ldots, M_m)$ is given by $(M_1, \ldots, M_m) \,|\, M \sim \mathrm{Multin}(M, (\hat{\nu}_1, \ldots, \hat{\nu}_m))$.

We show that the vertex counts constructed in Algorithm 3 obtain the same multinomial distribution, by separately considering the counts of common vertices and the counts of vertices which are not shared by the two processes. Let $\mathbb{Z}^j := (Z_1^j, \ldots, Z_m^j) \in \{0,1\}^m$, $j \in [M]$, be independent random variables defined by

$$\mathbb{P}_\lambda(\mathbb{Z}^j = (l_1, \ldots, l_m)) = \prod_{k\in[m]} \left(\frac{n_k}{n} \wedge \hat{\nu}_k\right)^{l_k}, \quad \text{for } l_1, \ldots, l_m \in \{0,1\} \text{ such that } \sum_{k\in[m]} l_k = 1;$$

$$\mathbb{P}_\lambda(\mathbb{Z}^j = (0, \ldots, 0)) = 1 - \sum_{\substack{l_1,\ldots,l_m \in \{0,1\} \\ \sum_{k\in[m]} l_k = 1}} \mathbb{P}_\lambda(\mathbb{Z}^j = (l_1, \ldots, l_m)) = 1 - \sum_{k\in[m]} \frac{n_k}{n} \wedge \hat{\nu}_k.$$

Note that this construction corresponds to the choice of common vertex counts in Algorithm 3.

Next we describe the construction of non-common vertex counts. First, define $\mathbb{K} \,|\, (N - M) := (K_1, \cdots, K_m) \,|\, (N - M) \sim \mathrm{Multin}(N - M, (n_1/n, \ldots, n_m/n))$. If $\frac{n_k}{n} = \hat{\nu}_k$ for all $k \in [m]$, then $\mathbb{N} \stackrel{d}{=} \mathbb{M} + \mathbb{K}$ and we obtain a coupling $\mathbb{N}$ and $\mathbb{M}$ such that $N_k = M_k + K_k = \sum_{j\in[M]} Z_k^j + K_k$ for all $k \in [m]$. Otherwise, there exists a $k \in [m]$ such that $\frac{n_k}{n} \neq \hat{\nu}_k$ and we obtain $\mathbb{P}_\lambda(\mathbb{Z}^j = (0, \ldots, 0)) > 0$. For $j \in [M]$, we define $\mathbb{Z}^{N,j} := (Z_1^{N,j}, \ldots, Z_m^{N,j})$ by

$$\mathbb{P}_\lambda(\mathbb{Z}^{N,j} = (l_1, \ldots, l_m)) = \big(\mathbb{P}_\lambda(\mathbb{N}^j = (l_1, \ldots, l_m)) - \mathbb{P}_\lambda(\mathbb{Z}^j = (l_1, \ldots, l_m))\big)\mathbb{P}_\lambda(\mathbb{Z}^j = (0, \ldots, 0))^{-1},$$

where $\mathbb{N}^j \sim \text{Multin}(1, (\frac{n_1}{n}, \dots, \frac{n_m}{n}))$ and $l_1, \dots, l_m \in \{0,1\}$ such that $\sum_{k\in[m]} l_k = 1$. Note that this defines a probability distribution as

$$\sum_{\substack{l_1,\dots,l_m\in\{0,1\}\\ \sum_{k\in[m]} l_k=1}} \mathbb{P}_\lambda(\mathbb{Z}^{N,j} = (l_1,\dots,l_m)) = \left(1 - \sum_{\substack{l_1,\dots,l_m\in\{0,1\}\\ \sum_{k\in[m]} l_k=1}} \mathbb{P}_\lambda(\mathbb{Z}^j = (l_1,\dots,l_m))\right)\mathbb{P}_\lambda(\mathbb{Z}^j = (0,\dots,0))^{-1} = 1.$$

Furthermore, $\mathbb{Z}^{N,j} = (l_1,\dots,l_m) \,|\, (\mathbb{Z}^j = (0,\dots,0)) \sim \text{Multin}(1, (\frac{n_1}{n}, \dots, \frac{n_m}{n}))$ and thus $(Z_1^{N,j} + K_1, \dots, Z_m^{N,j} + K_m) \sim \text{Multin}(N - Z, (\frac{n_1}{n}, \dots, \frac{n_m}{n}))$ for $Z = \sum_{j\in[M]} \sum_{k\in[m]} Z_k^j$. In particular, the construction of $\mathbb{Z}^{N,j}$ and $\mathbb{K}$ corresponds to the creation of non-common vertex counts in Algorithm 3. Analogously we define $\mathbb{Z}^{M,j}$ and set $\hat{\mathbb{N}} = \sum_{j\in[M]} \hat{\mathbb{N}}^j + \sum_{j\in[N]\setminus[M]} \mathbb{N}^j = \sum_{j\in[M]} \mathbb{Z}^j + \mathbb{Z}^{N,j}\mathbb{1}_{\{\mathbb{Z}^j=(0,\dots,0)\}} + \mathbb{L}$ as well as $\hat{\mathbb{M}} = \sum_{j\in[M]} \hat{\mathbb{M}}^j = \sum_{j\in[M]} \mathbb{Z}^j + \mathbb{Z}^{M,j}\mathbb{1}_{\{\mathbb{Z}^j=(0,\dots,0)\}}$. Then $(\hat{\mathbb{N}}, \hat{\mathbb{M}})$ is a coupling of $\mathbb{N}$ and $\mathbb{M}$ as for any $j \in [M]$

$$\begin{aligned}
&\mathbb{P}_\lambda(\hat{\mathbb{N}}^j = (l_1,\dots,l_m))\\
&= \mathbb{P}_\lambda(\hat{\mathbb{N}}^j = (l_1,\dots,l_m), \mathbb{Z}^j = (0,\dots,0)) + \mathbb{P}_\lambda(\hat{\mathbb{N}}^j = (l_1,\dots,l_m), \mathbb{Z}^j \neq (0,\dots,0))\\
&= \mathbb{P}_\lambda(\mathbb{Z}^{N,j} = (l_1,\dots,l_m))\mathbb{P}_\lambda(\mathbb{Z}^j = (0,\dots,0)) + \mathbb{P}_\lambda(\mathbb{Z}^j = (l_1,\dots,l_m))\\
&= \mathbb{P}_\lambda(\mathbb{N}^j = (l_1,\dots,l_m))
\end{aligned}$$

and analogously $\mathbb{P}_\lambda(\hat{\mathbb{M}}^j = (l_1,\dots,l_m)) = \mathbb{P}_\lambda(\mathbb{M}^j = (l_1,\dots,l_m))$. Furthermore, by the construction of $\mathbb{Z}$ we obtain for any $j \in [M]$ that

$$\mathbb{P}_\lambda(\hat{\mathbb{N}}^j \neq \hat{\mathbb{M}}^j) = \mathbb{P}_\lambda(\mathbb{Z}^j = (0,\dots,0)).$$

We now use the fact that the vertex counts in Algorithm 3 have the correct marginal distributions to follow that the vertex processes $\hat{\mathrm{H}}$ and $\hat{\Xi}$ have the correct marginal distributions. Recall that we constructed $\hat{\mathrm{H}} = \sum_{i=1}^m \sum_{j=1}^{\hat{N}_i} \delta_{(X_{ij},U_{ij})}$ for $X_{ij} \sim \mathcal{U}(\mathcal{X} \cap \Omega_i)$, $U_{ij} \sim \mathcal{U}([0,1])$ and $\hat{\Xi} = \sum_{i=1}^m \sum_{j=1}^{\hat{M}_i} \delta_{(Y_{ij},U_{ij})}$ for $Y_{ij} = y_i$. Note that the distributions of the attributes, the identifiers, and the number of attributes in each partition set are unchanged compared to the graph model in Section 4.1. Thus, we have shown that the vertex counts constructed in Algorithm 3 have the same distribution. Hence, this construction defines a coupling $(\hat{\mathrm{H}}, \hat{\Xi})$ of the vertex processes H and Ξ.

