UNIVERSITÉ DE LILLE

École doctorale Sciences pour l'Ingénieur

Thèse présentée par

Abdelrahman EID

Pour obtenir le grade de

Docteur en Mathématiques et leurs interactions

Spécialité

Statistique Mathématique

STOCHASTIC SIMULATIONS FOR GRAPHS AND MACHINE LEARNING

SIMULATIONS STOCHASTIQUES POUR LES GRAPHES ET L'APPRENTISSAGE AUTOMATIQUE

Soutenue publiquement le 10 Juillet 2020 devant le jury composé de

Président du jury:	Pr. Yann GUERMEUR	CNRS Nancy, France
Directeur de thèse:	Pr. Nicolas Wicker	Université de Lille, France
Rapporteurs:	Pr. Avner Bar-Hen	CNAM Paris, France
	Pr. Marianne Clausel	IECL Nancy, France
Examinateurs:	Dr. Charlotte Baey	Univérsité de Lille, France
	Dr. Rémi Bardenet	CNRS Lille, France

Contents

This work is dedicated to
my first teachers, my parents: Munther & Nuha.
the lady of my life, my beloved wife, Ghada.
my loving and wonderful blessing, Zaina.

Acknowledgment

First and above all, all thanks to Almighty God for giving me the power and patience to undertake this work and complete it satisfactorily. So ever appreciative of your blessings that you give me each and every day.

I would like to express my sincere gratitude for the reviewers of this thesis Prof. Avner Bar-Hen and Prof. Marianne Clausel for generously offering their time to review this work as well as for participating in my PhD committee. Your remarks, comments and suggestions on this work are really appreciated. Moreover, I am very grateful to Prof. Yann Guermeur for accepting to preside the defense committee. Besides, I want to thank Dr. Rémi Bardenet and Dr. Charlotte Baey for accepting to examine my work. I am really thankful for your participation, your brilliant comments, suggestions and letting my defense be an enjoyable moment, thanks a lot.

I would like to thank my supervisor Prof. Nicolas Wicker for his great support, guidance and the encouragement he has provided throughout my PhD study. It was a real privilege and an honor for me to work under his supervision and to share of his scientific knowledge. I have been extremely lucky to have a supervisor who cared so much about my work and academic future but also of his extraordinary human qualities and kind personality. I can't thank you enough for all of your support Nicolas.

I am also thankful to all of my colleagues and the staff members at Paul Painlevé laboratory for the enjoyable environment, the discussions and the time we shared together during the past four years. Moreover, I would like to thank Prof. Radu Stoica for his comments, discussions, suggestion in addition to the kind hospitality (at the University of Lorraine - Nancy) many times to finish part of chapter 2 in the thesis.

I would not have been able to complete this without some amazing people for whose help I am immeasurably grateful. I am greatly indebted to my parents, Munther and Nuha, for their continuous support and prayers. I am deeply thankful for always being there for me and for giving me the opportunities that have made me who I am.
I'm forever and particularly thankful for my loving wife, Ghada has been extremely supportive of me throughout this entire process and has made countless sacrifices to help me get to this point. I am also very thankful to my dearest daughter Zaina for giving me unlimited happiness, pleasure and the smile. This journey would not have been possible if not for

them. Thanks for all the love, rock steady support and energy you give to chase and achieve my dreams.
A warm thank you to my sisters, step-father, step-mother, step-brothers and all my friends, thank you all.

Lille,
July 12, 2020
Abdelrahman EID

Résumé

Bien qu'il ne soit pas pratique d'étudier la population dans de nombreux domaines et applications, l'échantillonnage est une méthode nécessaire permettant d'inférer l'information. Cette thèse est consacrée au développement des algorithmes d'échantillonnage probabiliste pour déduire l'ensemble de la population lorsqu'elle est trop grande ou impossible à obtenir. Les techniques Monte Carlo par chaîne de markov (MCMC) sont l'un des outils les plus importants pour l'échantillonnage à partir de distributions de probabilités surtout lorsque ces distributions ont des constantes de normalisation difficiles à évaluer.

Le travail de cette thèse s'intéresse principalement aux techniques d'échantillonnage pour les graphes. Deux méthodes pour échantillonner des sous-arbres uniformes à partir de graphes en utilisant les algorithmes de Metropolis-Hastings sont présentées dans le chapitre 2. Les méthodes proposées visent à échantillonner les arbres selon une distribution à partir d'un graphe où les sommets sont marqués. L'efficacité de ces méthodes est prouvée mathématiquement. De plus, des études de simulation ont été menées et ont confirmé les résultats théoriques de convergence vers la distribution d'équilibre.

En continuant à travailler sur l'échantillonnage des graphes, une méthode est présentée au chapitre 3 pour échantillonner des ensembles de sommets similaires dans un graphe arbitraire non orienté en utilisant les propriétés des processus des points permanents PPP. Notre algorithme d'échantillonnage des ensembles de k sommets est conçu pour surmonter le problème de la complexité de calcul lors du calcul du permanent par échantillonnage d'une distribution conjointe dont la distribution marginale est un kPPP.

Enfin, dans le chapitre 4, nous utilisons les définitions des méthodes MCMC et de la vitesse de convergence pour estimer la bande passante du noyau utilisée pour la classification dans l'apprentissage machine supervisé. Une méthode simple et rapide appelée KBER est présentée pour estimer la bande passante du noyau de la fonction de base radiale RBF en utilisant la courbure moyenne de Ricci de $\epsilon-$graphes.

Mots-clés: Échantillonnage MCMC, Échantillonnage de graphes, Échantillonnage de kPPP, La bande passante du noyau RBF, Estimation de la bande passante et Classification supervisée. .

Abstract

While it is impractical to study the population in many domains and applications, sampling is a necessary method allows to infer information. This thesis is dedicated to develop probability sampling algorithms to infer the whole population when it is too large or impossible to be obtained. Markov chain Monte Carlo (MCMC) techniques are one of the most important tools for sampling from probability distributions especially when these distributions have intractable normalization constants.

The work of this thesis is mainly interested in graph sampling techniques. Two methods in chapter 2 are presented to sample uniform subtrees from graphs using Metropolis-Hastings algorithms. The proposed methods aim to sample trees according to a distribution from a graph where the vertices are labelled. The efficiency of these methods is proved mathematically. Additionally, simulation studies were conducted and confirmed the theoretical convergence results to the equilibrium distribution.

Continuing to the work on graph sampling, a method is presented in chapter 3 to sample sets of similar vertices in an arbitrary undirected graph using the properties of the Permanental Point processes PPP. Our algorithm to sample sets of k vertices is designed to overcome the problem of computational complexity when computing the permanent by sampling a joint distribution whose marginal distribution is a kPPP.

Finally in chapter 4, we use the definitions of the MCMC methods and convergence speed to estimate the kernel bandwidth used for classification in supervised Machine learning. A simple and fast method called KBER is presented to estimate the bandwidth of the Radial basis function RBF kernel using the average Ricci curvature of $\epsilon-$graphs.

Keywords: MCMC sampling, Graph sampling, kPPP sampling, RBF kernel bandwidth, Bandwidth estimation and Supervised classification.

Publications

- Eid, Abdelrahman & Mamitsuka, Hiroshi & Wicker, Nicolas. (2019). *A Metropolis-Hastings Sampling of Subtrees in Graphs*. Austrian Journal of Statistics. 48. 17-33. 10.17713/ajs.v48i5.880.
- Eid, Abdelrahman & Wicker, Nicolas. *KBER: A Kernel Bandwidth Estimate using the Ricci Curvature*. Submitted, 2020.

List of Figures

List of Tables

Introduction

Random sampling is very important to describe selection procedures following to probability laws where the randomness is controlled by the design. In stochastic simulations, we simulate a system of variables that can change randomly with individual probabilities. Consequently, stochastic simulation methods are random number generators that allow samples to be drawn from a target density $\pi(x)$, such as the posterior probability $p(x/y)$. Hence, these samples are used to make inferences like approximating probabilities and expectations.

In this work, we focus on Markov chain Monte Carlo (MCMC) for sampling which is a class of stochastic simulation techniques. It works by constructing a Markov chain with a stationary distribution. In particular, we concentrate on MCMC algorithms for graph sampling to improve sampling from probability distributions and to handle the normalizing constants when needed.

The thesis is organized in four chapters where the first chapter is a general introduction to graphs, graphs sampling, the MCMC sampling methods, in addition to many definitions and concepts used in the applications through the chapters followed. Following, three chapters are presented which could be read independently of the others to some extent.

In **chapter 2**, two methods are presented to sample uniform subtrees within graphs using Metropolis-Hastings algorithms. Most methods in literature were proposed either to sample different structures from graphs that are generally inefficient to sample subtrees, or to sample trees where the structure is important. On the contrary, we aim to sample trees according to a distribution from a graph where the vertices are

labelled regardless the tree structure. To evaluate the efficiency of these methods, simulation studies were conducted and confirmed the theoretical convergence results of our Markov chains to the equilibrium distribution.

In **chapter 3**, we present a Metropolis-Hastings method to sample sets of similar vertices in an arbitrary undirected graph. We take advantage of the properties of the Permanental Point processes PPP to sample sets of k vertices using Markov chains. Moreover, our algorithm is designed to overcome the problem of computing the permanent for a huge number of vertices by sampling a joint distribution whose marginal distribution is a kPPP where computing the permanent is known of a high complexity and even impossible when the number of vertices increases.

Chapter 4 is devoted to link data analysis and Machine learning with the definitions and concepts studied through the previous chapters. More precisely, we use the definitions of the MCMC methods and convergence speed in order to estimate the kernel bandwidth used for classification in supervised Machine learning. A simple and fast method called KBER is presented to estimate the bandwidth of the Radial basis function RBF kernel using the average Ricci curvature of $\epsilon-$graphs.

Chapter 1

Preliminaries

This introductory chapter offers a presentation to the main concepts and techniques used in the subsequent chapters, in addition to present some results related to the work through the thesis.

1.1 From Graphs to Graph Sampling

A graph is a heavily used data structure in the world of algorithms. There are numerous applications of it in computer science like networks of communication, data organization, computational devices and the flow of computation. Graphs have proven particularly useful in linguistics in addition to the use of them in chemistry, physics, sociology and biology.

Let $G(V, M)$ be a graph consists of V as its finite set of vertices and M is a finite set of ordered pairs of the form (u, v) called edges. The graph is directed if the pair (u, v) is not the same as (v, u), where (u, v) indicates that there is an edge from vertex u to vertex v, while it is undirected if both pairs represent the same edge. Further, a graph is called regular if each vertex has the same number of neighbors; i.e. every vertex has the same degree. A special case of regular graphs is called complete graphs where any vertex in the graph is connected with all other vertices. A graph is connected if, for every pair (u, v) of distinct vertices, there is a path from u to v. A rigorous presentation of graph properties could be

found in (Bollobas, 1979) and (Bollobas, 2001).

Many applications are generating very huge graphs with thousands and millions of vertices. These large graphs are challenging to study because they need to run expensive algorithms such as simulations, in addition to the fact that it is hard to get an impression of the graph topology from visualization. Unfortunately, such activities on large graphs usually cost a lot of time that is computed at least polynomially in the number of vertices. One way to reduce the runtime is to reduce the graph size by sampling a structure from the large graph which approximates the original graph well.

Graph mining is a branch of data mining used for mining graph structures (Rehman *et al.*, 2012). It has gained much attention during the last years and finds its applications in many domains like: social and computer networks, bioinformatics and chemistry. Various approaches for graph mining in the literature have been proposed for classification, clustering and sampling. In general, the graph sampling methods, which are our concern, are divided into three categories (Ahmed *et al.*, 2011): node sampling, edge sampling and topology-based sampling.

Node sampling algorithm simply creates a representative structure by sampling the vertices uniformly where the edges between the sampled vertices in the large graph are considered as edges also in the sampled structure. A known related method is called random node-edge sampling (Hu and Lau, 2013) where vertices are uniformly sampled and edges that are incident to these vertices are also uniformly sampled in the sample graph. Additionally, some node sampling methods also consider the neighbors of the sampled vertices like the random node-neighbor sampling method (Leskovec and Faloutsos , 2006) where all the edges that are linked to the sampled vertices in the graph are sampled into the required structure.

Similarly, edge sampling builds a subgraph by sampling edges randomly. For instance, in random edge sampling (Ebbes *et al.* , 2008), the subgraph is built from edges sampled ran-

domly and uniformly. Another modified edge sampling method is the induced edge sampling (Ahmed *et al.*, 2012) with both of its extensions, the totally induced edge sampling and the partially induced edge sampling. The first one applies the random edge sampling and obtains adjacent vertices from these edges, then all edges attached to those vertices are chosen to the sampled graph. In contrast, partially induced edge sampling applies the edge sampling where edges are sampled according to a probability.

Finally, the topology-based sampling which is known by traversal based sampling and sampling by exploration is one of the most studied techniques recently. These methods use the graph topology information by integrating with topology-based sampling methods like the random walk sampling (Yoon *et al.* , 2007) and the Metropolis algorithm (Hübler *et al.*, 2008), which replaces some sampled vertices with other vertices, sample structure with properties consistent with the graph. Many related algorithms for traversal sampling were presented and proved their power like Breadth-First Search, Depth-First Search, Forest Fire and Snowball sampling. Similarly, random walk techniques like the classic random walk, Markov Chain Monte Carlo (MCMC) using Metropolis-Hastings algorithm, random walk with restart or with random jump.

1.2 MCMC Sampling

Statistical sampling is an important process to make inferences about underlying models from sources of collected data. Throughout the last decades, the amount of data has increased hugely which made it impossible to analyze them using the existing methods and technologies at that time. Consequently, a method called Monte Carlo which combines numerical simulation and random number was developed to analyze these models of data.

Markov Chain Monte Carlo (MCMC) methods represent a type of Monte Carlo methods. They are used to simulate samples from an unknown distribution then to benefit from those samples to perform subsequent analyses. MCMC methods construct a Markov chain according to a desired distribution represents the equilibrium distribution for the chain. Hence,

samples of the desired distribution are obtained from the chain and the distribution of these samples will be closer to the desired distribution as the the number of samples increases.

1.2.1 Markov chains

This section is intended as a minimalist introduction on Markov chains and their properties. Consider the sequence of random variables $(X_n)_{n\geq 0}$ with values belong to the set of states $S = \{1, \cdots, n\}$, the Markov chain $\left(X_n\right)_{n\geq 0}$ is defined over the same space S and, for any $x_0, \cdots x_n \in S$ and $n \in \mathbb{N}$, satisfies the Markov property:
if $\mathcal{P}(X_0 = x_0, \cdots, X_{n-1} = x_{n-1}) > 0$, then

$$\begin{aligned}\mathcal{P}(X_n = x_n | X_0 = x_0, \cdots, X_{n-1} = x_{n-1}) &= \mathcal{P}(X_n = x_n | X_{n-1} = x_{n-1}) \\ &= p_{x_{n-1}, x_n}\end{aligned}$$

Generally, $P = (p_{ij})_{i,j\in S}$ is called the transition probability so $p^{(n-1)}_{x_{n-1},x_n}$ is the transition probability from the state x_{n-1} to state x_n in time $n-1$. According to the Markov property, each next sample X_{n+1} depends only on the current state X_n and does not depend on the further history $X_0, X_1, \cdots, X_{n-1}$.

The Markov chain $\left(X_n\right)_{n\geq 0}$ is considered time homogeneous if $\mathcal{P}(X_n = x_n | X_{n-1} = x_{n-1}) = p_{x_{n-1},x_n}$ is independent of n. Also, a Markov chain is said to be irreducible if all states communicate with each other where the states x_i and x_j communicate with each other if x_j is reachable from x_i and vice-versa.

Additionally, the state x_i is recurrent if and only if $\sum_{n\geq 1} p^{(n)}_{x_i,x_i} = \infty$. As a rule, an irreducible Markov chain defined over a finite set of states is recurrent. Moreover, the state x_i has period k if any return to state x_i occurs necessarily in multiples of k time steps. If $k = 1$, then the state is said to be aperiodic. Furthermore, aperiodicity is a class property so an irreducible Markov chain is said to be aperiodic if there exists only one aperiodic state.

Finally, a Markov chain is said to be ergodic if it is irreducible, recurrent and aperiodic. The importance of ergodic Markov chains that they converge toward a unique stationary "often

called invariant" distribution where the stationary distribution of the Markov chain $\left(X_n\right)_{n\geq 0}$ is the probability distribution $\pi = (\pi_0, \cdots, \pi_n)$ such that $\pi = \pi P$.

1.2.2 Metropolis-Hastings algorithm

MCMC sampling methods are used for estimating constrained distributions. The power of these methods that they are fairly simple and simulate distributions close to analytically calculated distributions with less computation time and complexity in the calculations. Many algorithms were developed based on this technique. For a good review of these methods, see (Gilks *et al.*., 1996). Among the now-common MCMC methods, the Gibbs sampling method which is considered the workhorse of the MCMC world in addition to the Metropolis-Hastings algorithm. In this section, we only present the latter algorithm that is extensively applied in our work and all other MCMC methods could be considered as special cases of it.

The Metropolis algorithm was first presented by Metropolis *et al.* (1953), then it was generalized by Hastings (1970). Simply, the Metropolis algorithm generates a random walk that uses an acceptance/rejection ratio in order to converge to the desired distribution. The Metropolis-Hastings M-H algorithm is associated with the target density P and the conditional density q to produce a Markov chain converge to the invariant distribution π (Chib and Greenberg, 1995). Consequently, it generates a set of states according to the desired distribution P by using using a Markov process that asymptotically reaches the stationary distribution π such that $\pi = P$.

The M-H algorithm produces the Markov chains as follows. Consider x as the current state of the chain, a new candidate y is proposed according to a proposal distribution $q(x, y)$ then the Markov chain (X_{t+1}) is produced through the transition kernel:

$$X_{t+1} = \begin{cases} y \text{ with probability } = \alpha \\ x \text{ with probability } = 1 - \alpha \end{cases}$$

where α is the acceptance probability. Precisely, the proposed state is accepted as the new

state of the chain with probability:

$$\alpha = \frac{\pi(y)}{\pi(x)} \frac{q(y,x)}{q(x,y)} \wedge 1$$

Furthermore, the transition probability matrix $P(x,y)$ is given as

$$P(x,y) = \begin{cases} \left\{ \frac{\pi(y)}{\pi(x)} \frac{q(y,x)}{q(x,y)} \wedge 1 \right\} q(x,y) & \text{if } x \neq y \\ 1 - \left\{ \frac{\pi(y)}{\pi(x)} \frac{q(y,x)}{q(x,y)} \wedge 1 \right\} \sum_{i \neq x} q(x,i) & \text{otherwise} \end{cases}$$

1.3 Markov chains convergence

We defined in the previous section that all irreducible Markov chains have a unique stationary distribution π where any distribution over such a chain converges to π while the chain is ergodic. Naturally, the convergence of a Markov chain is evaluated by studying the spectral properties of a transition matrix P since the chain's stationary distribution is of a left eigenvector of its matrix P. Knowing that the transition matrix P has $N = |\omega|$ real eigenvalues $1 = \lambda_0 > \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{N-1} \geq -1$, the convergence of an ergodic chain is assured to the stationary distribution π by the second largest eigenvalue in absolute value, $\lambda_{max} = \max\{\lambda_1, |\lambda_{N-1}|\}$.

Following, we introduce a measure of distance to evaluate and prove the convergence of a distribution for a Markov chain then a summary for the used techniques in literature to assess how long the chain needs to take in order to be sufficiently close to the stationary distribution of the chain.

1.3.1 Total variation distance

In problems involving ergodic Markov chains, it is important to estimate the number of steps until the distribution of the chain is close to its stationary distribution. Two common choices are used in literature to evaluate Markov chains convergence, the total variation metric and the Wasserstein metric. In this thesis, the convergence is evaluated using the former measure. The total variation distance which is also called the statistical distance or variational distance is a distance measure for probability distributions.

Definition 1.1. *The total variation distance TV between the probability measures μ and ν is given by:*

$$||\mu - \nu||_{TV} = \sup_{A \subset \omega} |\mu(A) - \nu(A)|$$

where $\mu(A) = \sum_{x \in A} \mu(x)$

Definition 1.2 ((Freedman, 2017))**.** *For the same probability measures μ and ν, the total variation distance TV is defined as:*

$$||\mu - \nu||_{TV} = \frac{1}{2} \sum_{s \in S} |\mu(s) - \nu(s)|$$

Additionally, Freedman (2017) proved in his Convergence theorem, that an ergodic Markov chain converges at an exponential rate to the equilibrium distribution over time.

Theorem 1.1. *If P is the distribution of an irreducible and aperiodic Markov chain, with the target distribution π, then there exist $0 < \alpha < 1$ and $C > 0$ such that*

$$\max_{x \in \omega} ||P^t(x, \cdot) - \pi||_{TV} \leq C\alpha^t,$$

Accordingly, the distance approaches zero when the Markov chain converges to the target distribution π.

Similarly, convergence rate is measured using τ_x represents the mixing time and it is defined as: for $\epsilon > 0$

$$\tau_x(\epsilon) = \min\{t : \Delta_x(t^{'}) \leq \epsilon,\ \forall t^{'} > t\}$$

where a Markov chain is said to be rapidly mixing if $\tau_x(\epsilon)$ is $O(\text{poly}(\log(\frac{N}{\epsilon})))$ (Guruswami, 2000). Moreover, Aldous (1982) proved that a rapid convergence to the stationary distribution could be captured using the spectral gap $(1 - \lambda)$. This later result was also linked to the mixing time quantity τ_x as in the following proposition.

Proposition 1.1 ((Guruswami, 2000))**.** *For a Markov chain, the mixing time quantity $\tau_x(\epsilon)$ satisfies*

- $\tau_x(\epsilon) \leq \big(\frac{1}{1-\lambda_{max}}\big)(\ln \pi(x)^{-1} + \ln(\epsilon)^{-1})$.
- $\max_{x \in \Omega} \tau_x(\epsilon) \geq \frac{1}{2}\lambda_{max}\big(\frac{1}{1-\lambda_{max}}\big) \ln(2\epsilon)^{-1}$.

Consequently, a large gap $(1 - \lambda_{max})$ is necessary to guarantee the convergence to the stationary distribution. Additionally, λ_{N-1} is not important in practice since there are ways to ensure that $\lambda_1 > |\lambda_{N-1}|$, so in general we can just assume that λ_1 is the eigenvalue of interest. Hence, bounds on the second-largest eigenvalue λ_1 are needed to bound the mixing time.

1.3.2 Methods for bounding mixing times

A rich literature exists on Markov chain convergence in total variation distance where various tools have been presented for convergence using this measure. The developed methods are generally divided into probabilistic methods like the known technique called coupling (Jerrum, 1998), in addition to analytic methods like spectral analysis (Diaconis, 1988). Furthermore, geometric methods like path bounds (Jerrum, 1998). For a good comparison between these techniques, see (Guruswami, 2000). Generally, it is needed, but hard, to analyze the spectrum of the chain to obtain bounds on the second largest eigenvalue λ_1. On the contrary, it is more easier to analyze the chain directly without using the spectrum. Following, we present briefly two common and simple approaches, to prove Markov chains rapid mixing, as an introuction for the third approach in our interest because of its importance in this work.

1.3.2.1 Coupling

This classical technique was first presented by Aldous (1983) to prove rapid mixing. This powerful technique is extensively used to bound the total variation distance for Markov chains.

Definition 1.3. *X and Y are two random variables with probability distributions μ and v defined over Ω. The coupling ω is a distribution on $\Omega \times \Omega$ if*

- $\forall x \in \Omega, \quad \sum_{y \in \Omega} \omega(x, y) = \mu(x)$
- $\forall y \in \Omega, \quad \sum_{x \in \Omega} \omega(x, y) = v(y)$

- *For any coupling ω over the distributions μ and v*

$$||\mu - v||_{TV} \leq P(X \neq y),$$

and always there exists a coupling ω such that

$$||\mu - v||_{TV} = P(X \neq y).$$

Consequently, the total variation distance between is upper bounded by $P(X \neq Y)$. Hence, by coupling two Markov chains, the total variation distance is bounded as $||P_t(x,) - P_t(y,)||_{TV} \leq P(X_t \neq Y_t)$. For more details, see (Rosenthal, 1995).

1.3.2.2 Conductance

This method lower bounds the spectral gap $(1 - \lambda_1)$ using the chain geometric properties. The conductance Φ of the Markov chain is given in the following definition.

Definition 1.4. *Let S denotes a subset of states in a Markov chain where $\bar{S} = \Omega - S$ and $\pi(S)$ represents the probability that, at equilibrium, the Markov chain will be at the state S then*

$$\Phi = \min_{S \in \Omega} \frac{Q(S, \bar{S})}{\pi(S)}$$

$Q(S, \bar{S})$ denotes the sum of $Q(x, y)$ over all (x, y) where $Q(x, y) = \pi(x)P(x, y)$. As a result, the conductance of a Markov chain is defined as the minimum conductance over all subsets S with $\pi(S) \leq \frac{1}{2}$.

Similarly to the relationship between the mixing rate of a Markov chain and its eigenvalues, a relationship exists between the conductance and the eigenvalues. The second-largest eigenvalue λ_1 is guaranteed to satisfy the bound

$$1 - 2\Phi \leq \lambda_1 \leq 1 - \frac{\Phi^2}{2}$$

Thus, a large spectral gap necessarily leads to high conductance which yields faster mixing for the Markov chain. For graphs, a small conductance implies that there is a bottleneck that is defined as a subset of states from which it is difficult to move around the chain. However, one bottleneck in the chain does not necessarily imply slow mixing (King, 2003). We recommend (Mihail, 1989) for a comprehensive study of the conductance technique.

1.3.2.3 Canonical paths

Following to the previous technique, another method to bound the Markov chain conductance is called canonical paths. It was introduced by Jerrum and Sinclair in (Jerrum and Sinclair, 1988), (Jerrum and Sinclair, 1989) and (Sinclair and Jerrum, 1989). For a Markov chain, a family of paths in the underlying graph is called the set of canonical paths which includes a path γ_{xy} between each couple of states x, y. Using this method, the conductance could be bounded by obtaining a set of canonical paths that do not overload any transition of the Markov chain. It basically aims to minimize the maximum edge loading ρ by building th set of canonical paths $\Gamma = \gamma_{ij}$ such that no edge e in the graph is overloaded by paths where an overloaded edge is essentially a bottleneck in the Markov chain.

Let ρ denotes the maximum edge loading, also called the path congestion parameter. For the edge $e = (x, y)$, $Q(e)$ represents the weight of edge e and given by $Q(e) = Q(x, y) = w_{xy}$. The maximum loading is given by

$$\rho = \max_{e} = \frac{1}{Q(e)} \sum_{\gamma_{xy} \ni e} \pi(x)\pi(y)$$

Considering the Markov chain as a flow network, then then the quantity $\pi(x)\pi(y)$ represents the amount of units of flow travel from x to y along the canonical path connecting them. Moreover, $\rho(\Gamma)$ represents the maximum overloading of any edge relative to its capacity.

Sinclair (1992) showed that a low congestion of any choice of canonical paths for a reversible Markov chain yields a large value of conductance as

$$\Phi = \frac{1}{2\rho}$$

This result is linked and surely improved the bound on mixing time. In fact it bounds the second-largest eigenvalue λ_1 as

$$\lambda_1 \leq 1 - \frac{1}{8\rho^2}$$

The same author, in his work (Sinclair , 1992), enhanced the bound of λ_1 which was previously obtained in Diaconis and Stroock (1991), where they considered the paths lengths in calculating the maximum loading, by changing the way that the path length was considered

as

$$\rho = \max_{e} = \frac{1}{Q(e)} \sum_{\gamma_{xy} \ni e} \pi(x)\pi(y)|\gamma_{xy}|.$$

Bibliography

A. Freedman. Convergence Theorem for Finite Markov Chains. *Proc. REU 2017* , 2017. Available online: http://math.uchicago.edu/ may/REU2017/REUPapers/Freedman.pdf (accessed on 24 December 2019).

A. Sinclair. Improved Bounds for Mixing Rates of Markov Chains on Combinatorial Structures. *Combinatorics, Probability and Computing,* 1(4), pp. 351–370. ISSN 1469-2163, 0963-5483. 1992. doi:10.1017/S0963548300000390.

A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation,* 82(1), pp. 93–133, 1989.

B. Bollobás. *Graph Theory-An Introductory Course, Graduate Texts in Mathematics.* New York: Heidelberg and Berlin Springer-Verlag. 1979.

B. Bollobás. *Random Graphs.* Cambridge: Cambridge University Press. 2001.

C. Hübler, H. Kriegel, K. Borgwardt and Z. Ghahramani. Metropolis Algorithms for Representative Subgraph Sampling. *In: Proceedings of the 2008 eighth IEEE International Conference on Data Mining,* Dec 15–19, ICDM. IEEE Computer Society, Washington, DC, pp. 283–292, 2008.

D. Aldous. Some inequalities for reversible Markov chains. *Journal of the London Mathematical Society,* 25, pp. 564–576, 1982.

D. Aldous. Random walks on finite groups and rapidly mixing Markov chains. *Séminnaire de Probabilités XVII 1981/82,* Springer Lecture Notes in Mathematics 986, pp. 243–297, 1983.

J. King. Conductance and rapidly mixing Markov chains. *Technical report, University of Waterloo,* 2003.

J. Leskovec and C. Faloutsos. Sampling from Large Graphs. *In: Proceedings Of The 12th Acm Sigkdd International Conference On Knowledge Discovery And Data Mining-Kdd 06,* Philadelphia, Pa, Usa: Acm Press. 2006.

J. Rosenthal. Convergence rates of Markov chains. *SIAM Rev.37,* 387–405, 1995.

K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika* , 57 (1): 97–109. 1970.

M. Jerrum. Mathematical foundations of the Monte Carlo method. *In Probabilistic Methods for Algorithmic Discrete Mathematics, Algorithms and Combinatorics 16,* Springer-Verlag, pp. 116–165, 1998.

M. Jerrum and A. Sinclair. Conductance and the rapid mixing property for Markov chains: the approximation of the permanent resolved (preliminary version). *In Proceedings of the* 20^{th} *Annual ACM Symposium on Theory of Computing,* pp. 235–244, 1988.

M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal on Computing,* 18(6), PP. 1149–1178, 1989.

M. Mihail. Conductance and convergence of markov chains - a combinatorial treatment of expanders. *In Proc. of* 30^{th} *FOCS,* pp. 526–531, 1989.

N. Ahmed, J. Neville, and R. Kompella. Network Sampling Via Edge-Based Node Selection with Graph Induction. *Technical Report CSD TR 11–016,* Computer Science Department, Purdue University. 2011.

N. Ahmed, J. Neville, and R. Kompella. Network Sampling: From Static to Streaming Graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD),* 8(2), 2012. URL: https://arxiv.org/abs/1211.3412

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21 (6): 1087–1092. 1953.

P. Diaconis. *Group Representations in Probability and Statistics.* IMS Lecture Notes – Monograph Series 11. Hayward, CA: Institute of Mathematical Statistics. 1988.

P. Diaconis and D. Stroock. Geometric bounds for eigenvalues of Markov chains. *Annals of Applied Probability,* 1(1), pp. 36–61, 1991.

P. Ebbes, Z. Huang, A. Rangaswamy, H. Thadakamalla and O.R.G.B. Unit. Sampling Large-Scale Social Networks: Insights from Simulated Networks. *In* 18^{th} *Annual Workshop on Information Technologies and Systems,* Paris, France. 2008.

P. Hu and W. Lau. A Survey and Taxonomy of Graph Sampling. *ArXiv13085865 Cs Math Stat,* 2013. Available from: http://arxiv.org/abs/1308.5865

S. Chib, and E. Greenberg. Understanding the Metropolis-Hastings algorithm. *American Statistician* , volume 49, pp. 327–335, 1995.

S. Rehman, A. Khan and S. Fong. Graph Mining: A Survey of Graph Mining Techniques. *In Seventh International Conference on Digital Information Management,* pp. 88–92, 2012. doi:10.1109/ICDIM.2012.6360146.

S. Yoon, S. Lee, H. Yook and Y. Kim. Statistical Properties of Sampled Networks by Random Walks. Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics, 75(4 Pt 2), 046114. 2007. ISSN 1539-3755. doi:10.1103/PhysRevE.75.046114.

V. Guruswami. Rapidly mixing Markov chains: A comparison of techniques. 2000. Available at ftp://theory.lcs.mit.edu/pub/people/venkat/markov-survey.ps

W. Gilks, S. Richardson and D. Speigelhalter. *Markov Chain Monte Carlo in Practice.* Chapman and Hall, 1996.

Chapter 2

A Metropolis-Hastings Sampling of Subtrees in Graphs

This chapter presents two methods to sample uniform subtrees from graphs using Metropolis-Hastings algorithms. One method is an independent Metropolis-Hastings and the other one is a type of add-and-delete MCMC.
In addition to the theoretical contributions, we present simulation studies which confirm the theoretical convergence results on our methods by monitoring the convergence of our Markov chains to the equilibrium distribution.

2.1 Introduction

A graph without any cycle is a forest, a tree $T(V_T, M_T)$ is a connected forest where the order of a tree is its number of vertices $|V_T|$ and the tree size is its number of edges $|M_T|$. Here we consider the uniform distribution over trees $T(V_T, M_T)$ verifying $\mathrm{card}(\mathrm{V_T}) = k + 1$. A rigorous presentation of graph properties could be found in (Bollobas, 1979) and (Bollobas, 2001).

Various approaches that leverage tree mining algorithms were developed as trees are one of the most well studied probabilistic structures in graphs. Over the past decade, trees have found a surprising number of applications in Internet and computer science like, for example,

XML which is a markup language designed to store and transport data. For instance, XML data is very popular because of the nature of its tree structure and as a result it is necessary to develop methods that can treat and extract patterns from this type of data like, for example, tracking down the common trees existing among a set of such data. Additionally, it has been heavily researched in biology and bioinformatic where trees are widely used to represent various biological structures like glycans, RNAs, and phylogenies. For example, Shapiro and Zhan (1990) studied the function of the RNA where the RNA structures were collected in trees in order to compare any newly sequenced RNA to compare the similarities in the topological patterns. Takigawa *et al.* (2010) represented glycans as directed trees in which nodes are monosaccharides and edges are linkages and proposed an efficient method for mining frequent and statistically significant subtrees from glycan trees. For a good survey of trees mining algorithms in biology, see (Parthasarathy *et al.* , 2010). In a work of Hancock *et al.* (2012) they sampled pathways of genes within significantly coordinated expression profiles. They aimed at identifying the important metabolites which are driving the function of the network by comparing these metabolites to the metabolites in sampled pathways. As a consequence, a questions arose and motivated to search for another bigger structure that can better identify these metabolites.

Although there are many known methods in literature to sample from a graph, most are designed either to sample subgraphs that are representative to the original graph according to the graph properties as in the work of Hübler *et al.* (2008) or to sample frequent structure patterns as in (Yan and Han , 2002) which are not the concern of this work. Additionally, using most of these techniques generally is not efficient to sample subtrees specifically because these subtrees will represent a small proportion of the sampled structures. This work contributes in the thesis to present two techniques to sample trees according to a distribution from a graph where the vertices are labelled, i.e the tree structure, a.k.a pattern, is not taken into consideration in the developed methods as well as the graph properties.

2.2 Sampling Methods

Sampling uniformly subtrees of a given size that are obtained from an initial graph is not a trivial problem. A way to solve this problem is to use Markov chain Monte Carlo (MCMC) sampling (For general references on Markov chains and Markov chain Monte Carlo see (Kemeny and Snell , 1983), (Robert and Casella , 2009) and (Brooks *et al.* , 2011)). Here, for this purpose, a Metropolis-Hastings (M-H) dynamics are presented.
From chapter 1, the M-H algorithm is an iterative procedure that simulates a Markov chain. If the simulated Markov chain is irreducible and aperiodic, as the configuration space of the chain is finite, the algorithm is convergent. More precisely, this means that the outputs of the algorithm are asymptotically distributed according to the unique invariant distribution of the simulated Markov chain (Häggström, 2002).

The principle of the M-H algorithm is the following: let π be a distribution of interest and consider x^t the current state of the chain. A new candidate x^* is proposed according to a proposal distribution $q(x^t, x^*)$. The proposed candidate is accepted as the new state of the chain, i.e $x^{t+1} = x^*$ with probability :

$$\alpha = \frac{\pi(x^*)}{\pi(x^t)} \frac{q(x^*, x^t)}{q(x^t, x^*)} \wedge 1 = \frac{q(x^*, x^t)}{q(x^t, x^*)} \wedge 1$$

Two methods are presented for sampling uniformly trees with k edges from an undirected and unweighted graph. The first method samples uniform trees according to an independent Metropolis-Hastings, whereas the second method does it according to a non-independent Metropolis-Hastings algorithm.

The defined Markov chain (X^t) is produced through the transition kernel:

$$X^{t+1} = \begin{cases} x^* \text{ with probability } = \alpha \\ x^t \text{ with probability } = 1 - \alpha \end{cases}$$

For the independent method, a special case of the Metropolis-Hastings algorithm is used where the candidate x^* is independent of the present state of the chain x^t so $q(x^*, x^t) = q(x^*)$

and the transition kernel:

$$X^{t+1} = \begin{cases} x^* \text{ with probability } \alpha = \frac{q(x^*)}{q(x^t)} \wedge 1 \\ x^t \text{ with probability } 1 - \alpha \end{cases}$$

In what follows, first each method is detailed and then their convergence speeds are studied.

2.2.1 First Method: Independent Uniform Trees

In the following, the proposal distribution for the independent sampler is presented. This method generates the candidate tree x^* in the following way. A vertex v_{i_1} is selected randomly from the original graph and receives a weight $w_{i_1} = k$, next this weight w_{i_1} is distributed among all neighbours in the graph so that each vertex receives a weight, then vertices with weight greater than 0 are selected as neighbours for the first vertex in the generated subtree. The weight k represents the number of vertices we can connect for the whole subtree after choosing a given vertex, this weight is distributed on the selected neighbours according to a uniform multinomial distribution with equal probabilities. This process is conducted iteratively until no weight is left anywhere.

Algorithm 1 presents the detailed steps to generate a tree T, it is important to note that A is an ordered set where the last entered vertices are the smallest in the ordering sense.

Algorithm 1 Generate a random tree T

uniform selection of v among V
$T \leftarrow v$
$w(v) \leftarrow k + 1$
the ordered set of active vertices $A \leftarrow v$
while $A \neq \emptyset$ **do**
  select the first vertex v in A
  $A \leftarrow A \setminus v$
  let $n(v) = v_{i_1}, \ldots, v_{i_{|n(v)|}}$ be the neighbours of v in the graph not already selected in the tree
  if $n(v) = \emptyset$ and $w(v) > 1$ **then**
    the algorithm will stop and relaunch again from the beginning
  else
    the weight $w(v) - 1$ is distributed among $n(v)$ using a multinomial law $M(w(v) - 1, |n(v)|^{-1}, \ldots, |n(v)|^{-1})$
    for all $v^* \in n(v)$ **do**
      if $w(v^*) > 0$ **then**
        $A \leftarrow A \cup v^*$
        $T \leftarrow T \cup v^*$
      end if
    end for
  end if
end while

To compute $p(T)$, the probability of generating a subtree T, start from any vertex v of T and compute the probability of generating the subtree T starting from it. Algorithm 2 allows to compute the probability of generating a subtree as detailed in its description. $n_T(v)$ denotes the neighbours of v in subtree T. Also, $p(x^*)$ must be computed to take into account all the possible ways of generating x^*.