Step 2: Coupling of the edge process
Based on the coupling of the vertex processes we now show that the edge processes $(\hat{\mathrm{T}}, \hat{\Sigma})$ constructed in Algorithm 3 define a coupling of the edge processes T and Σ. Recall that the existence of an edge $(U_{ij}, U_{i'j'})$ in T and Σ is determined by Bernoulli random variables K_{ij}^{T} and K_{ij}^{Σ}, respectively, see Section 4.1. Furthermore, the connection probabilities in Algorithm 3 are given by

$$\begin{aligned}
\mathbb{P}_\lambda((\hat{K}^{\mathrm{T}}_{iji'j'}, \hat{K}^{\Sigma}_{iji'j'}) = (1,1)) &= \kappa(X_{ij}, X_{i'j'}) \wedge \kappa(Y_{ij}, Y_{i'j'})\\
\mathbb{P}_\lambda((\hat{K}^{\mathrm{T}}_{iji'j'}, \hat{K}^{\Sigma}_{iji'j'}) = (1,0)) &= \kappa(X_{ij}, X_{i'j'}) - (\kappa(X_{ij}, X_{i'j'}) \wedge \kappa(Y_{ij}, Y_{i'j'}))\\
\mathbb{P}_\lambda((\hat{K}^{\mathrm{T}}_{iji'j'}, \hat{K}^{\Sigma}_{iji'j'}) = (0,1)) &= \kappa(Y_{ij}, Y_{i'j'}) - (\kappa(X_{ij}, X_{i'j'}) \wedge \kappa(Y_{ij}, Y_{i'j'}))\\
\mathbb{P}_\lambda((\hat{K}^{\mathrm{T}}_{iji'j'}, \hat{K}^{\Sigma}_{iji'j'}) = (0,0)) &= 1 - \kappa(X_{ij}, X_{i'j'}) \vee \kappa(Y_{ij}, Y_{i'j'})
\end{aligned}$$

for any $i, i' \in [m]$ and any $j \in [Z_i]$ and $j' \in [Z_{i'}]$ and by $\hat{K}^{\mathrm{T}}_{iji'j'} \sim \text{Ber}(\kappa(X_{ij}, X_{i'j'}))$ and $\hat{K}^{\Sigma}_{iji'j'} \sim \text{Ber}(\kappa(Y_{ij}, Y_{i'j'}))$ for any (j, j') such that $j > Z_i$ or $j' > Z_{j'}$. This construction of $\hat{K}^{\mathrm{T}}$ and $\hat{K}^{\Sigma}$ corresponds to a (maximal) coupling of the two Bernoulli random variables K_{ij}^{T} and K_{ij}^{Σ}. A *maximal* coupling between two distributions p and q is a coupling of a pair of random variables (X, Y) that maximizes $\mathbb{P}(X = Y)$. In particular,

$$\mathbb{P}_\lambda(\hat{K}^{\mathrm{T}}_{iji'j'} \neq \hat{K}^{\Sigma}_{iji'j'}) = |\kappa(X_{ij}, X_{i'j'}) - \kappa(Y_{ij}, Y_{i'j'})|.$$

Thus, setting

$$\hat{\mathrm{T}} = \sum_{i,i'=1}^m \sum_{j=1}^{\hat{N}_i} \sum_{j'=1}^{\hat{N}_{i'}} \hat{K}^{\mathrm{T}}_{iji'j'} \delta_{\{U_{ij},U_{i'j'}\}} \text{ and } \hat{\Sigma} = \sum_{i,i'=1}^m \sum_{j=1}^{\hat{M}_i} \sum_{j'=1}^{\hat{M}_{i'}} \hat{K}^{\Sigma}_{iji'j'} \delta_{\{U_{ij},U_{i'j'}\}}$$

yields a coupling of the edge processes. Overall, this shows that $(\hat{\mathrm{H}}, \hat{\mathrm{T}})$ and $(\hat{\Xi}, \hat{\Sigma})$ constructed in Algorithm 3 yield a coupling of (H, T) and (Ξ, Σ) constructed in Section4.1 and thus have the claimed (marginal) distributions. □

E PROOFS AND DETAILS FOR SECTION 5

For convenience we re-state the results here.

Theorem 5.1. *Assume that κ is Lipschitz continuous with Lipschitz constant L_κ in both components and let (H,T) and (Ξ,Σ) be the networks with true and private data input of expected size $a>0$ and $b>0$, respectively, constructed by Algorithm 3. Then, using the FGW distance with parameter $\alpha\in[0,1]$,*

$$\begin{aligned}&\mathbb{E}\big(d_{FGW}(\eta^{(\mathrm{H},\mathrm{T})},\eta^{(\Xi,\Sigma)})\big)\\&\le \tilde{C}_1 \max_{k\in[m]}\mathrm{diam}(\Omega_k)\frac{1}{1+\frac{a\wedge b}{|a-b|}\mathbb{1}_{\{|a-b|>0\}}}\\&\quad+4\tilde{C}_2\frac{m}{n}\mathbb{E}(|\Lambda|)+\tilde{C}_2\bigg(1-\frac{1}{\big(1+\frac{a\wedge b}{|a-b|}\big)^2}\bigg)\mathbb{1}_{\{|a-b|>0\}},\end{aligned}$$

where $\tilde{C}_1=1-\alpha+\alpha(2\,C\,L_\kappa)$ and

$$\tilde{C}_2=(1-\alpha)\mathrm{diam}(\Omega)+\alpha\min\{C,2\,C\,L_\kappa\mathrm{diam}(\Omega)\},$$

with $C>0$ given in (5).

Proof. We write $\mathbb{P}_\lambda$ and $\mathbb{E}_\lambda$ to denote the conditional probability and expectation given $\lambda=(\lambda_i)_{i=1}^m$ and assume w.l.o.g. that $a\ge b$. We keep the notation introduced in the proof of Theorem 4.1. In particular, we denote the networks constructed with Algorithm 3 by $(\hat{\mathrm{H}},\hat{\mathrm{T}})$ and $(\hat{\Xi},\hat{\Sigma})$ with

$$\hat{\mathrm{H}}=\sum_{i=1}^m\sum_{j=1}^{\hat{N}_i}\delta_{(X_{ij},U_{ij})}\quad\text{and}\quad\hat{\Xi}=\sum_{i=1}^m\sum_{j=1}^{\hat{M}_i}\delta_{(Y_{ij},U_{ij})}$$

as well as

$$\hat{\mathrm{T}}=\sum_{i,i'=1}^m\sum_{j=1}^{\hat{N}_i}\sum_{j'=1}^{\hat{N}_{i'}}\hat{K}^{\mathrm{T}}_{iji'j'}\delta_{\{U_{ij},U_{i'j'}\}}\quad\text{and}\quad\hat{\Sigma}=\sum_{i,i'=1}^m\sum_{j=1}^{\hat{M}_i}\sum_{j'=1}^{\hat{M}_{i'}}\hat{K}^{\Sigma}_{iji'j'}\delta_{\{U_{ij},U_{i'j'}\}}.$$

Step 1: Derivation of the upper bound
We use the joint construction of the vertex- and edge processes to calculate the expected distance of the two graphs with respect to the FGW distance, see Section 2.1. Recall that we couple $|\hat{\mathrm{H}}|=N$ and $|\hat{\Xi}|=M$ such that $N=M+K$ for $K\sim\mathrm{Poi}(a-b)$. By conditioning on the (minimal) number of vertices (X_j,Y_j) that are in the same cell, we obtain

$$\begin{aligned}\mathbb{E}_\lambda(d_{FGW}(\eta^{(\mathrm{H},\mathrm{T})},\eta^{(\Xi,\Sigma)}))&=\mathbb{E}_\lambda\big[\mathbb{E}_\lambda(d_{FGW}(\eta^{(\hat{\mathrm{H}},\hat{\mathrm{T}})},\eta^{(\hat{\Xi},\hat{\Sigma})})\,|\,N,M)\big]\\&=\mathbb{E}_\lambda\bigg[\sum_{l=0}^M\mathbb{E}_\lambda(d_{FGW}(\eta^{(\hat{\mathrm{H}},\hat{\mathrm{T}})},\eta^{(\hat{\Xi},\hat{\Sigma})})\,\Big|\sum_{j\in[M]}\|\mathbb{Z}^j\|=l,N,M)\\&\qquad\binom{M}{l}\mathbb{P}_\lambda(\mathbb{Z}^1\neq(0,\dots,0))^l\,\mathbb{P}_\lambda(\mathbb{Z}^1=(0,\dots,0))^{(M-l)}\bigg].\end{aligned}\tag{10}$$

Let $(X_{ij},U_{ij}),(X_{i'j'},U_{i'j'})\in\hat{\mathrm{H}}$ and $(Y_{kl},U_{kl}),(Y_{k'l'},U_{k'l'})\in\hat{\Xi}$. Then

$$d((X_{ij},U_{ij}),(Y_{kl},U_{kl})):=d(X_{ij},Y_{kl})\le\mathrm{diam}(\Omega)$$

and

$$\begin{aligned}|d_{\hat{\mathrm{T}}}(U_{ij},U_{i'j'})-d_{\hat{\Sigma}}(U_{kl},U_{k'l'})|\le C\,\mathbb{P}(\hat{K}^{\mathrm{T}}_{iji'j'}\neq\hat{K}^{\Sigma}_{klk'l'})&=\min\{C,C\,|\kappa(X_{ij},X_{i'j'})-\kappa(Y_{kl},Y_{k'l'})|\}\\&\le\min\{C,2\,C\,L_\kappa\mathrm{diam}(\Omega)\},\end{aligned}$$

where the last inequality follows as the Lipschitz continuity of κ implies

$$\begin{aligned}|\kappa(x,x')-\kappa(y,y')|&\le|\kappa(x,x')-\kappa(y,y')|+|\kappa(y,x')-\kappa(y,y')|\\&\le L_\kappa\,(d(x,y)+d(x',y')).\end{aligned}\tag{11}$$

If $i = k$ and $i' = k'$ we have by construction $X_{ij}, Y_{il} \in \Omega_i$ and $X'_{i'j'}, Y_{i'l'} \in \Omega_{i'}$. Hence, we can improve the above bounds to

$$d(X_{ij}, Y_{il}) \leq \operatorname{diam}(\Omega_i) \leq \max_{k\in[m]} \operatorname{diam}(\Omega_k)$$

and

$$|d_{\hat{\mathrm{T}}}(U_{ij}, U_{i'j'}) - d_{\hat{\Sigma}}(U_{il}, U_{i'l'})| \leq C\,\mathbb{P}(\hat{K}^{\mathrm{T}}_{iji'j'} \neq \hat{K}^{\Sigma}_{ili'l'}) \leq 2\,C\,L_\kappa \max_{k\in[m]} \operatorname{diam}(\Omega_k).$$