Algorithm 2 Compute the probability $p(T)$ of generating T

$p(T) \leftarrow 0$
for all $v \in T$ **do**
  $T^{'} \leftarrow T \setminus v$
  compute $w(v)$
  $p \leftarrow 1$ and $A \leftarrow v$
  for all $v \in A$ **do**
    $A \leftarrow A \setminus v$
    $p \leftarrow p \times \frac{(w(v)-1)!}{w(v_i)!...w(v_{|n_T(v)|})!} \frac{1}{|n_T(v)|^{w(v)-1}}$
    $A \leftarrow A \cup \{v_1 \dots v_{n_T(v)}\}$
  end for
  $p(T) \leftarrow p(T) + p$
end for

The weights used to compute the probability of generating a subtree are computed as shown in algorithm 3.

Algorithm 3 Compute $w(v, T^{'})$

$w(v) \leftarrow 1$
for all $v_i \in n_T(v) \cap T^{'}$ **do**
  $T^{'} \leftarrow T^{'} \setminus v_i$
  compute weight $w(v_i)$
  $w(v) \leftarrow w(v) + w(v_i)$
end for

The chain is clearly irreducible as each subtree has a positive probability of being sampled at each step. This also implies the aperiodicity since it is always possible to stay at the same state.

Bollobas (2001) showed that for $r \geq 2$ and $n \geq 3$, if Y_i is the number of cycles of length at most i in a $r-$regular graph generated by the Erdős-Rényi random graph, then $Y_3, Y_4, ..., Y_n$ are asymptotically independent Poisson random variables with mean $\lambda_i = (r-1)^i/2i$. We

assume henceforth that these random variables follow a Poisson distribution, indeed in practical applications the graph size is often large. Under these assumptions the following result gives a bound on the speed of convergence for the independent Markov chain sampler.

Theorem 2.1. *For a random $r-$regular graph in Erdős-Rényi random graph model, suppose that the number of cycles of length at most $k+1$ distributes following to a Poisson distribution. Let the vertex degree $r \geq 2$ and each vertex is contained in a cycle of length at least $k+1$, we have for any starting state x:*

$$\| M_x^m - \pi \|^2 \leq \frac{1}{4\pi(x)} \left(1 - \frac{k+1}{nr^{k(k+1)/2}} \left(\frac{(r-1)^k}{2} \left(\frac{r-1}{k+1} - \frac{1}{k} \right) - 2\sqrt{\frac{(r-1)^{k+1}}{\alpha(k+1)}} \right) \right)^{2m}$$

with a probability larger than $1-\alpha$, where M_x^m is the m^{th} updated distribution and π is the target distribution.

Proof. For the convergence, we can use the bound on the total variation distance:

$$\| M_x^m - \pi \|^2 \leq \frac{1}{4\pi(x)} (1 - u(1))^{2m}$$

with $u(1) = \min(p(x)/\pi(x))$ (Liu, 1996). As $\pi(x)$ is the inverse of the number of subtrees, we need a lower bound on the number of subtrees of a given size k in a graph.

The number of subtrees can be bounded considering that a tree is obtained whenever a cycle of length $k+1$ is deprived of an edge, in this way we can obtain $k+1$ different subtrees of size k. Moreover, if we can have c such cycles we are able of generating $(k+1)c$ subtrees, this is the way we follow to lower bound the number of subtrees although it is a fact that the true number of subtrees in a graph is bigger than the number of subtrees generated in our way.

Let Y_{k+1} denote the number of cycles of length at most $k+1$. Let us denote by F the following event: $|Y_{k+1} - Y_k - (E(Y_{k+1}) - E(Y_k))| < \epsilon$ then the number of cycles of size exactly $k+1$ is $Y_{k+1} - Y_k$ which verifies:

$$P(F) \geq P\left(\left| Y_{k+1} - \frac{(r-1)^{k+1}}{2(k+1)} \right| < \epsilon/2 \cap \left| Y_k - \frac{(r-1)^k}{2k} \right| < \epsilon/2 \right)$$

$$\Leftrightarrow P\left(F^C\right) \leq P\left(\left| Y_{k+1} - \frac{(r-1)^{k+1}}{2(k+1)} \right| < \epsilon/2 \cap \left| Y_k - \frac{(r-1)^k}{2k} \right| < \epsilon/2 \right)^C$$

$$\Leftrightarrow P(F^C) \leq P\left(\left|Y_{k+1} - \frac{(r-1)^{k+1}}{2(k+1)}\right| > \epsilon/2\right) + P\left(\left|Y_k - \frac{(r-1)^k}{2k}\right| > \epsilon/2\right) -$$
$$P\left(\left|Y_{k+1} - \frac{(r-1)^{k+1}}{2(k+1)}\right| > \epsilon/2 \cap \left|Y_k - \frac{(r-1)^k}{2k}\right| > \epsilon/2\right)$$
$$\Rightarrow P(F^C) \leq \frac{2(r-1)^{k+1}}{(k+1)\epsilon^2} + \frac{2(r-1)^k}{k\epsilon^2} \text{ using Chebyshev inequality}$$
$$\Rightarrow P(F^C) \leq \frac{4(r-1)^{k+1}}{(k+1)\epsilon^2}$$
$$\Rightarrow P(F) \geq 1 - \frac{4(r-1)^{k+1}}{(k+1)\epsilon^2} \tag{2.1}$$
$$\text{let } \epsilon = \sqrt{\frac{4(r-1)^{k+1}}{(k+1)\alpha}}, \tag{2.2}$$

then we can conclude that with probability larger than $1-\alpha$:

$$\left|Y_{k+1} - Y_k - \left(\frac{(r-1)^{k+1}}{2(k+1)} - \frac{(r-1)^k}{2k}\right)\right| \leq 2\sqrt{\frac{(r-1)^{k+1}}{\alpha(k+1)}}$$
$$\Rightarrow \left|Y_{k+1} - Y_k - \frac{(r-1)^k}{2}\left(\frac{r-1}{k+1} - \frac{1}{k}\right)\right| \leq 2\sqrt{\frac{(r-1)^{k+1}}{\alpha(k+1)}}$$
$$Y_{k+1} - Y_k \geq \frac{(r-1)^k}{2}\left(\frac{r-1}{k+1} - \frac{1}{k}\right) - 2\sqrt{\frac{(r-1)^{k+1}}{\alpha(k+1)}} \tag{2.3}$$

Let S denote the set of all possible subtrees of size k that can be obtained from our graph. A given tree of size k can be generated for the proposal distribution in $k+1$ different ways when starting from any vertex in the tree. Additionally, each way involves l steps ($l \in 1, \ldots, k$) representing number of multinomial steps, at least 1 if all weights are distributed at once in a star-like manner, and at most k if weights are given every time to a single vertex producing thus a path of length $k+1$. As a consequence, each subtree has a proposal probability of the form:

$$\frac{1}{n}\prod_{i=1}^{l}\frac{w_i!}{w_{i1}!\ldots w_{ir_i}!}\frac{1}{r_i^{w_i}} \quad \text{where } i \leq r$$
$$\geq \frac{1}{n}\prod_{i=1}^{k}\frac{1}{r_i^{w_i}} \text{ with } w_i = w_{i1} + \cdots + w_{ir_i}$$
$$\geq \frac{1}{nr^{\sum_i w_i}}$$
$$\geq \frac{1}{n}\frac{1}{r^{k(k+1)/2}}$$

Thus the probability $p(T)$ of generating a tree is lower bounded by:

$$\frac{1}{n}\frac{k+1}{r^{k(k+1)/2}} \tag{2.4}$$

Gathering results of equations 2.3 and 2.4 we obtain:

$$\begin{aligned} \| M_x^m - \pi \|^2 &\leq \frac{1}{4\pi(x)}(1-u(1))^{2m} \\ &\leq \frac{1}{4\pi(x)}\left(1 - \frac{k+1}{nr^{k(k+1)/2}}\left(\frac{(r-1)^k}{2}\left(\frac{r-1}{k+1}-\frac{1}{k}\right) - 2\sqrt{\frac{(r-1)^{k+1}}{\alpha(k+1)}}\right)\right)^{2m} \end{aligned}$$

□

Clearly, this bound goes to 0 while assuming that the graph is large; i.e the number of vertices $|V| = n$ is large. We assumed that each vertex is contained in a cycle of length $k+1$ to guarantee that the proposed subtree x^* can be obtained from any of its vertices. In general, the success rate s of algorithm 1 will be strictly lower than 1. It is worth to mention that s is not the Metropolis-Hastings acceptance rate to avoid misleading interpretations. More precisely, algorithm 1 failure results from a shortage of the number of required neighbours for the selected vertex in comparison to the given weight of this vertex that will be distributed among neighbours which will cause algorithm 1 to stop and start again from the beginning. The success rate could be estimated as $s \approx$ (number of acceptances)/(number of steps) which is normally close to 1 unless the border is large as on expander graphs for example.

Theorem 2.2. *For the same settings defined in Theorem 2.1, let s represents the algorithm success rate. For a random $r-$regular graph, where $r \geq 2$, we have for any starting state x:*

$$\| M_x^m - \pi \|^2 \leq \frac{1}{4\pi(x)}\left(1 - \frac{s(k+1)}{nr^{k(k+1)/2}}\left(\frac{(r-1)^k}{2}\left(\frac{r-1}{k+1}-\frac{1}{k}\right) - 2\sqrt{\frac{(r-1)^{k+1}}{\alpha(k+1)}}\right)\right)^{2m}$$

with a probability larger than $1-\alpha$, where M_x^m is the m^{th} updated distribution and π is the target distribution.

Proof. Again, using the total variation distance to bound the convergence:

$$\| M_x^m - \pi \|^2 \leq \frac{1}{4\pi(x)}(1-u(1))^{2m}$$

with $u(1) = \min(p(x)/\pi(x))$ (Liu, 1996). If we ignore the constraint in Theorem 2.1 stating that each vertex is contained in a cycle of length at least $k+1$, then it is possible at any step during the process that the distributed weight among the chosen vertex is greater than the number of neighbours for this vertex. Consequently, algorithm 1 will stop and start again from another vertex.

The probability of generating a subtree needs to be adjusted, for this, first we compute it conditionally on the success of algorithm 1. It is bounded by

$$\frac{1}{nc}\prod_{i=1}^{k}\frac{w_i!}{w_{i1}!\dots w_{ir}!}\frac{1}{r_i^{w_i}} \geq \frac{1}{n}\prod_{i=1}^{k}\frac{w_i!}{w_{i1}!\dots w_{ir}!}\frac{1}{r_i^{w_i}} \geq \frac{1}{n}\frac{k+1}{r^{k(k+1)/2}}$$

where c is the normalizing constant upper bounded by 1 as the distribution of weights is constrained by the success of algorithm 1. Next, we can use again bound 2.3 on the number of cycles as well as the success rate s to obtain:

$$\begin{aligned}\parallel M_x^m - \pi \parallel^2 &\leq \frac{1}{4\pi(x)}(1-u(1))^{2m} \\ &\leq \frac{1}{4\pi(x)}\left(1 - \frac{s(k+1)}{nr^{k(k+1)/2}}\left(\frac{(r-1)^k}{2}\left(\frac{r-1}{k+1}-\frac{1}{k}\right) - 2\sqrt{\frac{(r-1)^{k+1}}{\alpha(k+1)}}\right)\right)^{2m}\end{aligned}$$

□

The convergence speed for this chain is $\mathcal{O}((s^{-1}nr^{\frac{k^2}{2}}))$.

Corollary 2.1. *For the random $r-$regular graph in Theorem 2.1, with a probability larger than $1-\alpha$, the method restricted to path sampling verifies:*

$$\parallel M_x^m - \pi \parallel^2 \leq \frac{1}{4\pi(x)}\left(1 - \frac{2}{nr^{k(k+1)/2}}\left(\frac{(r-1)^k}{2}\left(\frac{r-1}{k+1}-\frac{1}{k}\right) - 2\sqrt{\frac{(r-1)^{k+1}}{\alpha(k+1)}}\right)\right)^{2m}$$

Proof. The only thing to do is to replace factor $k+1$ by 2 in the proof of Theorem 2.1 in equation 2.4 as there can be only two starting points for a path. □

2.2.2 Second Method: Crawling Uniform Trees

As a first step, we initialize the algorithm by selecting a random vertex and adding k edges greedily to build a tree of order $k+1$. Following, the crawling method modifies the initial subtree by shifting randomly one of its edges, i.e an edge is deleted randomly and another

is randomly added. The Metropolis-Hastings acceptance rate is:

$$\frac{q(x^*, x^t)}{q(x^t, x^*)} \wedge 1 \text{ with } q(x^t, x^*) = \begin{cases} \frac{1}{2}\frac{1}{|n_{G'}(x^t)|} & \text{if } x^* \neq x^t \\ \frac{1}{2} & \text{if } x^* = x^t \end{cases}$$

where $n_{G'}(x^t)$ denotes the set of neighbours for the tree x^t in the graph $G'(V', E')$ representing the Markov chain state space. Let $n_{x^t}(v_i)$ denotes the set of neighbour vertices inside the tree x^t for all vertices belonging to the tree x^t, l_{x^t} represents the set of leaf vertices in the tree which are vertices connected to only one neighbour vertex in the tree x^t and finally let $n_G(v_i)$ be the set of neighbours for the tree leaves inside the graph $G(V, E)$ which don't belong to the tree x^t. Algorithm 4 presents the steps to generate a tree x^* from a randomly initialized tree x^t.

Algorithm 4 Generate a tree x^{t+1}

Require: A tree x^t where $V_T = \{v_1, v_2, \ldots, v_{k+1}\}$ and $E_T = \{e_1, e_2, \ldots, e_k\}$
  Initialize $l_{x^t} = \{v_i \in x^t \text{ s.t } |n_{x^t}(v_i)| = 1\}$
  Initialize $n_G(x^t) = \{v_j \in V \backslash V_T \ s.t. \ e = (v_i, v_j) \in E \text{ where } v_i \in x^t \ \& \ v_j \notin x^t\}$
  while not mixed **do**
    Sample v_i in l_{x^t}
    $x^t \leftarrow x^t \backslash \{v_i\}$
    Update $n_G(x^t)$
    Sample v_j in $n_G(x^t)$
    $x^{t+1} \leftarrow x^t \cup \{v_j\}$ with probability $\alpha \leftarrow min\left\{1, \frac{q(x^{t+1}, x^t)}{q(x^t, x^{t+1})}\right\}$
    $x^{t+1} \leftarrow x^t \cup \{v_i\}$ with probability $1 - \alpha$
  end while

This chain is aperiodic since $q(x^t, x^*) > 0$ for all states x^* which allow it to stay at the same state with a positive probability. Also, it is irreducible as will be seen along the proof of theorem 2.3.

In order to bound the mixing time of a Markov chain, we will use the second largest eigenvalue λ_1 by making the Markov chain lazy where a lazy chain stays in the current state at each step with probability at least 1/2. Factor 1/2 is a sufficient condition for the transition probability matrices to have only positive eigenvalues.

First, let us recall that the Cheeger constant for graph $G(V, E)$ is defined as

$$h = \min_S \frac{|E(S, \bar{S})|}{\min\{\text{vol}(S), \text{vol}(\bar{S})\}}$$

where $\text{vol}(S)$ stands for the sum of degrees of vertices in S and $|E(S, \bar{S})|$ for the number of

edges between S and $\bar{S}$. Henceforth, S will denote the subset of V realizing this minimum and, without loss of generality, we consider that $S = \min\{S, \bar{S}\}$ so

$$h = \frac{|E(S, \bar{S})|}{\text{vol}(S)}.$$

Lemma 2.1. *Let h represents the Cheeger constant for the graph $G(V, E)$ and $vol(V)$ denotes the sum of degrees of vertices in V, then there exists a set of paths Γ where b is the maximum number of paths containing an edge e such that $b \leq \lfloor \frac{\text{vol}(V)}{h} \rfloor + 1$.*

Proof. This is proven by contradiction. First, let us recall that a path is a sequence of edges which connect a sequence of distinct vertices. For convenience, c will stand for $\lfloor \text{vol}(V)/h \rfloor$. Let us suppose that whichever the set of paths Γ there is always an edge e with at least $c+2$ paths containing it. Then, e is an edge between two vertices a and b. Let us suppose that there is a path p between a and b with edges supporting less than $c+1$ paths, then edge e could have less than $c+2$ paths crossing it by replacing a path $x \to a \to b \to y$ which crosses e by the new path $x \to a \to p \to b \to y$. As a result, in any path between a and b, one will meet at some point an edge supporting at least $c+1$ paths. The consequence of it, is that there is a graph cut between a and b containing only edges with at least $c+1$ paths. This graph cut creates two subgraphs, A and B containing respectively a and b. We show now that this leads to a contradiction. Indeed, by definition of S

$$\begin{aligned}
&\frac{|E(S, \bar{S})|}{\text{vol}(S)} \leq \frac{|E(A, B)|}{\text{vol}(A)} \text{ if, without loss of generality, } \text{vol}(A) \leq \text{vol}(B) \\
&\Rightarrow \frac{\text{vol}(A)\text{vol}(B)}{|E(A, B)|} \leq \frac{\text{vol}(S)\text{vol}(B)}{|E(S, \bar{S})|} \\
&\Rightarrow \frac{|A|.|B|}{|E(A, B)|} \leq \left\lfloor \frac{\text{vol}(V)}{h} \right\rfloor + 1 \qquad (2.5)
\end{aligned}$$

By hypothesis, each edge in $|E(A, B)|$ belongs to at least $c+1$ paths and in particular e supports $c+2$ paths. It is then immediate, that the mean number of paths per edge belonging to $|E(A, B)|$ is strictly greater than $c+1$ which is impossible when observing the last equation. This concludes the proof. □

Theorem 2.3. *Let a be the maximum number of subtrees associated to a vertex , d_{max} is the graph maximum degree, D is the graph diameter and λ_1 is the second eigenvalue on the graph then for any starting state x such that M_x^m is the m^{th} updated distribution and π is*

the target distribution we have:

$$\| M_x^m - \pi \|^2 \leq \frac{1}{4\pi(x)} \left(1 - \frac{1}{K}\right)^{2m}$$

with $K \leq 2a^2 d_{max} k^2 (k+1)^2 (D+k) \left(\frac{2d_{max}}{\lambda_1} + 1\right)$.

Proof. Again, let $G^{'}(V^{'}, E^{'})$ be the Markov chain state space or simply the graph of subtrees with k edges. According to Sinclair (1992), the Markov chain has the following bound on its second eigenvalue:

$$\lambda_1 \leq 1 - \frac{1}{K} \text{ with } K = \max_e Q(e)^{-1} \sum_{\gamma_{xy} \ni e} |\gamma_{xy}| \pi(x) \pi(y)$$

where $|\gamma_{xy}|$ stands for the length of the path γ_{xy} and $Q(e) = \pi(e_1) P(e_1, e_2)$ if e is the edge (e_1, e_2).

K can be bounded in the following way

$$\begin{aligned} K &\leq \max_e \frac{1}{\pi(e_1) P(e_1, e_2)} \sum_{\gamma_{xy} \ni e} \pi(x) \pi(y) D' \text{ where } D' \text{ is the diameter of } G'(V', E') \\ &\leq \pi(x) b' D' \max_e \frac{1}{P(e_1, e_2)} \end{aligned} \tag{2.6}$$

where $P(e_1, e_2)$ is the transition matrix and b' stands for the maximum number of paths crossing an edge e in the graph $G'(V', E')$.

At this point, we have to make $P(e_1, e_2)$ more explicit, indeed $Q(e_1, e_2)$ is the Markov chain related to the Metropolis-Hastings algorithm being used, that is:

$$\begin{aligned} P(e_1, e_2) &= \left(\frac{q(e_2, e_1)}{q(e_1, e_2)} \wedge 1\right) q(e_1, e_2) \text{ where } q(e_1, e_2) \text{ is the proposal law} \\ &= q(e_1, e_2) \wedge q(e_2, e_1) \\ &\geq \frac{1}{2|n_{G'}(x^t)|} \\ &\geq \frac{1}{2k^2 d_{max}} \end{aligned} \tag{2.7}$$

where the probability $\frac{1}{2}$ is due to the laziness. Thus,

$$K \leq 2\pi(x) k^2 d_{max} b' D' \tag{2.8}$$

First, let us start by bounding the value of $b^{'}$. Here we need to use conductance information

on the graph $G(V,E)$ given by b. The set of paths Γ' on $G^{'}(V^{'},E^{'})$ is derived from the set of paths Γ in $G(V,E)$ in the following way. To each vertex $v \in V$ is associated the set of subtrees containing it and denoted by $\mathcal{T}(v)$. Hence, if we want to have a path between two subtrees s and t, we look for $v_s \in V$ and $v_t \in V$ such that $s \in \mathcal{T}(v_s)$ and $t \in \mathcal{T}(v_t)$. Therefore, if there is a path $v_s = v_1 \to v_2 \cdots \to v_n = v_t$ between v_s and v_t, there is an associated path between s and t, denoted by $s = t(v_1) \to t(v_2) \to \cdots \to t(v_n)$ where $t(v_i) \in \mathcal{T}(v_i)$. This is possible as $v_i \sim v_{i+1}$, so to obtain $t(v_{i+1})$ connected to $t(v_i)$ it is enough to remove the farthest edge of $t(v_i)$ to $t(v_{i+1})$ and replace it by the edge (v_i, v_{i+1}). By the way, this shows that $D^{'} \leq D+k$. The additional term is needed because it may happen that the first subtree reaching v_t is different from t so that additional moves are needed, to a maximum of k.
Now, if we denote by a the maximum number of subtrees associated to a vertex we can bound b' noting that an edge e' of $G'(V',E')$ will be crossed as many times as there are paths in Γ' associated to paths in Γ crossing edges (v_i, v_j) where v_i is a vertex of s and v_j a vertex of t. That makes $(k+1)^2$ pairs multiplied by the number of times these pairs can be used, so

$$b^{'} \leq b(k+1)^2a^2. \tag{2.9}$$

Accordingly, the bound on b of lemma 2.1 can be injected and we get

$$b^{'} \leq (k+1)^2a^2\left(\frac{\text{vol}(V)}{h}+1\right).$$

Besides, by Cheeger inequality $\lambda_1 \leq 2h$ so that

$$b^{'} \leq (k+1)^2a^2\left(\frac{2\text{vol}(V)}{\lambda_1}+1\right). \tag{2.10}$$

Consequently, injecting bound $D' \leq D+k$ and equation 2.10 into 2.8 we get

$$\begin{aligned} K &\leq 2\pi(x)k^2d_{max}(k+1)^2a^2\left(\frac{2\text{vol}(V)}{\lambda_1}+1\right)(D+k) \\ &\leq 2\pi(x)d_{max}k^2(k+1)^2a^2(D+k)\left(\frac{2d_{max}\pi(x)^{-1}}{\lambda_1}+1\right) \\ &\leq 2d_{max}k^2(k+1)^2a^2(D+k)\left(\frac{2d_{max}}{\lambda_1}+1\right) \end{aligned}$$

□

2.3 Experimental Evaluation

In this section we examine the theoretical results using simulations on three different types of graphs: Erdős-Rényi graph, regular graph and a barbell graph variant. The barbell graph consists of two connected grids, each grid contains a finite number of adjacent connected vertices where the connections between any two adjacent vertices represent an undirected edge.

We were careful to generate grids with different structures for the same graph, i.e any vertex in the first grid can be connected to another vertex only in a horizontal or vertical direction so the maximum number of edges touching a vertex is 4 whereas any vertex in the second grid can touch a maximum number of 8 neighbour vertices since it is possible for it to connect also diagonally, this is justified because we want to detect the behavior of the sampling methods with different structures.

During the simulations, we sampled subtrees of size $5, 10$ and 20 from graphs of size between 1000 and 1 million. To guarantee the efficiency of theses simulations, a burn-in period of size 1000 iterations was applied and we sampled only one sample each 1000 iterations.

2.3.1 Sampling from Erdős-Rényi graph

Erdős-Rényi graph $G(n, p)$ is constructed by connecting nodes randomly where each edge is included in the graph with probability p independently from every other edge. In our graph, we fixed the number of vertices and edges $|V| = 60000$ and $|E| = 600000$, respectively. This graph is generated with $p = (2|E|)/(|V||V - 1|) \approx 0.00033$.

Figures 2.1 and 2.2 present the ACF plots of the diameter and the number of leaves for sampled subtrees using both methods and for different subtrees sizes, respectively. For the same number of iterations, the convergence of the crawling method was quicker than the independent method. It is also clear that the mixing time was longer when increasing the subtree size for both methods.

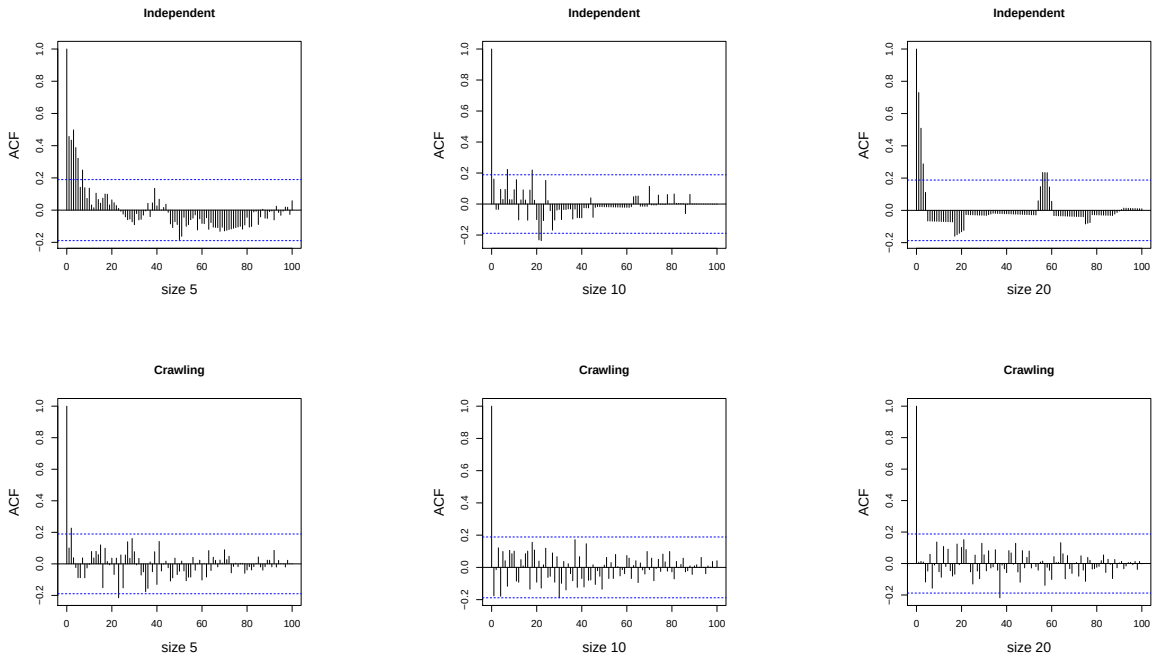

Figure 2.1: The ACF for the diameter of subtrees sampled using both methods

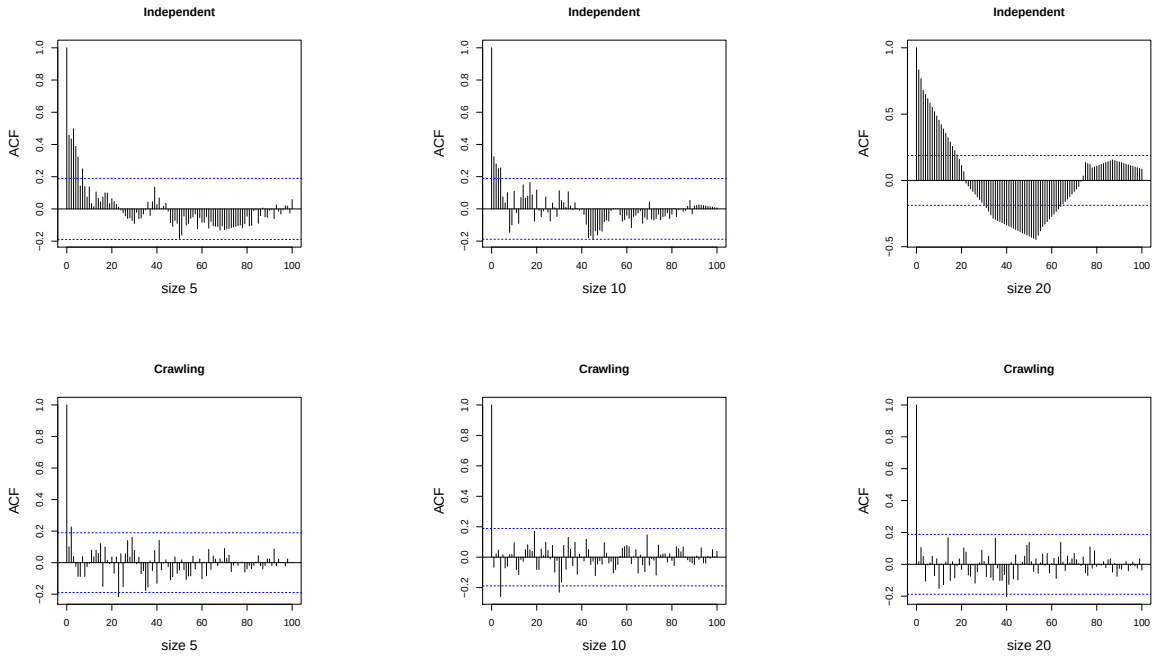

Figure 2.2: The ACF for the number of leaves of subtrees sampled using both methods

2.3.2 Sampling from $r-$Regular Graph

Both sampling methods were applied on a special case of graphs where all edges have the same degree r to see the effect of the vertex degree on the convergence speed. In this graph, we fixed the number of vertices $|V| = 100000$ and the degree $r = 40$.

Figures 2.3 and 2.4 present the ACF plots for the diameter and the number of leaves. The crawling algorithm again converged quicker in all presented cases. Unlike the crawling method, the mixing speed of the independent algorithm was slower when the subtree size increased which confirms our theoretical result showing the effect of the size on the convergence speed of the independent method.

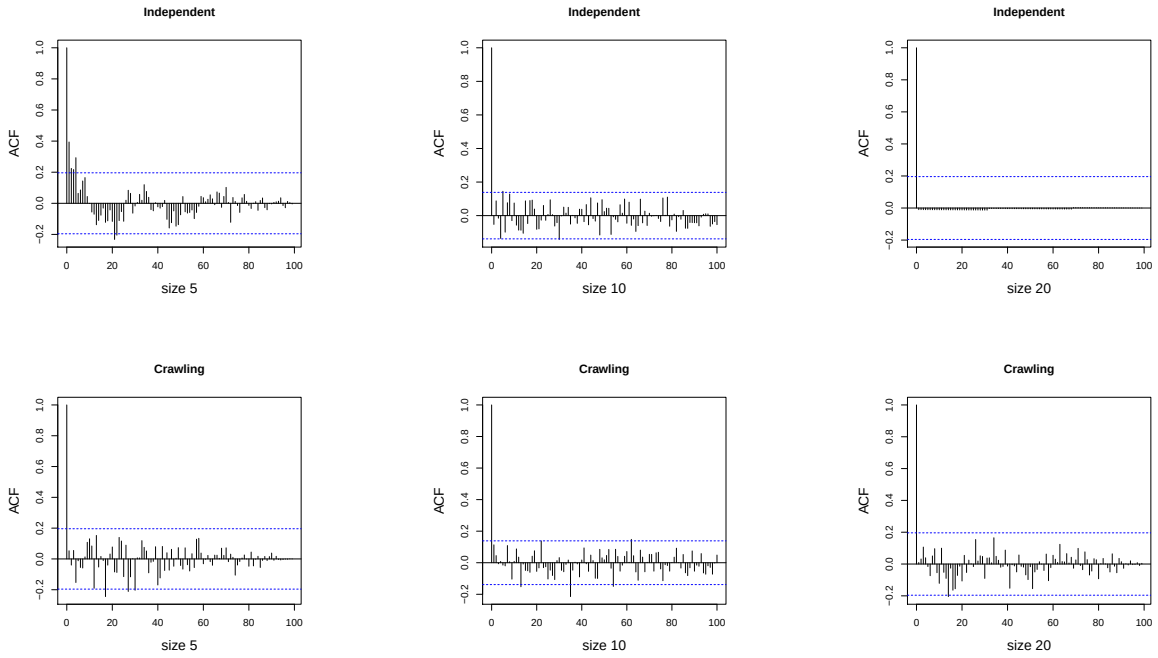

Figure 2.3: The ACF of the diameter of subtrees sampled using both methods

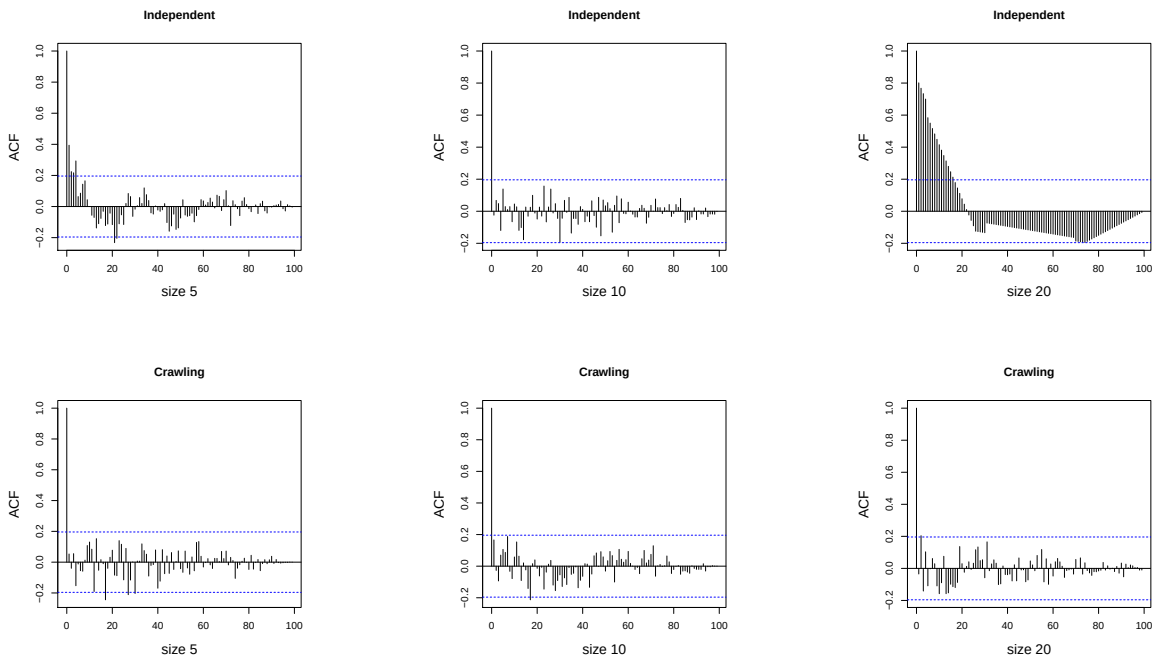

Figure 2.4: The ACF plot for the number of leaves of subtrees sampled using both methods

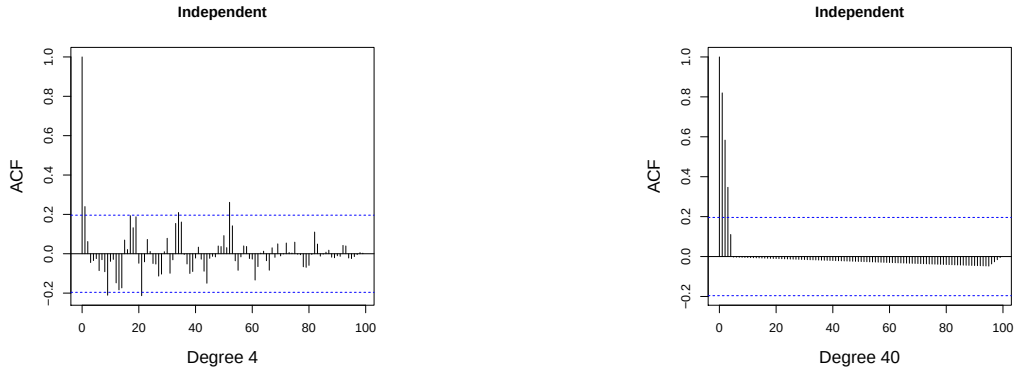

Figure 2.5: The ACF plot for the diameter of subtrees sampled by the independent methods from two graphs with different vertex degree

Figure 2.5 above presents the ACF for the diameter of subtrees of the same size sampled from graphs with different vertex degrees. When sampling from a $4-$regular graph, it is clear that the chain converged quickly unlike the case when we increased the degree, $r = 40$, then chain took more time to converge. This result is compatible to the one achieved in the theoretical part which proved the effect of the vertex degree r in the $r-$regular graphs on

the convergence speed of the independent method and it was assured that the convergence speed is at most $\mathcal{O}(nr^{k^2/2})$.

A simulation has been performed on a complete graph, which is the worst case for the independent method, in that case only star subgraphs have been produced.

2.3.3 Sampling from a barbell graph variant

This part was designed to study the effect of the graph's bottleneck on the efficiency of the presented sampling methods. Many cases were considered in this part to monitor the effect of the bottleneck on the mixing time. More precisely, we connected both grids, to construct the graph, by only one edge, 17 edges and then 34 edges and each grid consists of $(174 * 174)$ vertices.

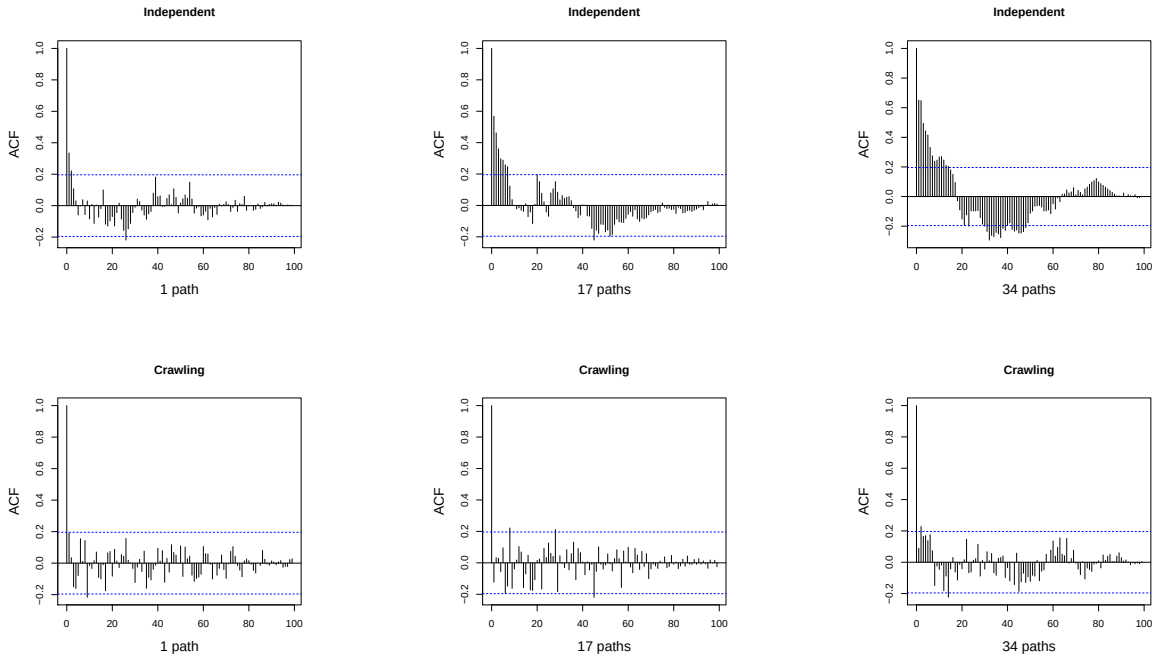

Figure 2.6: The ACF for the diameter of subtrees sampled using both methods from a graph consists of two different grids connected by only one edge, 17 edges and 34 edges respectively

Although the ACF in the plots of the diameter presented in Figure 2.6 seems to converge

when sampling using the crawling method, it is not in reality. This result was concluded after reviewing the sampled subtrees. Indeed, all subtrees were sampled only from the first grid of the graph which means that the chain didn't move to the second grid so the resulted sample does not represent the whole set of graph subtrees of size k.