Now consider the conditional expectation in Equation (10). As $\sum_{j\in[M]} \|\mathbb{Z}^j\| = l$, i.e. $\mathbb{Z}^j \neq (0,\dots,0)$ for l many $\mathbb{Z}^j$, we have that $\sum_{k\in[m]} Z_k = \sum_{k\in[m]} \sum_{j\in[M} Z^j_k = l$. In particular, we have at least l points of $\hat{\mathrm{H}}$ and $\hat{\Xi}$ that have attributes in the same partition set and $N-l$ and $M-l$ points, respectively, that have attributes in different partition sets. We calculate an upper bound of the FGW distance by restricting to "transport plans" that move mass $1/N$ from vertex X_{ij} (having mass $1/N$) to vertex Y_{ij} (having mass $1/M$) for all $i \in [m]$ and $j \in [Z_i]$ and distribute the remaining mass $(1 - l/N)$ according to some coupling of the remaining vertices. Then

$$\begin{aligned}
&\mathbb{E}_\lambda\Big(d_{FGW}(\eta^{(\hat{\mathrm{H}},\hat{\mathrm{T}})}, \eta^{(\hat{\Xi},\hat{\Sigma})}) \,\Big|\, \sum_{j\in[M]} \|\mathbb{Z}^j\| = l, N, M\Big) \\
&\leq \sum_{i,i'=1}^{m} \sum_{j=1}^{Z_i} \sum_{j=1}^{Z_{i'}} \mathbb{E}_\lambda\big[\big((1-\alpha)d(X_{ij}, Y_{ij}) + \alpha|d_{\hat{\mathrm{T}}}(U_{ij}, U_{i'j'}) - d_{\hat{\Sigma}}(U_{ij}, U_{i'j'})|\big)\big]\frac{1}{N^2} \\
&\quad + \frac{N^2 - l^2}{N^2}\big((1-\alpha)\operatorname{diam}(\Omega) + \alpha\min\{C, 2\,C\,L_\kappa \operatorname{diam}(\Omega)\}\big) \\
&\leq \frac{l^2}{N^2}\big(1 - \alpha + \alpha(2\,C\,L_\kappa)\big)\big(\max_{k\in[m]} \operatorname{diam}(\Omega_k)\big) + \frac{N^2 - l^2}{N^2}\big((1-\alpha)\operatorname{diam}(\Omega) + \alpha\min\{C, 2\,C\,L_\kappa \operatorname{diam}(\Omega)\}\big),
\end{aligned}$$

where we used the above bounds, applying the special case to the first term and the general case to the second term.

Recall that by the construction of $\mathbb{Z}^1$ we have $\mathbb{P}_\lambda(\mathbb{Z}^1 \neq (0,\dots,0)) = \sum_{k\in[m]} \left(\frac{n_k}{n} \wedge \hat{\nu}_k\right)$. We set $\tilde{C}_1 := \big(1 - \alpha + \alpha(2\,C\,L_\kappa)\big)$ and $\tilde{C}_2 := (1-\alpha)\operatorname{diam}(\Omega) + \alpha\min\{C, 2\,C\,L_\kappa \operatorname{diam}(\Omega)\}$. Given M, we introduce a random variable L having conditional distribution $L \,|\, M \sim \mathrm{Bin}(M, \mathbb{P}_\lambda(\mathbb{Z}^1 \neq (0,\dots,0)))$. Taking the expectation of λ and using Equation (10) yields

$$\begin{aligned}
&\mathbb{E}(d_{FGW}(\eta^{(\hat{\mathrm{H}},\hat{\mathrm{T}})}, \eta^{(\hat{\Xi},\hat{\Sigma})})) = \mathbb{E}\Big[\mathbb{E}_\lambda(d_{FGW}(\eta^{(\hat{\mathrm{H}},\hat{\mathrm{T}})}, \eta^{(\hat{\Xi},\hat{\Sigma})}))\Big] \\
&= \mathbb{E}\Bigg[\mathbb{E}_\lambda\Bigg[\sum_{l=0}^{M} \mathbb{E}_\lambda(d_{FGW}(\eta^{(\hat{\mathrm{H}},\hat{\mathrm{T}})}, \eta^{(\hat{\Xi},\hat{\Sigma})}) \,|\, \sum_{j\in[M]} \|\mathbb{Z}^j\| = l, N, M) \\
&\qquad\qquad \binom{M}{l} \mathbb{P}_\lambda(\mathbb{Z}^1 \neq (0,\dots,0))^l\, \mathbb{P}_\lambda(\mathbb{Z}^1 = (0,\dots,0))^{(M-l)}\Bigg]\Bigg] \\
&\leq \mathbb{E}\Bigg[\mathbb{E}_\lambda\Bigg[\sum_{l=0}^{M} \binom{M}{l}\Big(\frac{l^2}{N^2}\tilde{C}_1\big(\max_{k\in[m]} \operatorname{diam}(\Omega_k)\big) + \frac{N^2 - l^2}{N^2}\tilde{C}_2\Big) \\
&\qquad\qquad \mathbb{P}_\lambda(\mathbb{Z}^1 \neq (0,\dots,0))^l\, \mathbb{P}_\lambda(\mathbb{Z}^1 = (0,\dots,0))^{(N-l)}\Bigg]\Bigg] \\
&= \mathbb{E}\Bigg[\mathbb{E}_\lambda\Bigg[\tilde{C}_1\big(\max_{k\in[m]} \operatorname{diam}(\Omega_k)\big)\frac{\mathbb{E}_\lambda(L^2 \,|\, M)}{N^2} + \tilde{C}_2\Big(1 - \frac{\mathbb{E}_\lambda(L^2 \,|\, M)}{N^2}\Big)\Bigg]\Bigg].
\end{aligned}$$

We now use that

$$\mathbb{E}_\lambda(L^2 \,|\, M) = M\mathbb{P}_\lambda(\mathbb{Z}^1 \neq (0,\dots,0)) + (M^2 - M)\mathbb{P}_\lambda(\mathbb{Z}^1 \neq (0,\dots,0))^2$$

to obtain that the above expression equals

$$\begin{aligned}
&\mathbb{E}\Big[\mathbb{E}_\lambda\Big[\tilde{C}_1\big(\max_{k\in[m]}\operatorname{diam}(\Omega_k)\big)\Big[\frac{M}{N^2}\mathbb{P}_\lambda(\mathbb{Z}^1\neq(0,\dots,0))+\frac{M^2-M}{N^2}\mathbb{P}_\lambda(\mathbb{Z}^1\neq(0,\dots,0))^2\Big]\\
&\quad+\tilde{C}_2\Big[1-\frac{M}{N^2}\mathbb{P}_\lambda(\mathbb{Z}^1\neq(0,\dots,0))-\frac{M^2-M}{N^2}\mathbb{P}_\lambda(\mathbb{Z}^1\neq(0,\dots,0))^2\Big]\Big]\Big]\\
&\leq\mathbb{E}\Big[\mathbb{E}_\lambda\Big[\tilde{C}_1\big(\max_{k\in[m]}\operatorname{diam}(\Omega_k)\big)\Big[\frac{M}{N^2}\mathbb{P}_\lambda(\mathbb{Z}^1\neq(0,\dots,0))+\frac{M^2-M}{N^2}\mathbb{P}_\lambda(\mathbb{Z}^1\neq(0,\dots,0))\Big]\\
&\quad+\tilde{C}_2\Big[\frac{M}{N^2}\Big(1-\mathbb{P}_\lambda(\mathbb{Z}^1\neq(0,\dots,0))\Big)+\frac{N^2-M}{N^2}\Big(1^2-\mathbb{P}_\lambda(\mathbb{Z}^1\neq(0,\dots,0))^2\Big)\\
&\qquad\qquad-\frac{M^2-N^2}{N^2}\mathbb{P}_\lambda(\mathbb{Z}^1\neq(0,\dots,0))^2\Big]\Big]\Big]\\
&\leq\mathbb{E}\Big[\mathbb{E}_\lambda\Big[\tilde{C}_1\big(\max_{k\in[m]}\operatorname{diam}(\Omega_k)\big)\frac{M^2}{N^2}\sum_{k\in[m]}\Big(\frac{n_k}{n}\wedge\hat{\nu}_k\Big)\\
&\quad+\tilde{C}_2\Big[\frac{M}{N^2}\Big(1-\sum_{k\in[m]}\Big(\frac{n_k}{n}\wedge\hat{\nu}_k\Big)\Big)+2\Big(1-\frac{M}{N^2}\Big)\Big(1-\sum_{k\in[m]}\Big(\frac{n_k}{n}\wedge\hat{\nu}_k\Big)\Big)\\
&\qquad\qquad+\Big(1-\frac{M^2}{N^2}\Big)\Big(\sum_{k\in[m]}\Big(\frac{n_k}{n}\wedge\hat{\nu}_k\Big)\Big)^2\Big]\Big]\Big]\\
&\leq\mathbb{E}\Big[\tilde{C}_1\big(\max_{k\in[m]}\operatorname{diam}(\Omega_k)\big)\mathbb{E}\Big[\frac{M^2}{N^2}\Big]+\tilde{C}_2\mathbb{E}\Big[2-\frac{M}{N^2}\Big]\Big(1-\sum_{k\in[m]}\Big(\frac{n_k}{n}\wedge\hat{\nu}_k\Big)\Big)+\tilde{C}_2\mathbb{E}\Big[1-\frac{M^2}{N^2}\Big]\Big].
\end{aligned}$$