For the independent method, although it was successful to sample subtrees from this type of graphs it is clear that the chain took more time to converge when increasing the number of edges between both grids. The reason of this behaviour is that the structure of the graph is more complicated for the independent method and consequently to reach convergence the chain needs to run for more iterations.

From the following Figures 2.7 and 2.8, we can see the effect of the subtree size on the convergence speed of the independent method. On the other side, we didn't consider the plots of the ACF when sampling using the crawling method because of the same reason clarified above which is related to the bottleneck.

The crawling algorithm was more successful in sampling subtrees of different sizes from all type of graphs except from the grid graph, where the sampled subtrees originated from the first grid and it was hard for the chain to move to the second grid. Moreover, we see that both methods took more mixing time when we increased the subtree size.

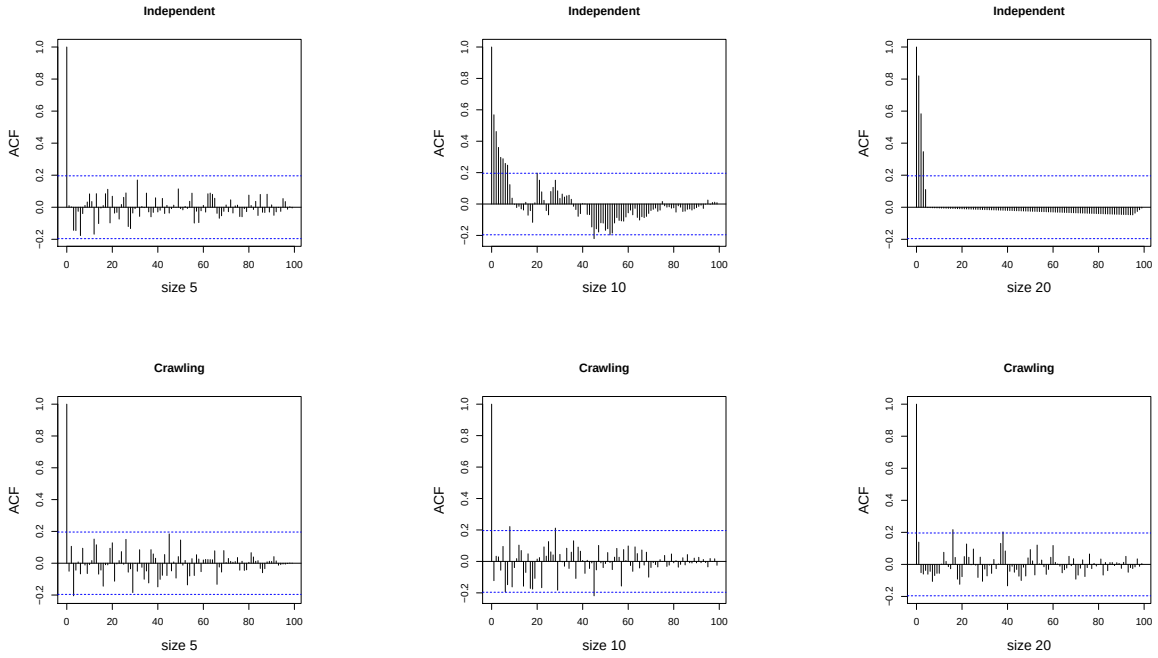

Figure 2.7: The ACF plot for the diameter of subtrees sampled using both methods

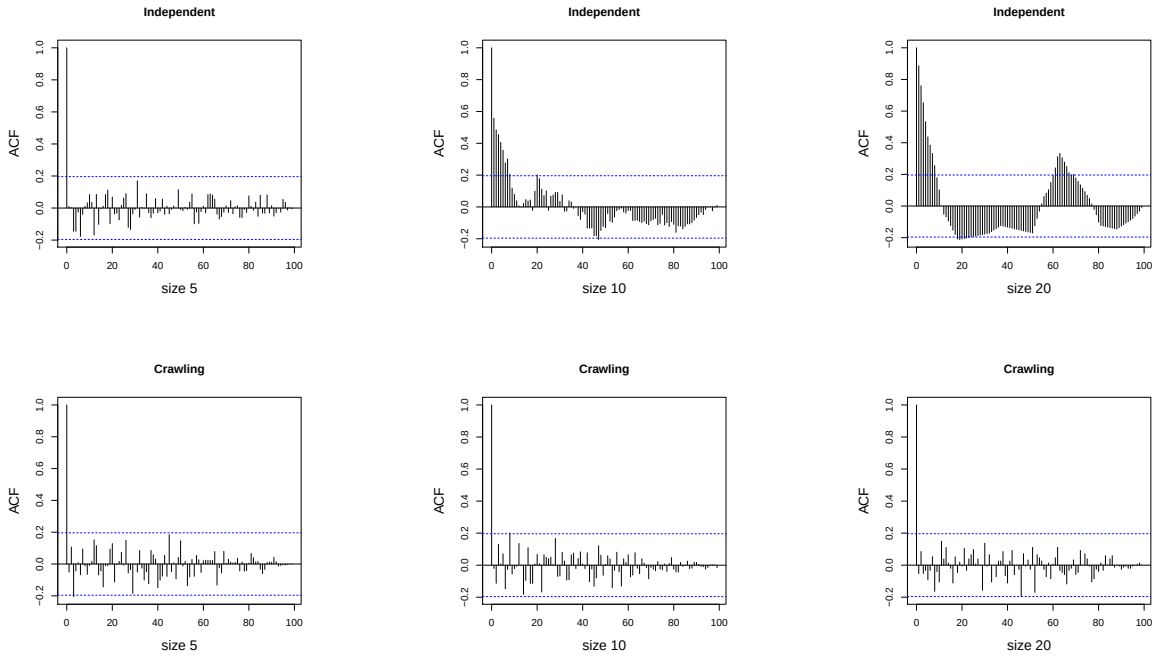

Figure 2.8: The ACF plot for the number of leaves of subtrees sampled using both methods

2.4 Conclusion

We presented two Markov chains where the convergence speed for the independent chain is $\mathcal{O}(s^{-1}nr^{k^2/2})$ whereas it is $\mathcal{O}(\lambda_1^{-1}a^2d_{max}^2k^5)$ for the non-independent method which could be very large when the bottleneck is narrow.

Considering the bound on the total variation distance in the crawling method different parameters appear, λ_1 which is the second eigenvalue that can be computed on the graph $G(V, E)$ and a the maximum number of subtrees associated to a vertex. Parameter a is more difficult (impossible often in practice) to compute. Indeed, for a $r-$regular graph it is obvious that for a tree of size k, $a \leq r^k$ which can be large. Hence, for a general random graph where a is known to be not too large then the crawling method could be better.

According to our simulations, the theoretical results were confirmed. The independent method showed better performance only when sampling subtrees from a graph with a narrow bottleneck whereas the crawling method was the best in the case of sampling from other types of graphs. As a result, we recommend to use a combination of both methods for sampling uniform subtrees, i.e to sample trees in two steps. Firstly one should sample subtrees globally from the graph using the independent method and after that to sample locally through the crawling method.

Bibliography

A. Shapiro and Z. Zhang. Comparing Multiple RNA Secondary Structures Using Tree Comparisons. *Bioinformatics,* 6(4), pp. 309–318, 1990.

A. Sinclair. Improved Bounds for Mixing Rates of Markov Chains on Combinatorial Structures. *Combinatorics, Probability and Computing,* 1(4), pp. 351–370, 1992.

B. Bollobás. *Graph Theory-An Introductory Course, Graduate Texts in Mathematics.* New York: Heidelberg and Berlin Springer-Verlag. 1979.

B. Bollobás. *Random Graphs.* Cambridge: Cambridge University Press. 2001.

C. Hübler, H. Kriegel, K. Borgwardt and Z. Ghahramani. Metropolis Algorithms for Representative Subgraph Sampling. *In: Proceedings of the 2008 eighth IEEE International Conference on Data Mining,* Dec 15–19, ICDM. IEEE Computer Society, Washington, DC, pp. 283–292, 2008.

C. Robert and G. Casella. *Introducing Monte Carlo Methods with R.* 2010 edition edition. Springer Verlag, New York. 2009. ISBN 978-1-4419-1575-7.

I. Takigawa, K. Hashimoto, M. Shiga, M. Kanehisa and M. Mamitsuka. Mining Patterns from Glycan Structures. In Proceedings of the International Beilstein Symposium on Glyco-Bioinformatics, pp. 13–24. Beilstein Institute. 2010.

J. Kemeny and J. Snell. *Finite Markov Chain: With a New Appendix "Generalization of a Fundamental Matrix".* 1st ed. 1960. 3rd Printing 1983 Edition. Springer, New York. 1983. ISBN 978-0-387-90192-3.

J. Liu. Metropolized Independent Sampling and Comparisons to Rejection Sampling and Importance Sampling. *Statistics and Computing,* 6(2), pp. 113–119, 1996. ISSN 1573-1375. doi: 10.1007/BF00162521. URL https://doi.org/10.1007/BF00162521.

O. Häggström. *Finite Markov Chains and Algorithmic Applications.* Cambridge: London Mathematical Society Student Texts, Cambridge University Press. 2002.

S. Brooks, A. Gelman, G. Jones, X. Meng and editors. *Handbook of Markov Chain Monte Carlo.* CRC Press, Taylor & Francis Group. 2011.

S. Parthasarathy, S. Tatikonda and D. Ucar. A survey of graph mining techniques for biological datasets. *In Managing and mining graph data, Springer,* pp. 547–580, 2010.

T. Hancock, N. Wicker, I. Takigawa and H. Mamitsuka. Identifying Neighborhoods of Coordinated Gene Expression and Metabolite Profiles. *Schönbach C, editor. PLoS ONE 7,* 2012. doi: 10.1371/journal.pone.0031345 PMID: 22355360.

X. Yan and J. Han. gSpan: Graph-Based Substructure Pattern Mining. *In 2002 IEEE International Conference on Data Mining, 2002. Proceedings.,* pp. 721–724, 2002. doi:10.1109/ICDM.2002.1184038.

Chapter 3

kPPP Graph Sampling

Graph sampling is very important to reduce graph size and preserve its properties in the same time. This chapter introduces a method to sample sets of close vertices from an arbitrary undirected graph using Markov chains.
We design a Markov chain to sample according to the Permanental Point Process (PPP) as the stationary distribution. The power of this distribution is that the probability of choosing a particular set of items is proportional to the permanent of a matrix that defines the similarity of those items.

3.1 Introduction

Graphs are non-linear data structures. Each graph $G(V, E)$ consists of a set of vertices V connected by edges E where edges may be directed or undirected. For a comprehensive introduction to graphs, see (Bollobas, 1979) and (Bollobas, 2001).

As seen in literature and through the thesis, graph theoretical ideas are extensively used in many domains and various applications like in the areas of computer science, biology, chemistry, physics, sociology, ... etc. Consequently, a huge amount of graph-structure date has appeared and so the necessity to develop methods and tools in order to analyze the properties of these graphs and study them profoundly. Hence, the term Graph mining came

into sight.

Graph mining gained a lot of attention during the last years due to its important applications. Different approaches have been proposed generally for classification, clustering and sampling. Coming down to the later, sampling methods are divided into three categories (Ahmed *et al.*, 2011): node sampling, edge sampling and topology-based sampling. For a good survey in graph sampling methods and the differences between them, see (Hu and Lau, 2013).

This work is mainly interested in sampling sets of similar vertices within a graph. Metropolis-Hastings algorithm (Metropolis *et al.*, 1953) is widely used in Markov chain Monte Carlo MCMC methods to sample a desired vertex distribution from a graph. Thus, MCMC sampling methods (Robert and Casella, 2010) could be used to sample the desired vertices in the existence of a suitable stationary distribution.

As we are looking for sets of close vertices that could be considered at clusters, a convenient stationary distribution would be the Permanental Point Process (PPP). This proposed distribution would be attracting due to the fact that the probability of choosing a particular set of items is proportional to the permanent of a matrix that defines the similarity of those items. We contribute in this chapter to sample k vertices among the n vertices of the graph. For this purpose, we design a Markov chain whose stationary distribution is the permanent of a suitable graph kernel. Additionally, we aim to overcome the computation problem for the graph kernel. In fact, it is believed that the permanent cannot be computed in polynomial time and that computing permanents is generally $\#P-$hard and $\#P-$complete for matrices with 0 or 1 entries (Valiant, 1979).

An outline of the chapter is as follows. Section 3.2 is a general presentation of permanents. A brief introduction and a summary of existing results about the graph Laplacian and the Moore-Penrose pseudo-inverse are presented in section 3.3. Following, section 3.4 is devoted

to present our designed MCMC algorithm to sample the desired vertices. The efficiency of the developed method is studied in section 3.4.2.

3.2 Permanental processes

Permanental process is a generalization of the squared centered Gaussian processes where their Laplace transform is given by the power $-\frac{1}{\alpha}$ of a determinant ($\alpha > 0$) involving a kernel (Eisenbaum and Kaspi, 2009). When $\alpha = -1$ it is the determinantal random process known as fermion, whereas, when $\alpha = 1$ it is a boson point process. In general, when $\alpha > 0$ the processes densities and their correlation functions are equal to permanents involving a kernel so they called the permanental random processes.

Definition 3.1 ((Eisenbaum and Kaspi, 2009))**.** *A real-valued positive process* $(\psi_x, x \in E)$ *is a permanental process if its finite-dimensional Laplace transforms satisfy for every* $(\alpha_1, \alpha_2, ..., \alpha_n)$ *in* $\mathbb{R}_+^{\kappa}$ *and every* $(x_1, x_2, ..., x_n)$ *in* E^n*,*

$$\mathbb{E}[\exp(-\frac{1}{2}\sum_{i=1}^{n}\alpha_i\psi_{xi})] = |I + \alpha G|^{-\frac{1}{\beta}}$$

where I *is the* $n \times n-$*identity matrix,* α *is the diagonal matrix* $diag(\alpha_i)_{1\leq i\leq n}$ *and* $G = (g(x_i, x_j))_{1\leq i,j\leq n}$ *and* β *is a fixed positive number.*

Such a process $(\psi_x, x \in E)$ *is called permanental process with kernel* $(G(x, y), x, y \in E)$ *and index* β*.*

Remark 3.1. *The permanent of an* $n \times n$ *matrix* $A = (a_{i,j})$ *is defined as*

$$\sum_{\sigma}\prod_{i} a_{i,\sigma(i)}$$

The sum is taken over all elements σ *so over all permutations of the numbers* $1, 2, ..., n$*.*

Simply, the permanent could be described as the determinant without signs. It has many applications in thermodynamics, graph theory, networks and computer science (Oh *et al.*, 2009), (Rezatofighi *et al.*, 2015). Recently, various problems in the physical sciences, combinatorics and linear algebra have been driven to the computation of permanents. Interestingly, many interpretations linked permanents to graphs. For a directed graph, the permanent is

the sum of weights of the cycle covers assuming that the square matrix A represents the adjacency matrix. Furthermore, for a bipartite graph, the permanent is the sum of weights of the perfect matchings.

Mathematically, the permanent calculation started around 1812 (Minc, 1982) but it didn't really attract the attention until 1979 when Valiant (1979) showed that computing permanent has NP complexity. After that, it has been extensively studied by complexity theorists. A review of the work concerning the complexity is found in (Jerrum and Sinclair, 1989). Indeed, it is found that developing an efficient general algorithm to compute the permanent is an objective difficult to catch. Among the many efforts to compute the permanent exactly, the algorithm presented by Ryser (1963) is considered the best although it is calculated using $\mathcal{O}(2^{n-1}n)$ arithmetic operations. A comprehensive review for various methods is found in (Jerrum *et al.*, 2009).

Additionally, similar interest devoted to evaluate the permanent in more applied domains as in the subject of boson sampling. In general, it is the subject of sampling from the probability distribution of detecting identical single photons at the output of the circuit. Indeed, the probability of of detecting a number of photons is proportional to the squared modulus of permanents of complex matrices (Scheel and Buhmann , 2008).

3.3 Graph Laplacian

As we are interested in sampling communities of similar vertices, Laplacian matrix would be a suitable choice to represent a graph due to its power in representing similarities between vertices in graph clustering. Although there is no unique convention in the literature about which matrix exactly is called graph Laplacian (Luxburg, 2006), it is generally defined as a matrix representation of the graph which is used to investigate various useful properties of the structure.

Among the many graph Laplacians presented and used to describe efficiently the graph

properties, the Laplacian matrix proved its accuracy to interpret the relationship between the graph vertices like the similarity between the vertices induced by the graph local structure. For an undirected graph $G = (V, E)$ consists of $|V| = n$ vertices and $|E| = m$ edges, the unnormalized graph Laplacian matrix is defined by

$$L = D - W$$

where W is the graph adjacency matrix given by

$$W_{ij} = \begin{cases} 1, & \text{when } i \sim j \\ 0, & \text{o.w} \end{cases}$$

and D is the diagonal matrix defined as $D = \text{Diag}(d_1, \cdots, d_n)$ such that $d_i = \sum_j W_{ij}$.

The Laplacian matrix satisfies many properties, like it it is symmetric and positive semi-definite. Additionally, the matrix L has n non-negative and real-valued eigenvalues $\lambda_1(L) \leq \lambda_2(L) \leq \cdots \leq \lambda_n(L)$, where the smallest eigenvalue $\lambda_1 = 0$ and its corresponding eigenvector is the constant one vector $\mathbb{1}$. For more interesting properties and a comprehensive overview, we recommend (Mohar, 1991).

The notion of normalized Laplacian matrix was first defined by Chung (1997). This matrix proved its efficiency in stochastic processes and spectral geometry due to the consistency between their eigenvalues. In addition to its power in generalizing many results, were basically found for regular graphs, to all graph types. The normalized Laplacian is given by:

$$\mathcal{L} := D^{-1/2}\ L\ D^{-1/2} = I - D^{-1/2}\ W\ D^{-1/2}$$

which is formulated for the graph G as

$$\mathcal{L}_{ij} = \begin{cases} -\frac{1}{\sqrt{d_i d_j}} & \text{if } i \neq j \text{ and } i \sim j \\ 1 & \text{if } i = j \text{ and } d_j \neq 0 \\ 0 & \text{o.w} \end{cases}$$

Several interesting results were presented about the eigenvalues of this matrix. It is found

that the spectrum of the normalized Laplacian satisfies

$$0 = \lambda_1(\mathcal{L}) \leq \lambda_2(\mathcal{L}) \leq \cdots \leq \lambda_n(\mathcal{L}) \leq 2.$$

Furthermore, it was proved that the second smallest normalized Laplacian eigenvalue $\lambda_2(\mathcal{L})$ equals the inverse of the largest eigenvalue of the pseudoinverse of the normalized Laplacian matrix presented below in definition 3.2.

Definition 3.2. *For a connected graph G. Let $\mathcal{L}$ be the normalized Laplacian, and its eigenvalues are defined as*

$$0 \leq \lambda_1(\mathcal{L}) \leq \cdots \leq \lambda_n(\mathcal{L}) \leq 2.$$

The Moore-Penrose pseudo-inverse is denoted by $\mathcal{L}^\dagger$ where its eigenvalues are the inverse of the eigenvalues of $\mathcal{L}$ and it expressed as

$$0 = \lambda_1(\mathcal{L}^\dagger) < \lambda_2(\mathcal{L}^\dagger) \leq \cdots \leq \lambda_n(\mathcal{L}^\dagger).$$

Definition 3.3. *For a connected graph G. Let the Moore-Penrose pseudo-inverse for the normalized laplacian matrix $\mathcal{L}^\dagger = WDW^{'}$, then, for all $i \in \{1, \ldots, n\}$,*

$$W_i D W_i^{'} = \sum_{j=1}^{n} W_{ij}^2 \lambda_j(\mathcal{L}^\dagger) = \sum_{j=2}^{n} W_{ij}^2 \lambda_j(\mathcal{L}^\dagger)$$

The Moore-Penrose pseudo-inverse of the normalized Laplacian $\mathcal{L}^\dagger$ over the graph G summarizes the similarities between pairs of vertices in a graph. Generally, it is found to be a recommended graph kernel to represent the Gram matrix. As a consequence, the Moore-Penrose pseudo-inverse of the normalized Laplacian matrix would be a suitable graph kernel in our work to sample k similar vertice according to the permanent stationary distribution.

3.4 The kPPP sampling algorithm

Sampling from particular distributions allows to collect a set of points that are generated by a theoretical distribution. Consequently, these points conform to the distribution parameters and properties. MCMC sampling methods are algorithms for sampling from a probabil-

ity distribution. Convincingly, they proved their efficiency in sampling different structures within graphs. Generally, each Markov chain has a desired distribution where the chain converges toward it in equilibrium.

In this work, we combine MCMC sampling methods and the permanental point processes. More precisely, we choose the permanental point process as the chains stationary distribution, due to the fact that choosing the set of k vertices is proportional to the permanent of the matrix represents the Moore-Penrose pseudo-inverse of the normalized Laplacian kernel so that the Markov chain stationary distribution $P(x) \propto \prod_{i=1}^{k} k(x_i, x_{\sigma_i})$.

The Markov chain is generated by a Metropolis-Hastings algorithm. Consider x is the current state of the chain, a new candidate y is proposed according to a proposal distribution $q(x, y)$.

The Markov chain (X_{t+1}) is produced through the transition kernel:

$$X_{t+1} = \begin{cases} y \text{ with probability } = \alpha \\ x \text{ with probability } = 1 - \alpha \end{cases}$$

where α is the acceptance probability. Precisely, the proposed state is accepted as the new state of the chain with probability:

$$\alpha = \frac{\pi(y)}{\pi(x)} \frac{q(y, x)}{q(x, y)} \wedge 1$$

Furthermore, the transition probability matrix $P(x, y)$ is given as

$$P(x, y) = \begin{cases} \left\{ \frac{\pi(y)}{\pi(x)} \frac{q(y,x)}{q(x,y)} \wedge 1 \right\} q(x, y) & \text{if } x \neq y \\ 1 - \left\{ \frac{\pi(y)}{\pi(x)} \frac{q(y,x)}{q(x,y)} \wedge 1 \right\} \sum_{i \neq x} q(x, i) & \text{otherwise} \end{cases}$$

According to the stationary distribution, the chain needs to calculate the permanent at each step in order to move for a new one or to stay in the current state. Thus, the general algorithm needs to compute the sum of all permutations σ of the symmetric group S_n for the n vertices which would be difficult of a high complexity and even impossible for a huge number of vertices. Our Markov chain is designed to overcome this problem by sampling a joint distribution whose marginal distribution is a kPPP.

3.4.1 The algorithm moves

For the graph $G(V, E)$ where $|V| = n$ and $|E| = m$, a state is described by an active set of k vertices belonging to $\{v_1, \ldots, v_n\}$ and a permutation over these k vertices. Equivalently, we can consider that we have at hand a permutation over the whole set of vertices $\{v_1, \ldots, v_n\}$ with the $n - k$ vertices of the set being fixed.

A way of coding this is thus to write only the cycles of the permutation which involve these k vertices. Consequently, Each state will contain an active set of k vertices ordered in cycles ( ) and a non-active set [ ] of $n - k$ fixed vertices, where each c_i is a cycle and $|c_i|$ is the size of the cycle so that $\sum_{i=1}^{l} |c_i| = k$.

At each step, the Markov chain will consider a move among the two presented moves below.

Moves of type 1 (replace move) The Markov chain replaces a vertex in the active set with a vertex in the non-active set.

$$x = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \\ v_n \end{pmatrix} = \begin{pmatrix} c_{x_1} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \\ c_{x_2} = \begin{pmatrix} v_3 \end{pmatrix} \\ c_{x_3} = \begin{pmatrix} v_4 \\ \vdots \\ v_k \end{pmatrix} \\ \begin{bmatrix} v_{k+1} \\ \vdots \\ v_n \end{bmatrix} \end{pmatrix} \rightarrow y = \begin{pmatrix} c_{y_1} = \begin{pmatrix} v_1 \\ v_i \end{pmatrix} \\ c_{y_2} = \begin{pmatrix} v_3 \end{pmatrix} \\ c_{y_3} = \begin{pmatrix} v_4 \\ \vdots \\ v_k \end{pmatrix} \\ \begin{bmatrix} v_2 \\ v_{k+1} \\ \vdots \\ v_n \end{bmatrix} \end{pmatrix}$$

Moves of type 2 (split-join move) The Markov chain composes the active set in the state x with a random transposition taken among it which produces one more or one fewer number of cycles. Considering this move leads to split a cycle into two cycles or to join two

of them in a cycle as below.

$$x = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \\ v_n \end{pmatrix} = \begin{pmatrix} c_{x_1} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \\ c_{x_2} = \begin{pmatrix} v_3 \end{pmatrix} \\ c_{x_3} = \begin{pmatrix} v_4 \\ \vdots \\ v_k \end{pmatrix} \\ \begin{bmatrix} v_{k+1} \\ \vdots \\ v_n \end{bmatrix} \end{pmatrix} \nearrow \quad y = \begin{pmatrix} c_{y_1} = \begin{pmatrix} v_1 \end{pmatrix} \\ c_{y_2} = \begin{pmatrix} v_2 \end{pmatrix} \\ c_{y_3} = \begin{pmatrix} v_3 \end{pmatrix} \\ c_{y_4} = \begin{pmatrix} v_4 \\ \vdots \\ v_k \end{pmatrix} \\ \begin{bmatrix} v_{k+1} \\ \vdots \\ v_n \end{bmatrix} \end{pmatrix}$$

$$\searrow \quad y = \begin{pmatrix} c_{y_1} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \\ c_{y_2} = \begin{pmatrix} v_4 \\ v_5 \\ \vdots \\ v_k \end{pmatrix} \\ \begin{bmatrix} v_{k+1} \\ \vdots \\ v_n \end{bmatrix} \end{pmatrix}$$

Additionally, we make the chain lazy so with probability 1/2 the state stays the same or, with 1/2 , we make a Metropolis-Hastings move with any type of moves with probability 1/2. Moreover, when the Markov chain is lazy then it's eigenvalues become nonnegative which can improve the bounds of the mixing time.

Algorithm 5 Sampling from the joint distribution $P(x) = \prod_{i=1}^{k} k(x_i, x_{\sigma_i})$

Require: A set of vertices V where $|V| = n$

Randomly select the k vertices to be permuted

Randomly initialize the set of permutation cycles $x_t = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_l \\ v_{k+1} \\ \vdots \\ v_n \end{pmatrix}$

while not mixed **do**

Select a move uniformly among the 2 types

Following to the move type, set $y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \\ v_{k+1} \\ \vdots \\ v_n \end{pmatrix}$

Set

$$\alpha \leftarrow min \left\{ 1, \frac{1}{2} \frac{p(y)q(y, x_t)}{p(x_t)q(x_t, y)} \right\}$$

Pick randomly $u \in U(0, 1)$

if $u \leq \alpha$ **then**

$x_{t+1} \leftarrow y$

else

$x_{t+1} \leftarrow x_t$

end if

end while

3.4.2 The algorithm convergence speed

We design a Metropolis-Hastings algorithm to simulate the Markov chain. Accordingly, it suffices for the designed chain over a finite configuration space to be irreducible and aperiodic in order to converge toward the target distribution. In fact, the algorithm samples will be asymptotically distributed according to the stationary distribution of the simulated Markov chain (Häggström, 2002).

The chain is clearly irreducible as each set of k vertices has a positive probability of being

sampled at each step. Furthermore, our lazy chain is aperiodic since $P(x,x) \geq 0$ for any state x which means that it is possible to stay at the same state with a positive probability. Hence, the convergence is assured. The following theorem bounds the algorithm convergence speed.

Theorem 3.1. *For the graph $G(V,E)$ with $|V| = n$ vertices and $|E| = m$ edges where we sample k vertices. Let $K_{\min}$ and $K_{\max}$ are the minimum and the maximum values for the Laplacian pseudo-inverse kernel, then the second eigenvalue on the graph λ_1 is bounded as*

$$\| M_x^m - \pi \|^2 \leq \frac{1}{4\pi(x)} \left(1 - \frac{1}{K}\right)^{2m}$$

s.t $K \leq \frac{4}{3} n(n+1)k^5(k^2+9k-10)\left(\frac{K_{max}}{K_{min}}\right)^{3k}$
with a probability larger than $1-\alpha$, where M_x^m is the m^{th} updated distribution and π is the target distribution.

Proof. We use the total variation distance to bound the convergence speed and the mixing time of our algorithm. It is defined as

$$\| M_x^m - \pi \|^2 \leq \frac{1}{4\pi(x)} \left(1 - \frac{1}{K}\right)^{2m} \tag{3.1}$$

where M_x^m is the m^{th} updated distribution and π is the target distribution.

According to Sinclair (1992), using canonical paths, the Markov chain has the following bound on its second eigenvalue:

$$\lambda_1 \leq 1 - \frac{1}{K}$$

where K is a measure of the maximum flow along any edge in the graph and is given by

$$K = \max_e \; Q(e)^{-1} \sum_{\gamma_{xy} \ni e} |\gamma_{xy}| \pi(x) \pi(y) \tag{3.2}$$

Assuming that edge $e = (e_1, e_2)$, then $Q(e) = \pi(e_1)P(e_1,e_2)$ where $P(e_1,e_2)$ is the transition matrix and $|\gamma_{xy}|$ stands for the length of the path γ_{xy}.

The canonical paths described in section 1.3.2.3 is the set of paths where one and only one path between every unequal (x, y) vertices, such that no edge is very heavily congested. The canonical path connects both permutations x and y is constructed by applying the defined moves in 3.4.1 on the active set cycles in x and hence, the permutation y is not always the same. In fact, the permutation y connected by x changes depending on the moves which construct the path. More precisely, the path is obtained by recreating the cycles of the permutation y active set one by one starting from permutation x using both types of moves. Simply, let that the active set of permutation y is defined by the cycle $(v_1, \ldots, v_k)$. Firstly, the method starts by verifying from the vertices of the active set cycles in x. Consequently, for a composition of transposition move, it verifies first if both vertices v_1, v_2 already exist in the active set of x. If not, add them vertex by vertex. Following, apply the composition move with the transposition (v_1, v_2) to produce the cycle (v_1, v_2) in the active set of the new permutation x_1. Similarly, the method continues, from permutation x_1 to x_2 to $\ldots$ to y, by applying the moves until creating the cycle $(v_1, \ldots, v_k)$ as in the active set of y.
On the other hand, when applying the replace move, the method verifies first if the vertex v_1 exists in the active set where it adds it to the cycle of the active set while not existing by replacing it with another vertex of that set. Whereas, it steps to replace v_2 if v_1 already exists. The method continues in this way until obtaining all cycles of the active set of y.

Accordingly, the number of canonical paths passing by the edge e could be bounded as follows. Fix an edge $e = (e_1, e_2)$ where the state e_2 is reachable from the state e_1 by a composition of transposition move, or by a replace move such that a vertex in the active set of e_1 is replaced by a vertex from y as in the following schema.

$$x = \left(\left(\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} \begin{pmatrix} v_5 \\ v_6 \\ v_7 \\ v_k = v_8 \end{pmatrix} \begin{bmatrix} v_{k+1} \\ \vdots \\ v_n \end{bmatrix} \right) \right) \rightarrow \ldots \nearrow e_1 = \left(\left(\begin{pmatrix} v_{k+2} \\ v_{k+1} \end{pmatrix} \begin{pmatrix} v_4 \\ v_5 \\ v_6 \\ v_2 \\ v_1 \\ v_8 \end{pmatrix} \begin{bmatrix} v_3 \\ \vdots \\ v_n \end{bmatrix} \right) \right) \rightarrow e_2 = \left(\left(\begin{pmatrix} v_{k+2} \\ v_{k+1} \end{pmatrix} \begin{pmatrix} v_1 \\ v_3 \\ v_5 \\ v_2 \\ v_4 \\ v_8 \end{pmatrix} \begin{bmatrix} v_6 \\ \vdots \\ v_n \end{bmatrix} \right) \right) \rightarrow \ldots \rightarrow y = \left(\left(\begin{pmatrix} v_{k+2} \\ v_{k+1} \end{pmatrix} \begin{pmatrix} v_1 \\ v_3 \\ v_5 \\ v_2 \\ v_7 \\ v_8 \end{pmatrix} \begin{bmatrix} v_4 \\ v_6 \\ \vdots \\ v_n \end{bmatrix} \right) \right)$$

$$\searrow e_1 = \left(\left(\begin{pmatrix} v_{k+2} \\ v_{k+1} \end{pmatrix} \begin{pmatrix} v_4 \\ v_5 \\ v_6 \\ v_2 \\ v_{11} \\ v_8 \end{pmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \right) \right) \rightarrow e_2 = \left(\left(\begin{pmatrix} v_{k+2} \\ v_{k+1} \end{pmatrix} \begin{pmatrix} v_1 \\ v_5 \\ v_6 \\ v_2 \\ v_{11} \\ v_8 \end{pmatrix} \begin{bmatrix} v_3 \\ \vdots \\ v_n \end{bmatrix} \right) \right)$$

Thus, to bound the number of paths in the constructed set of canonical paths, assume that i vertices in e_1 among the k vertices are similar to the vertices in y and well-placed in a cycle/cycles similar to the cycle/cycles in the active set of y. Hence, two scenarios would be considered as below:

- e_1 and e_2 are connected using a composition of transposition move, so we add at most two vertices to the active set, if they don't exist, in order to do the move. As a result, we have $(i+1)$ or $(i+2)$ vertices whose image is the same as in y and we still have at most to place $k-(i+1)$ among the $n-(i+1)$ vertices. These vertices could be added to the constructed cycle or would be used to create a new permutation among themselves. Considering the later option produces $(k-(i+1)+1)! = (k-i)!$ possibilities. Additionally, at most $(i+2)$ among the $n-(k-(i+2))$ vertices must be placed in

x as in e_1. Similarly, considering that those vertices could create a new permutation among themselves leaves $(i+2+1)! = (i+3)!$ possibilities.

As a results, the number of canonical paths which cross the edge e is bounded as

$$\#\{x,y : e \in \gamma_{xy}\}_1 \leq \sum_{i=1}^{k-1} \begin{pmatrix} n-(i+1) \\ k-(i+1) \end{pmatrix} (k-i)! \begin{pmatrix} n-(k-(i+2)) \\ (i+2) \end{pmatrix} (i+3)! \quad (3.3)$$

- e_1 and e_2 are connected using a replace move, where only one vertex is needed to be chosen in order to achieve the move. Consequently, by counting that vertex which is used for the move from e_1 to e_2, we still have at most $k-(i+1)$ among $n-(i+1)$ vertices which at most must be added to the active set to get a possible y. Taking into consideration that $(k-i)$ vertices could create a new permutation among themselves or by attaching to a current cycle, $(k-i+1)!$ possibilities are counted. Moreover, to obtain x from e_1, i vertices among the $n-(k-i)$ vertices of e_1 must be placed as in x. Additionally, $(i+1)!$ possibilities are counted by taking into account that a new permutation could be obtained from these vertices.

  Thus, the number of canonical paths which cross the edge e is bounded as

$$\#\{x,y : e \in \gamma_{xy}\}_2 \leq \sum_{i=1}^{k-1} \begin{pmatrix} n-(i+1) \\ k-(i+1) \end{pmatrix} (k-i+1)! \begin{pmatrix} n-(k-i) \\ i \end{pmatrix} (i+1)! \quad (3.4)$$

In general, the number of canonical paths while considering both possibilities above could be bounded as

$$\#\{x,y : e \in \gamma_{xy}\} \leq \max\{ \ \ \#\{x,y : e \in \gamma_{xy}\}_1, \#\{x,y : e \in \gamma_{xy}\}_2 \ \ \}$$

which is upper bounded by equation 3.3. Furthermore, the length of the path γ_{xy} is $2k$ at

maximum, since there are two moves could be applied for each vertex among the k vertices grouped in cycles.

Using equation 3.3, K in equation 3.2 is bounded as

$$\begin{aligned} K =& \max_{e} Q(e)^{-1} \sum_{\gamma_{xy} \ni e} |\gamma_{xy}| \pi(x)\pi(y) \\ \leq& \frac{1}{\pi(e_1)P(e_1,e_2)} 2k \sum_{i=1}^{k-1} \frac{(n-i-1)!(k-i)}{(n-k)!} \frac{(n-(k-(i+2)))!(i+3)}{(n-k)!} \pi(x)\pi(y) \end{aligned} \tag{3.5}$$

Considering that the minimum and the maximum values for the Laplacian pseudo-inverse kernel are denoted by $K_{\min}$ and $K_{\max}$, then the transition matrix $P(x,y)$ is bounded as

$$\begin{aligned} P(x,y) =& \frac{1}{2} q(x,y) \Big(1 \wedge \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\Big) \\ \geq& \frac{q(x,y)}{2} \wedge \frac{1}{4k^2 n} \Big(\frac{K_{\min}}{K_{\max}}\Big)^k \\ \geq& \frac{1}{4k^2 n} \Big(\frac{K_{\min}}{K_{\max}}\Big)^k \end{aligned} \tag{3.6}$$

Moreover, the stationary distribution $\pi(x) \propto \text{Per } K_x = \frac{1}{c}\text{Per } K_x$, where c is the normalization constant. Knowing that $K_{min}^k \leq \pi(x) \leq K_{max}^k$, then c could be bounded as

$$\begin{aligned} c &= \frac{1}{\sum_1^k \prod_1^k K_x} \\ &\leq \frac{1}{\binom{n}{k}(k-1)!\ K_{min}^k} \end{aligned} \tag{3.7}$$

Consequently, using equations 3.6 and 3.7, equation 3.5 could be bounded as

$$\begin{aligned} K \leq& 4k^2 n \frac{K_{max}^k}{K_{min}^k} \frac{K_{max}^{2k}}{K_{min}^k} 2k \sum_{i=1}^{k-1} \frac{(n-i-1)!(k-i)}{(n-k)!} \frac{(n-(k-(i+2)))!(i+3)}{(n-k)!} \frac{1}{\binom{n}{k}(k-1)!K_{min}^k} \\ \leq& 8nk^3 \Big(\frac{K_{max}}{K_{min}}\Big)^{3k} \frac{k!(n-k)!}{n!(k-1)!} \sum_{i=1}^{k-1} (k-i)(i+3) \frac{(n-i-1)!}{(n-k)!} \frac{(n-k+i+2)!}{(n-k)!} \end{aligned}$$

$$
\begin{aligned}
&\leq 8nk^4\Big(\frac{K_{max}}{K_{min}}\Big)^{3k}\sum_{i=1}^{k-1}(k-i)(i+3)\frac{(n-k+i+2)!}{n(n-1)\dots(n-i)(n-k)!}\\
&\leq 8nk^4\Big(\frac{K_{max}}{K_{min}}\Big)^{3k}\sum_{i=1}^{k-1}(k-i)(i+3)\frac{(n-k+i+2)(n-k+i+1)\dots(n-k+1)}{n(n-1)\dots(n-i)}\\
&\leq 8n(n+1)k^4\Big(\frac{K_{max}}{K_{min}}\Big)^{3k}\sum_{i=1}^{k-1}(k-i)(i+3)\\
&\leq \frac{4}{3}n(n+1)k^5(k^2+9k-10)\Big(\frac{K_{max}}{K_{min}}\Big)^{3k} \qquad (3.8)
\end{aligned}
$$

where K_{min} and K_{max} represent the upper and lower bounds for the Moore-Penrose pseudo-inverse kernel, presented in section 3.3, which could be computed for the laplacian matrix.