The second-to-last inequality follows as $(a^2-b^2)=(a-b)(a+b)\leq 2\max\{a,b\}(a-b)$ while the last inequality uses that N and M are independent of λ. Noting that $1=\sum_{k\in[m]}\frac{n_k}{n}$, we obtain

$$\mathbb{E}\Big[1-\sum_{k\in[m]}\frac{n_k}{n}\wedge\hat{\nu}_k\Big]=\mathbb{E}\Big[\sum_{k\in[m]}\frac{n_k}{n}-\big(\frac{n_k}{n}\wedge\hat{\nu}_k\big)\Big]=\mathbb{E}\Big[\sum_{k\in[m]}\big(\frac{n_k}{n}-\hat{\nu}_k\big)_+\Big]\leq\mathbb{E}\Big[\sum_{k\in[m]}|\frac{n_k}{n}-\hat{\nu}_k|\Big].$$

As the probability measure $\hat{\nu}$ was chosen as closest to ν with respect to the total variation metric, we have

$$\begin{aligned}
\mathbb{E}\Big[\sum_{k\in[m]}|\frac{n_k}{n}-\hat{\nu}_k|\Big]&\leq\mathbb{E}\Big[\sum_{k\in[m]}|\frac{n_k}{n}-\frac{n_k+\lambda_k}{n}|+\sum_{k\in[m]}|\frac{n_k+\lambda_k}{n}-\hat{\nu}_k|\Big]\\
&\leq 2\sum_{k\in[m]}\mathbb{E}\Big[|\frac{n_k}{n}-\frac{n_k+\lambda_k}{n}|\Big]\\
&=\frac{2}{n}\sum_{k\in[m]}\mathbb{E}[|\lambda_k|]=2\frac{m}{n}\mathbb{E}(|\Lambda|).
\end{aligned}$$

Step 2: Calculation of Poisson Expectations
It remains to calculate the expectations in the upper bound. Recall that we couple $N\sim\operatorname{Poi}(a)$ and $M\sim\operatorname{Poi}(b)$ such that $N{=}M+L$ for $L\sim\operatorname{Poi}(c)$ with $c=a-b$. In particular, N and M do not depend of λ.

For $c > 0$ we obtain

$$\begin{aligned}\mathbb{E}\Big[\frac{M}{M+L}\Big] &= \sum_{l=0}^{\infty} e^{-c}\frac{c^l}{l!}\sum_{m=1}^{\infty} e^{-b}\frac{b^m}{m!}\frac{m}{m+l}\\ &= \sum_{l=0}^{\infty} e^{-c}\frac{c^l}{l!}\sum_{m=1}^{\infty} e^{-b}\frac{b^m}{(m-1)!}\frac{1}{m+l}\\ &= b\sum_{l=0}^{\infty} e^{-c}\frac{c^l}{l!}\,\mathbb{E}\Big[\frac{1}{M+1+l}\Big]\\ &= \frac{b}{c}\sum_{l=1}^{\infty} e^{-c}\frac{c^l}{l!}\,\mathbb{E}\Big[\frac{l}{M+l}\Big]\\ &= \frac{b}{c}\mathbb{E}\Big[\frac{L}{M+L}\Big] = \frac{b}{c}\mathbb{E}\Big[1-\frac{M}{M+L}\Big].\end{aligned}$$

This implies $\mathbb{E}\Big[\frac{M}{M+L}\Big] = \frac{1}{1+\frac{b}{c}}\mathbb{1}_{\{c>0\}} + \mathbb{1}_{\{c=0\}} = \frac{1}{1+\frac{b}{c}\mathbb{1}_{\{c>0\}}}$. By Jensen's inequality, this yields

$$\mathbb{E}\Big[1-\frac{M^2}{N^2}\Big] \le 1-\mathbb{E}\Big[\frac{M}{N}\Big]^2 = 1-\mathbb{E}\Big[\frac{M}{M+L}\Big]^2 \le \Big(1-\frac{1}{\big(1+\frac{b}{c}\big)^2}\Big)\mathbb{1}_{\{c>0\}}.$$

Furthermore, as $M/(M+L) \le 1$ we have

$$\mathbb{E}\Big[\frac{M^2}{N^2}\Big] = \mathbb{E}\Big[\frac{M^2}{(M+L)^2}\Big] \le \mathbb{E}\Big[\frac{M}{M+L}\Big] \le \frac{1}{1+\frac{b}{c}\mathbb{1}_{\{c>0\}}}.$$

Finally, the upper bound obtained in Step 4 yields

$$\begin{aligned}&\mathbb{E}(d_{FGW}(\eta^{(\hat{\mathrm{H}},\hat{\mathrm{T}})},\eta^{(\hat{\Xi},\hat{\Sigma})}))\\ &\le \tilde{C}_1\big(\max_{k\in[m]}\operatorname{diam}(\Omega_k)\big)\mathbb{E}\Big[\frac{M^2}{N^2}\Big] + \tilde{C}_2\mathbb{E}\Big[2-\frac{M}{N^2}\Big]\mathbb{E}\Big[1-\sum_{k\in[m]}\frac{n_k}{n}\wedge\hat{\nu}_k\Big] + \tilde{C}_2\mathbb{E}\Big[1-\frac{M^2}{N^2}\Big],\\ &\le \tilde{C}_1\big(\max_{k\in[m]}\operatorname{diam}(\Omega_k)\big)\frac{1}{1+\frac{b}{c}\mathbb{1}_{\{c>0\}}} + 4\tilde{C}_2\frac{m}{n}\mathbb{E}(|\Lambda|) + \tilde{C}_2\Big(1-\frac{1}{\big(1+\frac{b}{c}\big)^2}\Big)\mathbb{1}_{\{c>0\}}.\end{aligned}$$

□

Remark E.1. *A special case of Theorem 5.1 can be obtained by setting the edge probability function $\kappa \equiv 0$. Then we a.s. have no edges and the random graphs constructed as in Algorithm 3 can be described by their vertex processes* H *and* Ξ*. The FGW distance then simplifies to the 1-Wasserstein distance. In particular, we can bound the accuracy of Algorithm 3 for vertex generation with the results obtained in Theorem 5.1 by setting $\alpha = 0$.*

Theorem 5.3. *Assume that κ is Lipschitz continuous with Lipschitz constant L_κ in both components and let $(P^{\mu_{\mathcal{X}}^a}\otimes Q^\kappa)$ and $(P^{\mu_{\mathcal{Y}}^b}\otimes Q^\kappa)$ be distributions of the network models with true and private data input of expected size $a > 0$ and $b > 0$, respectively, constructed as in Section 4.1 by using a probability measure $\hat{\nu}$ created by Algorithm 1 with discrete random noise Λ. Then, using the FGW distance with parameter $\alpha \in [0,1]$,*

$$\begin{aligned}&d_{\mathcal{F}}(P^{\mu_{\mathcal{Y}}^b}\otimes Q^\kappa, P^{\mu_{\mathcal{X}}^a}\otimes Q^\kappa)\\ &\le \Big(1+\frac{1}{1+\frac{a\wedge b}{|a-b|}\mathbb{1}_{\{|a-b|>0\}}}\Big)(1-\alpha)\max_{k\in[m]}\operatorname{diam}(\Omega_k) + \tilde{C}_2\Big(1-\frac{1}{\big(1+\frac{a\wedge b}{|a-b|}\big)^2}\Big)\mathbb{1}_{\{|a-b|>0\}}\\ &\quad + 2c_V(a\wedge b)\frac{a\wedge b}{n}\mathrm{Leb}^d(\Omega)\mathbb{E}|\Lambda| + c_E(a\wedge b)\,2L_\kappa\max_{k\in[m]}\operatorname{diam}(\Omega_k)^3(a\wedge b)^2,\end{aligned} \tag{7}$$

where

$$c_V(c) := \min\Big\{1, \frac{1}{c}\big(1 + (1 - e^{-c}) \log^+ c\big)\Big\} C_\alpha \quad \textit{and} \quad c_E(c) := \min\Big\{1, \frac{2 - e^{-c}}{c} - \frac{1}{c^2}\Big(\frac{3}{2} - e^{-c}\Big)\Big\} \alpha C$$

for $C_\alpha := (1-\alpha)\mathrm{diam}(\Omega) + \alpha C$ *and* $\tilde{C}_2 = (1-\alpha)\mathrm{diam}(\Omega) + \alpha \min\{C, 2CL_\kappa \mathrm{diam}(\Omega)\}$ *with* $C > 0$ *given in* (5).

Proof. The strategy of this proof is as follows. We start by introducing so-called intermediate graphs (of the same size) that are an auxiliary tool to handle the different support of H (having attributes in $\mathcal{X}$) and Ξ (having attributes in $\mathcal{Y}$). This is necessary as the general bounds in Schuhmacher and Wirth (2024, Theorem 4.7) are of total variation type and thus will not vanish in a direct comparison. Furthermore, we bound the distance of the intermediate graphs to their respective original graph. We then compare the two intermediate graphs using a special case of Schuhmacher and Wirth (2024, Theorem 4.7). Finally, we assemble all results to obtain to obtain the claimed upper bound. We write $\mathbb{P}_\lambda$ and $\mathbb{E}_\lambda$ to denote the conditional probability and expectation given $\lambda = (\lambda_i)_{i=1}^m$.