□

3.5 Conclusion

The proposed joint stationary distribution is proved to be a powerful tool for close vertices sampling purpose as it combines the permanent properties in selecting close items, in addition to solve the issue of computing the permanent for large matrices. Our kPPP sampling algorithm works well mathematically while k is small as the bound presented in Theorem 3.1 is hugely affected its value. Generally, a tighter bound for the convergence speed would be recommended.

An indispensable simulation studies would be involved in further work to evaluate experimentally the speed of the method in various circumstances.

Bibliography

A. Sinclair. Improved bounds for mixing rates of Markov chains on combinatorial structures. *Combinatorics, Probability and Computing 1,* pp. 351–370, 1992.

B. Bollobás. *Graph Theory-An Introductory Course, Graduate Texts in Mathematics*. New York: Heidelberg and Berlin Springer-Verlag, 1979.

B. Bollobás. *Random Graphs.* Cambridge: Cambridge University Press. 2001.

B. Mohar. The Laplacian spectrum of graphs. *In Graph theory, combinatorics, and applications.* Vol. 2 (Kalamazoo, MI, 1988), pp. 871–898, New York: Wiley. 1991.

C. Robert and G. Casella. *Introducing Monte Carlo Methods with R*. Springer Science+Business Media, 2010.

F. Chung. *Spectral Graph Theory.* American Math. Soc. Providence. 1997.

H. Minc. *Permanents.* Encyclopedia of Mathematics and Its Applications Vol. 6, Addison-Wesley, Reading, Mass. 1982.

H. Ryser. *Combinatorial Mathematics.* Carus Mathematical Monographs, No. 14., Math. Assoc. Amer., Washington, DC, 1963.

L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science 8,* pp. 189–201, 1979.

M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal on Computing 18,* pp. 1149–1178, 1989.

M. Jerrum, A. Sinclair and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. *In Proceedings of the 33rd ACM Symposium on Theory of Computing,* pp. 712–721. ACM, 2001.

N. Ahmed, J. Neville and R. Kompella. Network Sampling via Edge-Based Node Selection with Graph Induction. *In Purdue University, CSD TR # 11–016* , pp. 1–10, 2011.

N. Eisenbaum and H. Kaspi. On permanental processes. *Stochastic Process. Appl.119,* pp. 1401–1415, 2009.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics,* 21:1087, 1953.

O. Häggström. *Finite Markov Chains and Algorithmic Applications.* London Mathematical Society Student Texts. Cambridge University Press, Cambridge. 2002.

P. Hu and W. Lau. A Survey and Taxonomy of Graph Sampling. *ArXiv13085865 Cs Math Stat* , 2013. Available from: http://arxiv.org/abs/1308.5865

S. Oh, S. Russell and S. Sastry. Markov chain monte carlo data association for multi-target tracking. *IEEE Transactions on Automatic Control,* 54(3): 481–497, 2009.

S. Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick and I. Reid. Joint probabilistic data association revisited. *In Proceedings of the IEEE international conference on computer vision,* pp. 3047–3055, 2015.

S. Scheel and S. Buhmann. MACROSCOPIC QUANTUM ELECTRODYNAMICS - CONCEPTS AND APPLICATIONS. *Acta Physica Slovaca 58,* 675, 2008.

U. Luxburg. A tutorial on spectral clustering, *Technical Report 149.* Max Planck Institute for Biological Cybernetics. 2006.

Chapter 4

KBER: A Kernel Bandwidth Estimate Using the Ricci Curvature

The choice of a bandwidth can affect dramatically the accuracy of classification methods relying on it. Recently, a large number of methods to choose the bandwidth have been developed. This chapter presents a simple method called KBER to estimate the bandwidth using the average Ricci curvature of $\epsilon-$graphs. The Radial Basis Function kernel (RBF) has been chosen in our work for its simplicity and its popularity in this kind of research; it is also possible to apply our method to any kernel with the same kind of parameter.

4.1 Introduction

Support vector machines (SVM) technique is one of the most theoretically sound and widely used techniques for supervised classification. This technique showed more accuracy or at least an equivalent performance comparing to other classifiers in many applications (John and Nello, 2004). One requirement for SVM to perform well is the choice of the kernel function and its parameter (Chang and Lin, 2001). Several kernel functions were developed, such as the Gaussian radial basis (RBF) kernel, which has proven to be an efficient choice of kernel for a variety of tasks like classification.

4.1.1 Gaussian kernel

The Gaussian radial basis (RBF) kernel is one of the most used kernel functions and is known to be a reasonable default choice (Chang and Lin, 2001). It is defined as

$$K_\sigma(x,y) = \exp\Big\{-\frac{d(x,y)^2}{\sigma^2}\Big\},\ x,y \in \mathbb{R}^d. \tag{4.1}$$

where $\sigma^2 \in (0,\infty)$ is the kernel parameter called the bandwidth or the width of the Gaussian kernel.

The efficiency of the RBF kernel depends on the choice of this parameter. The larger σ^2, the more influence y will exert on x. More precisely, if y is a support vector, a large σ^2 implies that the class of this support vector will have influence on deciding the class of the vector x even if the distance between them is large. On the other hand, if σ^2 is small, then the support vector does not have such wide-spread influence.

4.1.2 Methods for bandwidth selection

The way to select the most suitable RBF kernel function and its bandwidth for SVM and MSVM classification is a research problem in Machine learning. Many methods were developed to estimate this parameter, see (Haykin, 1994), (Nakayama *et al.*, 2002) and (Benoudjit and Verleysen, 2003). Through the years, some methods were extensively used starting by a cross-validation procedure (Muller *et al.*, 2001). In general, this method applies a grid-search on σ of SVMs with the $k-$fold cross-validation. This method works well in practice, but it has some shortfalls. The quality of the parameter could be affected by the range of the grid. Also, the computational complexity is high since this method evaluates each bandwidth parameter with respect to the $k-$fold classification task. This is time consuming especially for high dimensional or large data.

Later, Chapelle *et al.* (2002) developed a method and showed that the RBF parameter could be selected by an optimization approach. The work of Soares *et al.* (2004) suggested also to choose the bandwidth within a range for regression using meta-learning. Previously,

(Schölkopf, 2003) suggested to search the RBF bandwidth value between 0.2 and 1 for SVM classification (Felici and Vercellis, 2008). Surprisingly, this range is the commonly tested range in the literature although it is a tight range which could yield a less accurate classification. Later on, Ali and Smith (2005) have investigated empirically how to select the RBF parameter and showed that its value could be out of the range $0.2 - 1.2$. Furthermore, many heuristic approaches were proposed to estimate the same parameter depending, for example, on the dimension of the data set like the methods presented in (Lauer and Guermeur, 2011) and (Pedregosa *et al.*, 2011). We recommend (Ali and Smith, 2005), for a comprehensive comparison between the most common methods and their efficiency.

In this work, we attempt to present a new simple method based on the average Ricci curvature which doesn't suffer from the aforementioned limitations. Moreover, this method could yield better estimates for the bandwidth when increasing the size of the data set or the number of dimensions.

4.2 Ricci Curvature

Curvature is known as the amount by which a geometric object such as a surface deviates from being a flat plane or a curve from being straight as in the case of a line, but this is defined and generalized in different ways depending on the context. One of these generalizations is the Ricci curvature, which was defined on general metric spaces by many researchers like Sturm (2006) and Lott and Villani (2009). Moreover, it was presented for graphs by Chung and Yau (1996) and defined on Markov chains in (Ollivier, 2009).

Geometrically, the Ricci curvature is the mathematical object that controls the growth rate of the volume of metric balls in a manifold. This curvature represents the amount by which the volume of a geodesic ball in a curved Riemannian manifold deviates from that of the standard ball in Euclidean space. If Ricci curvature at a specific point is non-negative, the volume growth with respect to the radius of the ball centered at that point is polynomial while if the Ricci curvature is negative everywhere, the volume growth is exponential.

Many definitions for Ricci curvature were presented in the literature using the geodesics on the manifold which allow to generalize to discrete settings. The Ollivier's coarse Ricci curvature presented in (Ollivier, 2009) works particularly well on discrete spaces like graphs. It is formulated in terms of the transportation distance between local measures. For a pair of nodes $x, y \in X$, it is obtained by comparing the Wasserstein distance from v_1 to v_2 and the distance $d(x, y)$.

Definition 4.1 (Ricci curvature, (Ollivier, 2009))**.** *Let (X, d) be a metric space with a random walk m. Let $x, y \in X$ be two distinct points lying on distance $d(x, y)$, the coarse Ricci curvature of (X, d, m) along xy is*

$$\kappa(x, y) := 1 - \frac{W_1(m_x, m_y)}{d(x, y)} \tag{4.2}$$

where $W_1(m_x, m_y)$ is the Wasserstein distance, presented below in Definition 4.2. It represents the minimum average traveling distance that can be achieved by a transportation plan.

Definition 4.2 (Wasserstein distance, (Ollivier, 2009))**.** *Let (X, d) be a metric space and let v_1, v_2 be two measures on X, the L^1 transportation distance between v_1, v_2 is*

$$W_1(v_1, v_2) = \inf_{\xi \in \prod(v_1, v_2)} \int_{(\mathrm{x,y}) \in \mathrm{X} \times \mathrm{X}} d(x, y) \mathrm{d}\xi(x, y) \tag{4.3}$$

where $\prod(v_1, v_2)$ is the set of measures on $X \times X$ projecting to the measures v_1 and v_2. Seen from the optimal transport point of view $\mathrm{d}\xi(x, y)$ represents the infinitesimal mass that travels from x to y, $\mathrm{d}v_1(x)$ and $\mathrm{d}v_2(y)$ the masses that must quit x and arrive in y respectively and $d(x, y)$ the transportation cost from x to y. This transportation distance is usually associated with the names of Kantorovich, Wasserstein, Ornstein and Monge. To read more, a good reference is (Hazewinkel, 2011).

In order to compute the Ricci curvature, the Wasserstein distance can be calculated by linear programming on a graph as follows, where v_1 and v_2 stand for the one-step Markov chains starting respectively in x and y.

Remark 4.1. *Intuitively, the Wasserstein distance measures the optimal cost to move one pile of sand to another with the same mass. In the case of a graph G, the supports of v_1 and v_2 are finite discrete sets, and thus, η is just a matrix with terms $\eta(x', y')$ representing the mass moving from $x' \in$ support of v_1 to $y' \in$ support of v_2 where $x' \sim x$, $y' \sim y$ means that x' and x, y' and y are neighbors. That is, in this case*

$$W_1(v_1, v_2) = \inf_{\eta} \sum_{x', x' \sim x} \sum_{y', y' \sim y} \eta(x', y') d(x', y')$$

where the infimum is taken over all matrices η which satisfy

$$\sum_{x', x' \sim x} \eta(x', y') = \frac{w_{yy'}}{d_y}, \qquad \sum_{y', y' \sim y} \eta(x', y') = \frac{w_{xx'}}{d_x}.$$

In general, d_x represents the degree of $x \in G$ s.t $d_x = \sum_{y, y \sim x} w_{xy}$, and w_{xy} denotes the weight associated to $x, y \in G$.

The Ricci curvature has many connections to the local and global properties of graphs. In the literature, some references paid attention to the positive Ricci curvature on graphs. Jost and Liu (2014) showed that when two vertices share many triangles, then the transportation distance should be small and the curvature therefore correspondingly large. Additionally, they proved that we can force the curvature $\kappa(x, y)$ to be positive by increasing the number of triangles that include x, y as vertices. Cushing *et al.* (2017) mentioned that positive lower bounds on curvature ensure the existence of spectral gaps, which was also proved previously in (Ollivier, 2009). Additionally, these graphs have a bounded diameter, since those graphs with a positive lower bound of curvature cannot be large. As a result, we can't have families of expander graphs in the space of positively curved graphs. Moreover, they proved that an edge $e = (x, y)$ in a $3-$regular graph must be a triangle or a square to have non-negative Ricci curvature.

In the work of (Ni *et al.*, 2015), they studied the discrete Ricci curvature of Internet and

showed interesting results on networks and graphs in general. They showed that the Ricci curvature is connected to local measures like the clustering coefficient, in addition to global measures such as network connectivity and centrality. Moreover, it appears that edges with positive Ricci curvature are clustered together while edges with negative curvature are working as links between clusters which implies that a positive Ricci curvature could be used to estimate the suitable kernel bandwidth parameter σ that plays an important role in the classification process and massively affects its result.

4.3 KBER Method

The kernel bandwidth estimate using the Ricci curvature, briefly called as the KBER method, estimates the kernel bandwidth parameter σ^2 in a few simple steps. It starts by constructing a $kNN-$graph from the target data set where k represents the number of neighbors connected to each vertex and ϵ_k denotes the distance to the k^{th} neighbor. KBER method chooses σ as the smallest value close to ϵ_k, yielding a positive Ricci curvature for the constructed graph, and greater than the inverse of the neighbors k. In other words, we estimate the value of σ s.t $|\epsilon_k - \sigma|$ is minimum and satisfies $\kappa(x, y) \geq \frac{1}{k}$. The rationale behind the later comparison is driven by the convergence speed result of Ollivier (2009) which states that the convergence speed is of the same order of magnitude as the inverse of the curvature meaning here that we expect a Markov chain restricted to a $k-$neighborhood to converge in only k steps.

Knowing that the Ricci curvature of the graph is local, it is necessary to compute it for each edge $e = (e_1, e_2)$ in the graph as in equation 4.2. The Wasserstein distance could be represented for discrete settings using Remark 4.1 as

$$W_1(m_x, m_y) = \sum_{i=1}^{|V|=n} \sum_{j=1}^{|V|=n} p(x \to x_i, y \to y_j)\ d(x_i, y_j) \tag{4.4}$$

Furthermore, equation 4.4 can be computed for each edge by solving the following linear

programming system

$$\min \sum_{i=1}^{|V|=n} \sum_{j=1}^{|V|=n} p(x \to x_i, y \to y_j) d(x \to x_i, y \to y_j) \text{ where}$$
$$\forall j, \sum_{i=1}^{|V|=n} p(x \to x_i, y \to y_j) = p(y \to y_j) \propto k(y, y_j) = \exp\Big\{-\frac{||y - y_j||^2}{\sigma^2}\Big\}$$
$$\forall i, \sum_{j=1}^{|V|=n} p(x \to x_i, y \to y_j) = p(x \to x_i) \propto k(x, x_i) = \exp\Big\{-\frac{||x - x_i||^2}{\sigma^2}\Big\}$$

where $p(x \to x_i, y \to y_j)$ represents, for the edge $e = (x, y)$, the probability that the 1^{st} vertex x will move to a neighbor vertex x_i while the 2^{nd} vertex y will move to its neighbor y_j, and $d(x_i, y_j)$ is the graph distance between both vertices x, y after moving to their neighbors $x_i,\ y_j$, respectively.

Theoretically, the Wasserstein distance could be computed quickly on graphs. This is also confirmed in practice. As a result, computing the Ricci curvature for our algorithm, consists in the previous simple calculations, is pretty fast for any data set. Among the many algorithms that were developed to construct a graph G from a data set, the most common are $k-$nearest neighbors (kNN) and $\epsilon-$nearest neighbors (ϵNN). In $kNN-$graphs, an edge e connects every vertex with its k nearest neighbors whereas an (ϵNN) graph connects an edge between vertices x and y if $d(x, y) < \epsilon$. However, for each data set, many graphs could be constructed depending on the value of k or ϵ. In addition to the effect of σ^2 on the positive Ricci curvature, simulations using the KBER method showed that the computed curvature would be also affected by the selected value of k to construct the $kNN-$graph. Generally, it is noticed that k and σ^2 have an inverse relationship where increasing k decreases σ^2 and vice versa. Consequently, the choice of k affects the value of the computed average Ricci curvature and the value of σ^2.

In practice, it is challenging to decide what is the suitable value of k. Although the parameter may be chosen empirically or through the use of heuristics, there are datasets in which there is not a single parameter value that can well-characterize the entire dataset (Bachmann *et*

al., 2010). In addition to the many heuristic methods presented in literature, a general rule of thumb is the square root of the number of instances for the data set. Unfortunately, none of the present methods was able to determine the suitable value of k in all cases and under all circumstances; for example when the number of instances or the number of dimensions is large or when the data set needs normalizing. A way to overcome the effect of this parameter on our method is to make use of ϵ_k which represents the distance to the k^{th} neighbor. Roughly speaking, the value of σ^2 yields a competitive recognition rate in the classification step when σ approaches ϵ_k, i.e when $|\epsilon_k - \sigma|$ is the smallest, which is confirmed in simulations.

4.4 A Bound for the RBF Bandwidth

A graph $G = (V, E)$ consists of two finite sets, the vertex set and the edge set denoted by V and E, respectively. In this article, we consider only undirected graphs where each edge e is represented by the unordered pair of its endpoints (x, y).

For the theoretical contribution in this work, we consider a toric grid graph on $\mathbb{Z}^2/(\mathbb{Z})^2$, known also as the toric lattice graph in two dimensions. It is the graph G whose vertices correspond to the points in the plane with integer coordinates where two vertices are connected by an edge whenever the corresponding points are at distance 1.

Theorem 4.1. *For a Markov chain defined on a toric* $2-$*dimensional grid, if* $\epsilon_k = 2$ *the average Ricci curvature on edge* $e = (x, y)$ *is bounded as follows*

$$\kappa(x, y) \geq \frac{1}{18} \exp\{-\frac{3}{\sigma^2}\} + \frac{1}{18} \exp\{-\frac{1}{\sigma^2}\}$$

Proof. Consider a Markov chain on a $2-$dimensional grid having n vertices with an associated graph obtained with the ϵNN method. Construct a circle centered at each vertex to contain all it's possible neighbors within distance $\epsilon_k = 2$ which could be counted to $k = 12$ neighbors as in the figure below.

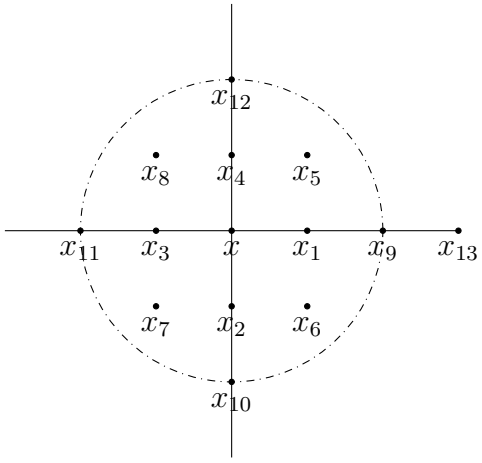

Figure 4.1: A circle of radius $\epsilon_k = 2$ centered at the origin x to contain its $k = 12$ neighbors.

For any edge $e = (x, y)$ inside the grid, let $d(x, y)$ denotes the Euclidean distance between vertices x and y then the one-step Markov chain is defined as

$$P(x \to y) = \frac{1}{C} \exp\Big\{-\frac{d(x,y)^2}{\sigma^2}\Big\}$$

if y is a neighbor among all the neighbors of x lying within the circle of radius $\epsilon_k = 2$ and

$$P(x \to y) = 0, \quad \text{otherwise.}$$

C represents the normalizing constant bounded as

$$\begin{aligned} C &= 4\exp\Big\{-\frac{1}{\sigma^2}\Big\} + 4\exp\Big\{-\frac{2}{\sigma^2}\Big\} + 4\exp\Big\{-\frac{4}{\sigma^2}\Big\} \\ &\leq 12\exp\Big\{-\frac{1}{\sigma^2}\Big\} \end{aligned} \tag{4.5}$$

In order to have an upper bound on the Wasserstein distance, we consider the following coupling:

Case 1: $(x, y) = (x, x_5)$, similar cases are given by $(x, y) \in \big\{(x, x_6), (x, x_7), (x, x_8), (x, x_9), (x, x_{10}), (x, x_{11}), (x, x_{12}\big\}$. Then, $W(m_x, m_y) \leq 1$ by considering the following coupling:

$P(x \to x+v, y \to y+v) = P(x \to x+v) = P(y \to y+v)$ for all $v \in \big\{(0,1),(0,2),(0,-1),(0,-2),(1,0),(2,0),(-1,0),(-2,0),(1,1),(1,-1),(-1,1),(-1,-1)\big\}$.
This case happen 8 out of 12 times and the Ricci curvature is then equal to

$$\begin{aligned}\kappa(x,y) &\geq 1 - W_d(m_x, m_y)\\ &\geq 0.\end{aligned}$$

Case 2: $(x,y) = (x,x_1)$, similar cases are given by $(x,y) \in \big\{(x,x_2),(x,x_3),(x,x_4)\big\}$. Here the coupling is different, on one side we have: $P(x \to x+v, y \to y+v) = P(x \to x+v) = P(y \to y+v)$ for all $v \in \big\{(0,1),(0,2),(0,-1),(0,-2),(1,1),(1,-1),(-1,1),(-1,-1)\big\}$.
On the other side, we gain positive curvature with $P(x \to x_3, y \to x_3) = \frac{1}{C}\exp\big\{-\frac{4}{\sigma^2}\big\}$, $P(x \to x_3, y \to x) = \frac{1}{C}\Big(\exp\big\{-\frac{1}{\sigma^2}\big\} - \exp\big\{-\frac{4}{\sigma^2}\big\}\Big)$ and $P(x \to x_{11}, y \to x) = \frac{1}{C}\exp\big\{-\frac{4}{\sigma^2}\big\}$. Additionally, $P(x \to x_4, y \to x_4) = \frac{1}{C}\exp\big\{-\frac{2}{\sigma^2}\big\}$, $P(x \to x_4, y \to x_5) = \frac{1}{C}\Big(\exp\big\{-\frac{1}{\sigma^2}\big\} - \exp\big\{-\frac{2}{\sigma^2}\big\}\Big)$ and $P(x \to x_8, y \to x_5) = \frac{1}{C}\exp\big\{-\frac{2}{\sigma^2}\big\}$.

Similarly, $P(x \to x_9, y \to x_9) = \frac{1}{C}\exp\big\{-\frac{4}{\sigma^2}\big\}$, $P(x \to y, y \to x_9) = \frac{1}{C}\Big(\exp\big\{-\frac{1}{\sigma^2}\big\} - \exp\big\{-\frac{4}{\sigma^2}\big\}\Big)$ and $P(x \to y, y \to x_{13}) = \frac{1}{C}\exp\big\{-\frac{4}{\sigma^2}\big\}$ and also $P(x \to x_2, y \to x_2) = \frac{1}{C}\exp\big\{-\frac{2}{\sigma^2}\big\}$, $P(x \to x_2, y \to x_6) = \frac{1}{C}\Big(\exp\big\{-\frac{1}{\sigma^2}\big\} - \exp\big\{-\frac{2}{\sigma^2}\big\}\Big)$ and $P(x \to x_7, y \to x_6) = \frac{1}{C}\exp\big\{-\frac{2}{\sigma^2}\big\}$.

Considering again this case yields a Ricci curvature:

$$\begin{aligned}\kappa(x,y) &\geq 1 - W_d(m_x, m_y)\\ &1 - \Big(1 - \frac{2}{C}\exp\big\{-\frac{4}{\sigma^2}\big\} - \frac{2}{C}\exp\big\{-\frac{2}{\sigma^2}\big\}\Big)\\ &\geq \frac{2}{C}\exp\big\{-\frac{4}{\sigma^2}\big\} + \frac{2}{C}\exp\big\{-\frac{2}{\sigma^2}\big\} \qquad (4.6)\end{aligned}$$

By dividing over the bound of C in equation 4.5,

$$\kappa(x,y) \geq \frac{1}{6}\exp\big\{-\frac{3}{\sigma^2}\big\} + \frac{1}{6}\exp\big\{-\frac{1}{\sigma^2}\big\}$$

and clearly since this case happens 4 out of 12 times then consequently, on average:

$$\kappa(x,y) \geq \frac{1}{18}\exp\{-\frac{3}{\sigma^2}\} + \frac{1}{18}\exp\{-\frac{1}{\sigma^2}\} \tag{4.7}$$

□

As a consequence, we have a bound on the mean Ricci curvature which is increasing with σ^2. In corollary 4.1, we take advantage of the convergence speed of the random walk defined by Ollivier (2009) to present an estimate of the bandwidth σ^2 according to the KBER method.

Corollary 4.1. *For a toric* $2-$*dimensional grid graph, the KBER method yields*

$$\sigma \in [2.58, 2.59].$$

Proof. According to the KBER method, we increase ϵ_k (or k) until $|\sigma - \epsilon_k|$ is minimized, equality to zero being the best possible result. For $\epsilon_k < 2$, the curvature $\kappa(x,y)$ is always equal to 0 whichever the value of σ. By theorem 4.1 we obtain that there exists $\sigma \in [2.58, 2.59]$ verifying that the curvature is greater than $1/k$ when $k = 12$. Thus, $\sigma - \epsilon_k$ will be equal to 0 for some $\epsilon_k \in [2, 2.59]$. □

The bandwidth we find in that case between 2 and 2.59 is of the same order of magnitude as the grid unit size giving confidence that this method is able to provide a sensible bandwidth in some cases.

4.5 Simulations

Many data sets were used to test the efficiency of the KBER method in estimating the bandwidth parameter. The Ricci curvature is used as a tool to estimate the suitable σ^2 needed for the classification by calibrating the value of σ^2 in equation 4.1 then a classification process has been conducted using this estimate. Besides, some data sets were completely normalized before building the $kNN-$graph if they showed a huge variability between the values of their dimensions, where ignoring this step could negatively affect the choice of σ^2. In fact,

the presence of a huge variability indicates the existence of measurements at different scales which yield a bias when selecting k for the $kNN-$graphs. As a result, this affects ϵ_k and consequently the accuracy of selecting σ^2.

Table 4.1: The recognition rate for the classification of data sets using the bandwidths estimated by the KBER method and the default method

			The Recognition Rate	
Data set	Instances	Dimensions	The KBER	The default
Tumors_C(Pomeroy *et al.*, **2002)**	60	7130	62.22%	44.44%
Dbworld-bodies(Filannino, 2011)	64	4702	72.92%	52.08%
SCADI(Zarchi *et al.*, **2018)**	70	206	71.15%	57.69%
rsctc2010_5**(Vanschoren** *et al.*, **2013)**	89	54614	66.15%	56.92%
rsctc2010_3**(Vanschoren** *et al.*, **2013)**	95	22278	54.93%	23.94%
rsctc2010_1**(Vanschoren** *et al.*, **2013)**	105	22284	53.85%	53.85%
LSVT(Tsanas *et al.*, **2014)**	126	309	76.60%	65.96%
mouseType(Vanschoren *et al.*, **2013)**	214	45101	64.38%	45.00%
Clean(1)**(Vanschoren** *et al.*, **2013)**	476	169	93.00%	90.20%
Anuran Calls(MFCCs)(Dua and Graff, 2019)	7195	22	98.00%	88.66%
Grid Stability(Arzamasov *et al.*, **2018)**	10000	14	89.77%	89.71%
Epileptic Recognition (Andrzejak *et al.*, **2001)**	11500	178	53.96%	42.45%
HTRU2**(Lyon** *et al.*, **2016)**	17898	8	97.72%	97.65%
Letter Recognition(Dua and Graff, 2019)	20000	16	92.67%	91.85%
Avila(De Stefano *et al.*, **2018)**	20867	10	60.17%	58.70%
Crowdsourced (Johnson and Iizuka, 2016)	102944	116	98.09%	91.23%

Table 1 presents a comparison between the recognition rates for the classification of data sets when using two values for the bandwidth; the first value is estimated by the KBER method whereas the second value is calculated by a default heuristic method. The simulations were done using a MSVM open source server (Lauer and Guermeur, 2011) and the default bandwidth used in this server is defined as $\sigma = \sqrt{5 \times \dim(x)}$, where $\dim(x)$ represents the number of dimensions for the data set x.

It is clear from the table above that our method yields competitive results in addition to its superiority over the default one, especially when increasing the number of instances and the number of dimensions.

4.6 Conclusion

We have proposed a simple and fast method to estimate the bandwidth value of the RBF kernel. Mathematically, according to the KBER method, we showed that the parameter could be estimated in simple calculations since it depends basically on computing the Wasserstein distance in a simple optimization problem. Also, corollary 4.1 shows that in a simple setting, i.e when considering a graph in $2-$dimensions and $\epsilon_k = 2$, the Ricci curvature yields a perfect estimation of the bandwidth.

According to the simulations, KBER method yielded competitive results in all possible settings even when increasing the number of dimensions. In fact, it didn't show shortages under any circumstances when considering a suitable k for the constructed $kNN-$graph. Also, it is worth mentioning that, following to the results of KBER method, we confirm the argument of the possibility to increase the kernel performance accuracies by considering values out of the range $0.2 - 1$. Further work would involve providing more theoretical proofs of the soundedness of our method.

Bibliography

A. Tsanas, M. Little, C. Fox, and L. Ramig. Objective Automatic Assessment of Rehabilitative Speech Treatment in Parkinson's Disease. *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* vol. 22, no. 1, pp. 181-190, 2014.

B. Johnson and K. Iizuka. Integrating OpenStreetMap crowdsourced data and Landsat time-series imagery for rapid land use/land cover (LULC) mapping: Case study of the Laguna de Bay area of the Philippines. *Applied Geography,* 67, pp. 140–149. 2016.

B. Schölkopf. *In personal communication with the authors of:* Kernel Width Selection for SVM Classification: A Meta-Learning Approach. 2003.

C. Bachmann, T. Ainsworth, R. Fusina, R. Topping and T. Gates. Manifold coordinate representations of hyperspectral imagery: Improvements in algorithm performance and computational efficiency. Geoscience and Remote Sensing Symposium (IGARSS). *IEEE,* pp. 4244–4247. 2010.

C. Chang and C. Lin. LIBSVM: a library for support vector machines. 2001. http://www.csie.ntu.edu.tw/cjlin/libsvm

C. De Stefano, M. Maniaci, F. Fontanella, A. Freca. Reliable writer identification in medieval manuscripts through page layout features: The 'Avila' Bible case. *Engineering Applications of Artificial Intelligence,* 72, pp. 99–110. 2018.

C. Ni, Y. Lin, J. Gao, D. Gu and E. Saucan. Ricci curvature of the Internet topology. *Proceedings of the IEEE Conference on Computer Communications, INFOCOM 2015, IEEE Computer Society.* 2015.

C. Soares, P. Brazdil and P. Kuba. A meta-learning method to select the kernel width in support vector regression. *Machine Learning,* 54, 3, pp. 195–209. 2004.

D. Cushing, R. Kangaslampi, V. Lipiäinen, S. Liu and G. Stagg. The Graph Curvature Calculator and the curvatures of cubic graphs. *arXiv:1712.03033.* 2017.

D. Dua and C. Graff. *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. 2019

F. Chung and S. Yau. Logarithmic Harnack inequalities. *Mathematical Research Letters,* vol. 3, pp. 793–812. 1996.

F. Lauer and Y. Guermeur. MSVMpack: a Multi-Class Support Vector Machine Package. *Journal of Machine Learning Research,* 12:2269-2272. 2011.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research,* 12, pp. 2825–2830. 2011.

G. Felici and C. Vercellis. *Mathematical methods for knowledge discovery and data mining.* Information Science Reference IGI Global Publishers, UK, 2008.

G. Hardy. *Ramanujan: Twelve Lectures on Subjects Suggested by His Life and Work.* 3rd ed. New York: Chelsea. 1999.

H. Nakayama, M. Arakawa and R. Sasaki. Simulation-Based Optimization Using Computational Intelligence. *Optimization and Engineering,* Vol.3, pp. 201-214. 2002.

J. Jost and S. Liu. Ollivier's Ricci curvature, local clustering and curvature dimension inequalities on graphs. *Discrete Comput Geom,* 51:300. 2014. [Online].Avilable:https://doi.org/10.1007/s00454-013-9558-1

J. Lott and C. Villani. Ricci curvature for metric-measure spaces via optimal transport. *Annals of Mathematics,* vol. 169, pp. 903–991. 2009.

J. Vanschoren, J. Rijn, B. Bischl and L. Torgo. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations,* 15(2), pp. 49–60, 2013.

K. Muller, S. Mika, G. Rätsch, K. Tsuda and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transaction on Neural Networks,* 12(2), pp. 181–201. 2001.

K. Sturm. On the geometry of metric measure spaces. Springer, *Acta Mathematica.* 2006.

M. Filannino DBWorld E-mail Classification Using a Very Small Corpus. *Project of machine learning course, university of Manchester.* 2011.

M. Hazewinkel. Wasserstein metric. *The Online Encyclopaedia of Mathematics,* 2011. http://eom.springer.de/W/w120020.htm.

M. Zarchi, S. Bushehri and M. Dehghanizadeh. SCADI: A Standard Dataset for Self-Care Problems Classification of Children with Physical and Motor Disability. *In International Journal of Medical Informatics,* Vol. 114, pp. 81–87, 2018.

N. Benoudjit and M. Verleysen. On the Kernel Widths in Radial-Basis Function Networks. *Neural Processing Letters,* Vol.18, pp. 139–154. 2003.

O. Chapelle, V. Vapnik, O. Bousquet and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning,* 46(1), pp. 131-159. 2002.

R. Andrzejak, K. Lehnertz, C. Rieke, F. Mormann, P. David and C. Elger. Indications of nonlinear deterministic and finite dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Phys. Rev. E,* 64:061907. 2001.

R. Lyon, B. Stappers, S. Cooper, J. Brooke and J. Knowles. Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society,* 459(1), pp. 1104–1123, 2016. DOI: 10.1093/mnras/stw656

S. Ali and K. Smith. Kernel width selection for SVM classification: A meta-learning approach. *International Journal of Data Warehousing and Mining.* 1, pp. 78–97. 2007.

S. Haykin. Neural Networks: A Comprehensive Foundation. *Macmillan College Publishing Company,* pp. 236–284. 1994.

S. John and C. Nello. *Kernel Methods for Pattern Analysis.* Cambridge, U.K.: Cambridge University Press. 2004.

S. Pomeroy, P. Tamayo, M. Gaasenbeek, L. Sturla, M. Angelo, M. McLaughlin, J. Kim, L. Goumnerova, P. Black, C. Lau, J. Allen, D. Zagzag, J. Olson, T. Curran, C. Wetmore, J. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D. Louis, J. Mesirov, E. Lander and T. Golub. Prediction of Central Nervous System Embryonal Tumour Outcome based on Gene Expression. *Nature,* 415, pp. 436–442, 2002.

V. Arzamasov, K. Böhm, and P. Jochem. Towards Concise Models of Grid Stability. *Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), 2018 IEEE International Conference on. IEEE.* 2018.

Y. Ollivier. Ricci curvature of Markov chains on metric spaces. *Journal of Functional Analysis,* 256, pp. 810–864. 2009.

Conclusion

This thesis is mainly devoted to MCMC graph sampling methods and applications. In chapter 2, we developed two algorithms to sample unifrom subtrees from graphs. The first method is an independent Metropolis-Hastings algorithm which is evaluated to converge in a speed $\mathcal{O}(s^{-1}nr^{k^2/2})$ whereas it is $\mathcal{O}(\lambda_1^{-1}a^2d_{max}^2k^5)$ for the second non-independent method. The theoretical results were confirmed using a simulations study. The independent method showed better performance only when sampling subtrees from a graph with a narrow bottleneck whereas the crawling method was the best in the case of sampling from other types of graphs. A conclusion would be recommended to use a combination of both methods for sampling uniform subtrees, i.e to sample trees in two steps. Firstly one should sample subtrees globally from the graph using the independent method and after that to sample locally through the crawling method.

In chapter 3, another graph sampling method is presented to sample k close vertices in the graph. The objective in this work is achieved by sampling using a metropolis-Hastings algorithm called kPPP. The complexity of computing the permanent is overcome by sampling a joint distribution whose marginal distribution is a kPPP. Considering a small value for k, the bound on the convergence speed of the designed algorithm motivates to consider our stationary distribution for the purpose of sampling close vertices, in addition to provide tighter bounds when it increasing k.

Chapter 4 presents an application in data analysis and Machine learning for the definition of Markov chains over graphs. It presents a simple and fast method called KBER method to

estimate the bandwidth value of the RBF kernel. We showed that the bandwidth parameter could be estimated in simple calculations unlike other methods in literature. In our mathematical contribution when considering a graph in $2-$dimensions and $\epsilon_k = 2$, we showed that the Ricci curvature yields a perfect estimation of the bandwidth. In contrast to the existed heuristic methods, the simulations of KBER method yielded competitive results in all possible settings even when considering a suitable k for the constructed $kNN-$graph.

In summary, the objectives of the thesis were achieved. The importance of our work is not only that it answers the already asked questions, but it opens possible future research directions where this thesis is only the beginning of the author future research.

Perspectives

Working on various applied issues motivated to answer other questions rolled out during the thesis besides continuing and developing the current results. A future research would be considered in:

- Perspective on chapter 2

  Firstly, following to the result at the end of chapter 2, the forthcoming research is to develop a new method combines both of the already developed methods especially that the target distribution is the same, which could overcome the limitations when using each method separately.

  Additionally, we considered the uniform distribution as the Markov chain stationary distribution in order to sample random trees. It has an easy application and widespread use. In contrast, we have the impression that simulating according to another distribution related to data could summarize more information from the population. This would be interesting although it is hard in similar applications in Biology where the uniform distribution is the indispensable choice.

  Moreover, trees structures have been proved their efficiency to describe graphs, and sampling these structures yields interesting results to summarize information. A question could be asked here, could we sample more complicated structures from the graph in a close efficient method.

  The main question in chapter 2 was: Is it possible to sample a bigger structure than pathways that could provide more information to identify the important metabolites

which are driving the function of the network considered as a graph. An indispensable perspective is to study the efficiency of our methods with an application in a cooperation with Biology.

- Perspective on chapter 3

  According to the bound obtained for the convergence speed, a proposed perspective is to search for a much tighter bound by looking for other methods to bound the convergence speed. An idea to enhance our bound is to construct sets of canonical paths instead of the canonical paths method which consider the flow on one and only one path between each pair of vertices. Further, for the aim of obtaining a tighter bound, we think about bounding the convergence speed using the Wassermann distance instead of the total variation distance where it is known that many methods converge using it unlike the latter measure although it is usually used for continuous Markov chains. Furthermore, it is preferable in the coming future to conduct simulations studies to evaluate experimentally the convergence speed.

  Basically, the work in this chapter came into sight following to a question in a video game project asked to sample sets of close vertices where a video game would be considered as a graph. The objective is achieved by designing a Markov chain to sample a joint distribution whose marginal distribution is a kPPP which samples k vertices among the n vertices. Further studies would link practically this work to video games and other fields in order to discover its limitations and weakness, which helps to enhance it.

- Perspective on chapter 4

  We proved that the Ricci curvature for a $2-$dimensional graph yields a competitive estimator for the RBF kernel bandwidth. A perspective on this work is to generalize similar results for $d-$dimensional graphs. In fact, designing a similar coupling for a d-dimensional graph would be possible but seems complicated. Another possibility is to present different theoretical proofs by considering other distances over graphs.

  Moreover, we proved the efficiency of our method to estimate σ^2 by comparing the

recognition rate for a classification while using our estimate and a heuristic method depends on the number of dimensions to estimates the same parameter. For the completeness of this work, we look to compare it with other common methods like the cross validation which is a common choice in Machine learning. Additionally, more technical details, but important, would be considered in comparison like the comparing other methods to the needed time for our method to estimate the hyper-parameter.

Selecting the suitable value of k for the $kNN-$graph is an issue in Statistics. No perfect method in literature where all existed methods are heuristics. Through the work on this project, a strong relation was noticed between the positive Ricci curvature and the suitable value for k. Further studies are needed to discover this relation profoundly in order to develop a more certified method to select k.

Upload our code, written to estimate the bandwidth, to a library in order to use the KBER method easily and quickly.