Step 1: Intermediate graphs
Recall from Section 4.1, that we have $N \sim \mathrm{Poi}(a)$ vertices in H and $M \sim \mathrm{Poi}(b)$ vertices in Ξ; assume w.l.o.g. that $a \geq b$. Then $N \overset{d}{=} M + K$ for $K \sim \mathrm{Poi}(a-b)$. In particular, we can couple N and M such that $N \geq M$.

The intermediate graphs are obtained by resampling the (first) M vertices uniformly in their corresponding cell of the partition while keeping the edge probabilities of the original graph. We start with describing the first intermediate graph $(\mathrm{H}_I, \mathrm{T}_I)$ corresponding to the graph (H, T) with true input data $\mathcal{X}$. Let the vertex process H_I be given by

$$\mathrm{H}_I = \sum_{i=1}^m \sum_{j=1}^{N_i^I} \delta_{(X_{ij}^I, U_{ij}^I)}$$

where $X_{ij}^I \sim \mathcal{U}(\Omega_i)$, $U_{ij}^I \sim \mathcal{U}([0,1])$ and $N_i^I \sim \mathrm{Poi}(\frac{n_i}{n} b)$ are independent of other random variables. Furthermore, define the edge process T_I by

$$\mathrm{T}_I = \sum_{i,i'=1}^m \sum_{j=1}^{N_i^I} \sum_{j'=1}^{N_{i'}^I} \tilde{K}_{iji'j'}^{\mathrm{T}_I} \delta_{\{U_{ij}^I, U_{i'j'}^I\}}$$

for $\tilde{K}_{iji'j'}^{\mathrm{T}_I} \overset{i.i.d.}{\sim} \mathrm{Ber}(\kappa(\tilde{X}_{ij}, \tilde{X}_{i'j'}))$, where $\tilde{X}_{kl} \sim \mathcal{U}(\mathcal{X} \cap \Omega_k)$ for $k \in [m]$ and $l \in [N_k^I]$. We denote the distribution of the first intermediate graph $(\mathrm{H}_I, \mathrm{T}_I)$ by $P_I^{\mu_{\mathcal{X}}^a} \otimes Q_I^{\kappa_{\mathrm{T}}}$.

We construct a coupling $((\hat{\mathrm{H}}, \hat{\mathrm{T}}), (\hat{\mathrm{H}}_I, \hat{\mathrm{T}}_I))$ of the intermediate graph $(\mathrm{H}_I, \mathrm{T}_I)$ and the graph (H, T) with true input data $\mathcal{X}$ by setting $\hat{\mathrm{H}} = \mathrm{H}$ with $N = M + L$ for $L \sim \mathrm{Poi}(a-b)$ and $\hat{\mathrm{T}} = \mathrm{T}$. Furthermore, we define $\hat{\mathrm{H}}_I := \sum_{i=1}^M \delta_{(\hat{X}_i^I, U_{ij})}$ for $\hat{X}_i^I \sim \mathcal{U}(\Omega_l)$ whenever $X_i \in \Omega_l$ and set $\hat{\mathrm{T}}_I = \sum_{\substack{i,j=1 \\ i \neq j}}^M K_{ij}^{\mathrm{T}} \delta_{\{\hat{U}_i, \hat{U}_j\}}$. In particular, the intermediate graph is obtained by uniformly resampling each attribute in the corresponding partition cell while keeping the edges fixed.

It remains to show that this defines a coupling, i.e. $(\hat{\mathrm{H}}_I, \hat{\mathrm{T}}_I) \overset{d}{=} (\mathrm{H}_I, \mathrm{T}_I)$. Let $\hat{N}_i := \hat{\mathrm{H}}_I(\Omega_i)$ be the number of vertices in $\hat{\mathrm{H}}_I$ that are in Ω_i. Recall from Section 5.1 that the numbers of vertices $\mathbb{N} \,|\, N := (N_1, \ldots, N_m)$ of H in each partition set is multinominal distributed, i.e. $\mathbb{N} = \sum_{j=1}^N \mathbb{N}^j$ for $\mathbb{N}^j = (N_1^j, \ldots, N_m^j) \sim \mathrm{Multin}(1, (\frac{n_1}{n}, \ldots, \frac{n_m}{n}))$

independently. In particular, the construction of $\hat{\mathrm{H}}_I$ yields that for any $k_1, \dots, k_m \in \mathbb{Z}$ with $k := \sum_{i=1}^m k_i$ we have

$$\begin{aligned}
\mathbb{P}(\hat{N}_1 = k_1, \dots \hat{N}_m = k_m) &= \mathbb{P}\Big(\sum_{j=1}^{M} (N_1^j, \dots, N_m^j) = (k_1, \dots, k_m) \Big) \\
&= \sum_{l=0}^{\infty} \mathbb{P}(M = l) \mathbb{P}\Big(\sum_{j=1}^{M} (N_1^j, \dots, N_m^j) = (k_1, \dots, k_m) \,\Big|\, M = l \Big) \\
&= \mathbb{P}(M = k) \mathbb{P}\Big(\sum_{j=1}^{k} (N_1^j, \dots, N_m^j) = (k_1, \dots, k_m) \Big) \\
&= e^{-b} \frac{b^k}{k!} \, \frac{k!}{k_1! \cdots k_m!} \prod_{i=1}^{m} \left(\frac{n_i}{n} \right)^{k_i} \\
&= \prod_{i=1}^{m} e^{-\frac{n_i}{n} b} \frac{\left(\frac{n_i}{n} b\right)^{k_i}}{k_i!} \\
&= \mathbb{P}(N_1^I = k_1, \dots N_m^I = k_m),
\end{aligned}$$

where we used the independence of the N_i^I for the last equation. This implies $\mathrm{H}_I \overset{d}{=} \hat{\mathrm{H}}_I$ as in both cases vertices were chosen uniformly from the partition cell. Furthermore, $\mathrm{T}_I \overset{d}{=} \hat{\mathrm{T}}_I$ as $\tilde{X}_{kl} \overset{d}{=} X_i \,|\, (X_i \in \Omega_k)$.

To bound the expected FGW distance of the intermediate graph and the graph with true input data, consider the coupling $((\hat{\mathrm{H}}, \hat{\mathrm{T}}), (\hat{\mathrm{H}}_I, \hat{\mathrm{T}}_I))$ and note that $\hat{\mathrm{H}}_I$ is obtained from $\hat{\mathrm{H}}$ by resampling the first M vertices in $\hat{\mathrm{H}}$ uniformly in its corresponding partition cell. In particular, $d(X_i, \hat{X}_i^I) \leq \max_{k \in [m]} \mathrm{diam}(\Omega_k)$ for all $i \in [M]$. Furthermore, we constructed $\hat{\mathrm{T}}_I$ in such a way that the edges are the same as in $\hat{\mathrm{T}}$. In particular, we obtain $d_{\hat{\mathrm{T}}}(U_i, U_j) - d_{\hat{\mathrm{T}}_I}(U_i, U_j) = 0$ for all $i, j \in [M]$.

We derive an upper bound of the expected FGW distance of the intermediate graph and the graph with true input data by restricting to transport plans that move mass $1/N$ from vertex X_i (having mass $1/N$) to vertex $\hat{X}_i^I$ (having mass $1/M$) for all $i \in [M]$ and distribute the remaining mass $(1 - M/N)$ according to some coupling of the remaining vertices. Note that this is analogous to the approach in Step 3 in the proof of Theorem 5.1. Then for $\eta^{(\hat{\mathrm{H}}, \hat{\mathrm{T}})}$ and $\eta^{(\hat{\mathrm{H}}_I, \hat{\mathrm{T}}_I)}$ the random measures that capture the (attributed) graph information of $(\hat{\mathrm{H}}, \hat{\mathrm{T}})$ and $(\hat{\mathrm{H}}_I, \hat{\mathrm{T}}_I)$ we obtain

$$\begin{aligned}
&\mathbb{E}[\mathbb{E}(d_{FGW}(\eta^{(\hat{\mathrm{H}}, \hat{\mathrm{T}})}, \eta^{(\hat{\mathrm{H}}_I, \hat{\mathrm{T}}_I)}) \,|\, N, M)] \\
&\leq \mathbb{E}\left[\frac{M^2}{N^2} \right] (1 - \alpha) (\max_{k \in [m]} \mathrm{diam}(\Omega_k)) + \tilde{C}_2 \mathbb{E}\left[1 - \frac{M^2}{N^2} \right] \\
&\leq \frac{1}{1 + \frac{b}{c} \mathbb{1}_{\{c > 0\}}} (1 - \alpha) (\max_{k \in [m]} \mathrm{diam}(\Omega_k)) + \tilde{C}_2 \left(1 - \frac{1}{\left(1 + \frac{b}{c}\right)^2} \right) \mathbb{1}_{\{c > 0\}},
\end{aligned}$$

for $\tilde{C}_2 = (1 - \alpha)\mathrm{diam}(\Omega) + \alpha \min\{C, 2\, C\, L_\kappa \mathrm{diam}(\Omega)\}$ and $c = b - a$. We refer to Step 1 and Step 2 in the proof of Theorem 5.1 for details.

We define the second intermediate graph (Ξ_I, Σ_I) in an analogous way. That is, we set

$$\Xi_I = \sum_{i=1}^{m} \sum_{j=1}^{M_i} \delta_{(Y_{ij}^I, V_{ij}^I)},$$

where $V_{ij}^I \sim \mathcal{U}([0,1])$ and $Y_{ij}^I \sim \mathcal{U}(\Omega_i)$ for $j \in [M_i]$ with $M_i \sim \mathrm{Poi}(\hat{\nu}_i b)$. Further, we define the edge process Σ_I by

$$\Sigma_I = \sum_{i,i'=1}^{m} \sum_{j=1}^{M_i} \sum_{j'=1}^{M_{i'}} \tilde{K}_{ij}^{\Sigma_I} \delta_{\{V_i^I, V_j^I\}}$$

for $\tilde{K}_{iji'j'}^{\Sigma_I} \overset{i.i.d.}{\sim} \mathrm{Ber}(\kappa(y_i, y_{i'}))$. We denote the distribution of the second intermediate graph (Ξ_I, Σ_I) by $P_I^{\mu_{\mathcal{Y}}^b} \otimes Q_I^{\kappa_\Sigma}$. Analogously to the study of the first intermediate graph we can couple (Ξ_I, Σ_I) and (Ξ, Σ) such that $|\Xi| = M =$

$\sum_{i=1}^m M_i = |\Xi_I|$. In particular, we compare two graphs of the same size and the bound derived for the first intermediate graph simplifies to

$$\mathbb{E}(d_{FGW}(\eta^{(\Xi,\Sigma)}, \eta^{(\Xi_I,\Sigma_I)})) \leq (1-\alpha)(\max_{k\in[m]} \mathrm{diam}(\Omega_k)).$$

Step 2: Comparison of the intermediate graphs
Consider the intermediate graphs and note that both vertex processes H_I and Ξ_I are Poisson point processes, with intensity functions $\lambda_{\mathrm{H}_I}(x) = \frac{n_i}{n} b$ and $\lambda_{\Xi_I}(x) = \hat{\nu}_i b$ for $x \in \Omega_i$, respectively. Furthermore, conditionally on the vertex attributes, the edges are independent. In particular, both intermediate graphs are so called generalized random geometric graphs, see Schuhmacher and Wirth (2024, Section 2.2). Thus, we can apply Schuhmacher and Wirth (2024, Theorem 4.9) in the special case that the third term vanishes, see Schuhmacher and Wirth (2024, Corollary 3.2). This yields

$$\begin{aligned} &d_{\mathcal{F}}(P_I^{\mu_{\mathcal{Y}}^b} \otimes Q_I^{\kappa_\Sigma}, P_I^{\mu_{\mathcal{X}}^a} \otimes Q_I^{\kappa_\mathrm{T}}) \\ &\leq c_V(b)\mathbb{E}\Big(\int_{[0,1]}\int_\Omega \big|\lambda_{\Xi_I}(x) - \lambda_{\mathrm{H}_I}(x)\big|\,\alpha(\mathrm{d}x)\,\mathrm{d}u\Big) \\ &\quad + c_E(b)\,\mathbb{E}\Big(\int_{[0,1]}\int_\Omega \sum_{(y,v)\in\Xi_I} |\kappa_\mathrm{T}((x,u),(y,v)) - \kappa_\Sigma(x,y)|\,\lambda_{\mathrm{H}_I}(x)\,\alpha(\mathrm{d}x)\,\mathrm{d}u\Big), \end{aligned} \tag{12}$$

where $\kappa_\mathrm{T}((x,u),(y,v)) = \kappa_\mathrm{T}(x,y)$ and $\kappa_\Sigma((x,u),(y,v)) = \kappa_\Sigma(x,y)$ denote the probabilities of an edge u, v to be in T and Σ, respectively. Furthermore,

$$c_V(b) = \min\Big\{C_\alpha, \frac{1}{b}\left(1 + (1-e^{-b})\log^+ b\right) C_\alpha\Big\}, \qquad c_E(b) = \min\Big\{1, \frac{2-e^{-b}}{b} - \frac{1}{b^2}\Big(\frac{3}{2} - e^{-b}\Big)\Big\}\,\alpha C$$

for $C_\alpha := (1-\alpha)\mathrm{diam}(\Omega) + \alpha C$ an upper bound of d_{FGW}.

Note that in contrast to Schuhmacher and Wirth (2024) we choose the FGW distance d_{FGW} instead of the GOSPA metric $d_\mathbb{G}$ as a underlying metric for the Wasserstein metric. However, $d_{FGW} \leq d_\mathbb{G}$ (assuming that in both cases the vertex part is bounded by $(1-\alpha)\mathrm{diam}(\Omega)$ while the edge part is bounded by αC; compare Vayer et al. (2020) and Schuhmacher and Wirth (2023)). Thus, we obtain the same type of bounds.

Applying the Mecke formula, see for example Kallenberg (1997, Corollary 31.11), and using that $\lambda_{\mathrm{H}_I}(x) = \frac{n_i}{n} b$ and $\lambda_{\Xi_I}(x) = \hat{\nu}_i b$ for $x \in \Omega_i$, we obtain for $\tilde{X}_i \sim \mathcal{U}(\mathcal{X} \cap \Omega_i)$

$$\begin{aligned} d_{\mathcal{F}}(P_I^{\mu_{\mathcal{Y}}^b} \otimes Q_I^{\kappa_\Sigma}, P_I^{\mu_{\mathcal{X}}^a} \otimes Q_I^{\kappa_\mathrm{T}}) &\leq c_V(b)\,\mathbb{E}\Big(\int_\Omega \big|\lambda_{\Xi_I}(x) - \lambda_{\mathrm{H}_I}(x)\big|\,\alpha(\mathrm{d}x)\Big) \\ &\quad + c_E(b)\int_\Omega\int_\Omega \mathbb{E}\Big[\mathbb{E}_\lambda\big|\kappa_\mathrm{T}(x,y) - \kappa_\Sigma(x,y)\big|\,\lambda_{\mathrm{H}_I}(x)\,\lambda_{\Xi_I}(y)\Big]\,\alpha(\mathrm{d}x)\,\alpha(\mathrm{d}y) \\ &= c_V(b)\,\mathbb{E}\Big(\sum_{i=1}^m \int_{\Omega_i}\left|\frac{n_i}{n} - \hat{\nu}_i\right| b\,\alpha(\mathrm{d}x)\Big) \\ &\quad + c_E(b)\sum_{i,j=1}^m \int_{\Omega_i}\int_{\Omega_j} \mathbb{E}_\lambda|\kappa(\tilde{X}_i,\tilde{X}_j) - \kappa(y_i,y_j)|\,\frac{n_i}{n}\,\mathbb{E}[\hat{\nu}_j]\,b^2\,\alpha(\mathrm{d}x)\alpha(\mathrm{d}y) \\ &= c_V(n)\sum_{i=1}^m \mathrm{Leb}^d(\Omega_i)\mathbb{E}\left|\frac{n_i}{n} - \hat{\nu}_i\right| b \\ &\quad + c_E(b)\sum_{i,j=1}^m \mathrm{Leb}^d(\Omega_i)\mathrm{Leb}^d(\Omega_j)\mathbb{E}_\lambda|\kappa(\tilde{X}_i,\tilde{X}_j) - \kappa(y_i,y_j)|\,\frac{n_i}{n}\,\mathbb{E}[\hat{\nu}_j]\,b^2 \\ &\leq c_V(b)\sum_{i=1}^m \mathrm{Leb}^d(\Omega_i)\mathbb{E}\left|\frac{n_i}{n} - \hat{\nu}_i\right| b \\ &\quad + c_E(b)\sum_{i,j=1}^m \mathrm{Leb}^d(\Omega_i)\mathrm{Leb}^d(\Omega_j)\,2L_\kappa \max_{k\in[m]}\mathrm{diam}(\Omega_k)\,\frac{n_i}{n}\,\mathbb{E}[\hat{\nu}_j]\,b^2, \end{aligned}$$

where the last inequality follows by the Lipschitz property of κ, see Inequality (11).

Step 3: Derivation of the upper bound
Finally, we use the results of Step 1 and Step 2 to obtain the claimed upper bound. An application of the triangle inequality yields

$$
\begin{aligned}
&d_{\mathcal{F}}(P^{\mu_{\mathcal{Y}}^b} \otimes Q^{\kappa}, P^{\mu_{\mathcal{X}}^a} \otimes Q^{\kappa}) \\
&\leq d_{\mathcal{F}}(P^{\mu_{\mathcal{Y}}^b} \otimes Q^{\kappa}, P_I^{\mu_{\mathcal{Y}}^b} \otimes Q_I^{\kappa_\Sigma}) + d_{\mathcal{F}}(P_I^{\mu_{\mathcal{Y}}^b} \otimes Q_I^{\kappa_\Sigma}, P_I^{\mu_{\mathcal{X}}^a} \otimes Q_I^{\kappa_\mathrm{T}}) + d_{\mathcal{F}}(P_I^{\mu_{\mathcal{X}}^a} \otimes Q_I^{\kappa_\mathrm{T}}, P^{\mu_{\mathcal{X}}^a} \otimes Q_I^{\kappa_\mathrm{T}}) \\
&\leq \mathbb{E}(d_{FGW}((\Xi, \Sigma), (\Xi_I, \Sigma_I))) + d_{\mathcal{F}}(P_I^{\mu_{\mathcal{Y}}^b} \otimes Q_I^{\kappa_\Sigma}, P_I^{\mu_{\mathcal{X}}^a} \otimes Q_I^{\kappa_\mathrm{T}}) + \mathbb{E}(d_{FGW}((\mathrm{H}, \mathrm{T}), (\mathrm{H}_I, \mathrm{T}_I))) \\
&\leq \left(1 + \frac{1}{1 + \frac{b}{c}\mathbb{1}_{\{c>0\}}}\right)(1-\alpha) \max_{k\in[m]} \mathrm{diam}(\Omega_k) + \tilde{C}_2\left(1 - \frac{1}{\left(1+\frac{b}{c}\right)^2}\right)\mathbb{1}_{\{c>0\}} \\
&+ c_V(b) \sum_{i=1}^m \mathrm{Leb}^d(\Omega_i)\mathbb{E}\left|\frac{n_i}{n} - \hat{\nu}_i\right| b + c_E(b) \sum_{i,j=1}^m \mathrm{Leb}^d(\Omega_i)\mathrm{Leb}^d(\Omega_j)\, 2L_\kappa \max_{k\in[m]} \mathrm{diam}(\Omega_k)\, \frac{n_i}{n}\, \mathbb{E}[\hat{\nu}_j]\, b^2 \\
&\leq \left(1 + \frac{1}{1 + \frac{b}{c}\mathbb{1}_{\{c>0\}}}\right)(1-\alpha) \max_{k\in[m]} \mathrm{diam}(\Omega_k) + \tilde{C}_2\left(1 - \frac{1}{\left(1+\frac{b}{c}\right)^2}\right)\mathbb{1}_{\{c>0\}} \\
&+ 2c_V(b)\frac{b}{n}\mathrm{Leb}^d(\Omega)\mathbb{E}|\Lambda| + c_E(b)\, 2L_\kappa \max_{k\in[m]} \mathrm{diam}(\Omega_k)^3\, b^2,
\end{aligned}
$$

where the last inequality follows as analogous to the proof of Theorem 5.1; we have

$$
\sum_{i=1}^m \mathrm{Leb}^d(\Omega_i)\mathbb{E}\left|\frac{n_i}{n} - \hat{\nu}_i\right| \leq \frac{2}{n}\sum_{i=1}^m \mathrm{Leb}^d(\Omega_i)\mathbb{E}|\Lambda| = \frac{2}{n}\mathrm{Leb}^d(\Omega)\mathbb{E}|\Lambda|.
$$

□

Corollary 5.4. *Set the expected network sizes to be $a = b$ and $\Omega = [0,1]^d$. Assume that κ is Lipschitz continuous with Lipschitz constant L_κ in both components. Consider networks being distributed according to Section 4.1 w.r.t. a probability measure $\hat{\nu}$ created by Algorithm 1 using discrete noise Λ and a partition $(\Omega_k)_{k=1}^m$ satisfying* $\mathrm{diam}(\Omega_k) \asymp m^{-1/d}$. *Then Theorem 5.3 simplifies to*

$$
d_{\mathcal{F}}(P^{\mu_{\mathcal{Y}}^b} \otimes Q^{\kappa}, P^{\mu_{\mathcal{X}}^a} \otimes Q^{\kappa}) \leq 2\frac{1-\alpha}{m^{1/d}} + C_\alpha \frac{\left(1 + \log^+ a\right)}{n}\mathbb{E}|\Lambda| + 4\alpha C L_\kappa \frac{a}{m^{3/d}}. \qquad \text{(eq: cor special case Stein)}
$$

Proof. As $a = b$, Theorem 5.3 yields that

$$
\begin{aligned}
&d_{\mathcal{F}}(P^{\mu_{\mathcal{Y}}^b} \otimes Q^{\kappa}, P^{\mu_{\mathcal{X}}^a} \otimes Q^{\kappa}) \qquad (13)\\
&\leq 2(1-\alpha) \max_{k\in[m]} \mathrm{diam}(\Omega_k) + 2c_V(a)\frac{a}{n}\mathrm{Leb}^d(\Omega)\mathbb{E}|\Lambda| + c_E(a)\, 2L_\kappa \max_{k\in[m]} \mathrm{diam}(\Omega_k)^3 a^2.
\end{aligned}
$$

Furthermore, we can bound

$$
\begin{aligned}
c_V(a) &= \min\left\{1, \frac{1}{a}\left(1 + (1-e^{-a})\log^+ a\right)\right\} C_\alpha \leq \frac{1}{a}(1 + \log^+ a)C_\alpha, \\
c_E(a) &= \min\left\{1, \frac{2-e^{-a}}{a} - \frac{1}{a^2}\left(\frac{3}{2} - e^{-a}\right)\right\}\alpha C \leq \frac{2}{a^2}.
\end{aligned}
$$

Applying these bounds together with the assumption that $\mathrm{diam}(\Omega_k) \asymp m^{-1/d}$ to Equation (13) we obtain the claim. □

E.1 Derivation of optimal parameter choices

In the following we provide details on how the graph size a and the partition size $m = \lceil f_n(\epsilon)n \rceil$ should be chosen to obtain optimal rates in the bounds derived in Corollary 5.2 and Corollary 5.4, see Section 5.3.

We first consider the graph size a. As the bound in (9) does not depend on a, we can focus on (8) to obtain an optimal parameter choice for the graph size a. On the one hand, we would like to choose the expected graph

size $a = a_n$ as large as possible to have a good sample size for our graphs. On the other hand, (8) yields better bounds the smaller a is. As the first term in (8) does not depend on a, an optimal choice of a is such that the first and third terms in (8) have the same speed of convergence. Note that we do not consider the second term here as due to the log-term the influence of a is negligible compared to the third term. Then

$$m^{-1/d} \asymp a m^{-3/d} \iff a = m^{2/d}.$$

Recall that in abuse of the notion, although optimality conditions are only given in terms of orders of magnitude, here we take the respective constants to be 1, and any lower order terms are taken to be zero.

In particular, using that $m = \lceil f_n(\epsilon)n \rceil$ Inequality (8) turns into

$$d_{\mathcal{F}}(P^{\mu_{\mathcal{Y}}^b} \otimes Q^\kappa, P^{\mu_{\mathcal{X}}^a} \otimes Q^\kappa) \leq \big(2(1-\alpha) + 4\alpha C L_\kappa\big)(f_n(\epsilon)n)^{-\frac{1}{d}} + C_\alpha(1 + 2\log^+(f_n(\epsilon)n))(\epsilon n)^{-1}. \tag{14}$$

Under this optimal graph size $a = m^{2/d}$, we now ask for a partition size $m = \lceil f_n(\epsilon)n \rceil$ that optimizes the rate of convergence in (9) and (14). We formulate this optimal choice of m by deriving an optimal decay of $f_n(\epsilon)$ as $\epsilon \to 0$. Note that in both, (9) and (14), we obtain a trade-off for the choice of f_n. More precisely, to obtain fast convergence (to zero) in the first terms, we would like $f_n(\epsilon)$ to go to zero slowly. On the other hand, the second terms decrease faster the faster $f_n(\epsilon)$ goes to zero. However, note that due to the logarithmic expression the observed trade-off in (14) is weaker than the trade-off in (9). In particular, we focus on (9) to derive an optimal decay for f_n.

Due to the trade-off between the two terms in (9), an optimal decay for f_n can be obtained by assuming the same speed of convergence for both terms in (9). This corresponds to setting

$$f_n(\epsilon)/\epsilon \asymp \big(f_n(\epsilon)n\big)^{-1/d} \iff f_n(\epsilon) \asymp \epsilon^{\frac{d}{d+1}} n^{-\frac{1}{d+1}}.$$

This choice of f_n yields an optimal partition size $m = \lceil f_n(\epsilon)n \rceil \asymp \epsilon^{\frac{d}{d+1}} n^{\frac{d}{d+1}}$.

In Table 10 and Table 11 explicit values for an optimal choice of the partition size $m = (\epsilon n)^{\frac{d}{d+1}}$ and the expected graph size $a = m^{2/d}$ for different dimensions d are presented. It can be observed, that the larger the sample size n, the larger the partition size m and the expected graph size a, while a smaller the privacy parameter ϵ leads to smaller the partition size m and the expected graph size a. In particular, a reasonable sizes m and a can be obtained by choosing $\epsilon \asymp n^{-1}$. We would like to emphasize, that the optimality of these parameters is derived asymptotically, see Section 5.3. In particular, the optimality holds up to constant factors.

Table 10: Explicit values for optimal parameter choices $m = (\epsilon n)^{\frac{d}{d+1}}$ and $a = m^{2/d}$ for $d = 1$ made according to the optimal rates derived in Section 5.3 under the assumptions of Corollary 5.2 and Corollary 5.4. Note that the optimality of these parameter choices is established asymptotically and thus holds up to constant factors.

		n					
		100		1000		10,000	
		m	a	m	a	m	a
	1	10	100	31.62	1024	100	10000
ϵ	0.1	3.16	16	10	100	31.62	1024
	0.01	1	4	3.16	16	10	100

E.2 Further Results on Explicit Bounds

Here we give further results on explicit bounds, see Section 5.3. The results show a similar picture as in Table 2; Corollary 5.2 and Corollary 5.4 give bounds of the same order. Furthermore, the bounds decrease with increasing sample size n and increase with decreasing privacy parameter ϵ. A small α is favorable for both, Corollary 5.4 and Corollary 5.2. Furthermore, the obtained bounds decrease as the dimension d of the attribute space decreases. Overall the results illustrate the theoretic results obtained in Section 5.3.

Table 11: Explicit values for optimal parameter choices for the partition size $m = (\epsilon n)^{\frac{d}{d+1}}$ and the expected graph size $a = m^{2/d}$ for $d = 2$ made according to the optimal rates derived in Section 5.3 under the assumptions of Corollary 5.2 and Corollary 5.4. Note that the optimality of these parameter choices is established asymptotically and thus holds up to constant factors.

		n					
		100		1000		10,000	
		m	a	m	a	m	a
	1	21.54	22	100	101	464.16	465
ϵ	0.1	4.64	5	21.54	22	100	101
	0.01	1	2	4.64	5	21.54	22

Table 12: Explicit bounds obtained in Corollary 5.2 and Corollary 5.4 assuming discrete Laplacian noise and $\Omega = [0,1]^d$, for $d = 2$, $\alpha = 1/4$ as well as $C = L_\kappa = 1$. Furthermore, $m = (\epsilon n)^{\frac{d}{d+1}}$ and $a = m^{2/d}$ are chosen according to the optimal rate, see Section 5.3.

		n					
		100		1000		10,000	
		Corollary 5.2	Corollary 5.4	Corollary 5.2	Corollary 5.4	Corollary 5.2	Corollary 5.4
	1	0.697	0.574	0.324	0.254	0.151	0.117
ϵ	0.1	1.487	1.378	0.697	0.574	0.324	0.254
	0.01	2.884	3.367	1.487	1.378	0.697	0.574

F OTHER NOISE DISTRIBUTIONS: DISCRETE GAUSSIAN NOISE

In Section 5 we gave theoretical guarantees on the utility of our graph generator PSGG for general noise distributions that fulfill the condition in Proposition 3.1. We simplified the obtained bounds under the assumption that the privatized vertex attributes are generated using discrete Laplacian noise with scale given by $\frac{1}{\epsilon}$, noting that discrete Laplacian noise $\Lambda \sim \text{Lap}_{\mathbb{Z}}(1/\epsilon)$ has $\mathbb{E}|\Lambda| = 2\frac{e^{-\epsilon}}{1-e^{-2\epsilon}}$, see Inusah and Kozubowski (2006). This choice allowed a precise comparison to the results obtained in He et al. (2023).

In the following we derive a bound on $\mathbb{E}|\Lambda|$ for the (centered) discrete Gaussian noise truncated to a finite interval $[-b, b]$ for some $b > 0$. In Canonne et al. (2020), the discrete Gaussian noise distribution with mean μ and scale σ^2 is introduced as the distribution having probability mass function $p(k) \propto e^{\frac{-(k-\mu)^2}{2\sigma^2}}$ for $k \in \mathbb{Z}$. This distribution can be shown to have variance at most σ^2. We refer to Canonne et al. (2020) also for an interesting discussion of the (not truncated) discrete Gaussian noise in the context of differential privacy. Here we introduce the truncated discrete Gaussian noise distributon with mean μ, scale σ^2 and truncation parameter b by

$$P_\Lambda(k) \propto \begin{cases} e^{\frac{-k^2}{2\sigma^2}} & \text{for } k \in \mathbb{Z} \cap [-b, b] \\ 0 & \text{otherwise} \end{cases}.$$

We first show that the centered truncated discrete Gaussian noise fulfills the assumptions in Proposition 3.1, that

Table 13: Explicit bounds obtained in Corollary 5.2 and Corollary 5.4 assuming discrete Laplacian noise and $\Omega = [0,1]^d$, for $d = 2$, $\alpha = 3/4$ as well as $C = L_\kappa = 1$. Furthermore, $m = (\epsilon n)^{\frac{d}{d+1}}$ and $a = m^{2/d}$ are chosen according to the optimal rate, see Section 5.3.

		n					
		100		1000		10.000	
		Corollary 5.2	Corollary 5.4	Corollary 5.2	Corollary 5.4	Corollary 5.2	Corollary 5.4
	1	0.804	0.787	0.374	0.354	0.174	0.163
ϵ	0.1	1.711	1.825	0.804	0.787	0.374	0.354
	0.01	3.237	4.074	1.711	1.825	0.804	0.787

Table 14: Explicit bounds obtained in Corollary 5.2 and Corollary 5.4 assuming discrete Laplacian noise and $\Omega = [0,1]^d$, $d = 1$, $\alpha = 1/2$ as well as $C = L_\kappa = 1$. Furthermore, $m = (\epsilon n)^{\frac{d}{d+1}}$ and $a = m^{2/d}$ are chosen according to the optimal rate, see Section 5.3.

		n					
		100		1000		10,000	
		Corollary 5.2	Corollary 5.4	Corollary 5.2	Corollary 5.4	Corollary 5.2	Corollary 5.4
	1	0.350	0.356	0.110	0.102	0.035	0.031
ϵ	0.1	1.007	1.127	0.350	0.356	0.110	0.102
	0.01	2.750	3.861	1.007	1.127	0.350	0.356

Table 15: Explicit bounds obtained in Corollary 5.2 and Corollary 5.4 assuming discrete Laplacian noise and $\Omega = [0,1]^d$, $d = 5$, $\alpha = 1/2$ as well as $C = L_\kappa = 1$. Furthermore, $m = (\epsilon n)^{\frac{d}{d+1}}$ and $a = m^{2/d}$ are chosen according to the optimal rate, see Section 5.3.

		n					
		100		1000		10,000	
		Corollary 5.2	Corollary 5.4	Corollary 5.2	Corollary 5.4	Corollary 5.2	Corollary 5.4
	1	1.623	1.414	1.107	0.952	0.754	0.647
ϵ	0.1	2.379	2.202	1.623	1.414	1.107	0.952
	0.01	3.5	4	2.379	2.202	1.623	1.414

is, we want to show that for every $k \in \mathbb{Z} \cap [-b, b]$ and for $a \in \{-1, 1\}$ we have

$$\frac{P_\Lambda(k+a)}{P_\Lambda(k)} \leq e^{\tilde{\epsilon}}.$$

For this fix $k \in \mathbb{Z} \cap [-b, b]$ and note that if $k + a \notin \mathbb{Z} \cap [-b, b]$ the inequality is trivial. For $k + 1 \in \mathbb{Z} \cap [-b, b]$,

$$\frac{P_\Lambda(k+1)}{P_\Lambda(k)} = \frac{e^{\frac{-(k+1)^2}{2\sigma^2}}}{e^{\frac{-k^2}{2\sigma^2}}} = e^{\frac{-k^2-2k-1+k^2}{2\sigma^2}} = e^{\frac{-2k-1}{2\sigma^2}} \leq e^{\frac{b}{\sigma^2}}.$$

Analogously, we have for $k - 1 \in \mathbb{Z} \cap [-b, b]$ that

$$\frac{P_\Lambda(k-1)}{P_\Lambda(k)} = \frac{e^{\frac{-(k-1)^2}{2\sigma^2}}}{e^{\frac{-k^2}{2\sigma^2}}} = e^{\frac{-k^2+2k-1+k^2}{2\sigma^2}} = e^{\frac{2k-1}{2\sigma^2}} \leq e^{\frac{b}{\sigma^2}}.$$

Thus the assumptions of Proposition 3.1 are fulfilled for $\tilde{\epsilon} := \frac{b}{\sigma^2}$, which yields that the TV-PSMM algorithm with truncated discrete Gaussian noise is $\tilde{\epsilon}$-differential private. In what follows we choose σ^2 in such a way that $\epsilon \to 0$ if and only if $\tilde{\epsilon} \to 0$.

Next we bound $\mathbb{E}|\Lambda|$. For given $\epsilon > 0$ we choose $\sigma^2 = 1/\epsilon^2$ and let $b = 1/\epsilon$. Then $\tilde{\epsilon} = b/\sigma^2 = \epsilon$. In particular,

$$\mathbb{E}|\Lambda| \leq b = \frac{1}{\epsilon}.$$

We can use this bound in (6) and in (7). As this bound was used in Section 5.3 to further bound the absolute expectation of the discrete Laplacian noise, we obtain Equation (9) and the optimal parameter choices derived in Section 5.3 also apply to the discrete Gaussian noise distribution. This also shows that the utility guarantees for the PSGG based on truncated discrete Gaussian noise (with scale $\sigma^2 = 1/\epsilon^2$ and truncation to $[-\frac{1}{\epsilon}, \frac{1}{\epsilon}]$) is of the same order as the utility guarantees for discrete Laplacian noise (with scale $b^2 = 2/\epsilon^2$). Similarly, the privacy guarantees are the same as $\tilde{\epsilon} = \epsilon$. We note that we did not investigate an optimal choice of b. There is a trade-off: While a large b yields good lower bounds on the normalization constant and thus good utility guarantees, it also increases $\tilde{\epsilon} = b\epsilon^2$ which implies weaker privacy guarantees. Thus, a general comparison of the two noise distributions is difficult, see also Canonne et al. (2020).