Extending convolutional neural networks to irregular domains through graph inference

Bastien Pasdeloup

▶ **To cite this version:**

Bastien Pasdeloup. Extending convolutional neural networks to irregular domains through graph inference. Machine Learning [cs.LG]. Ecole nationale supérieure Mines-Télécom Atlantique, 2017. English. NNT : 2017IMTA0048 . tel-01708824v1

HAL Id: tel-01708824

https://theses.hal.science/tel-01708824v1

Submitted on 14 Feb 2018 (v1), last revised 14 Feb 2018 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UNIVERSITE BRETAGNE LOIRE

THÈSE / IMT Atlantique

sous le sceau de l'Université Bretagne Loire

pour obtenir le grade de

DOCTEUR D'IMT Atlantique

Mention : Informatique

École Doctorale Mathématiques et STIC

Présentée par

Bastien Pasdeloup

Préparée dans les départements Electronique, Signal & communications, Image & traitement de l'information

Laboratoire Labsticc

Extending Convolutional Neural Networks to Irregular Domains through Graph Inference

Thèse soutenue le 12 décembre 2017

devant le jury composé de :

Christine Guillemot
Directrice de recherche, Inria – Rennes / présidente

Pascal Frossard
Associate Professor, Ecole Polytechnique de Lausanne / rapporteur

Pierre Borgnat
Directeur de recherche, Ecole normale supérieure de Lyon / rapporteur

Rémi Gribonval
Directeur de recherche, Inria – Rennes / examinateur

Vincent Gripon
Chargé de recherche, IMT Atlantique / examinateur

Dominique Pastor
Professeur, IMT Atlantique / directeur de thèse

Michael G. Rabbat
Permanent Researcher, McGill University / invité

Table des matières

Acknowledgments

So… it is now time for what I believe to be one of the most complicated parts in the redaction of this manuscript: trying not to forget anyone. Honestly, I already know that I will since this manuscript wouldn't exist without countless persons, so please forgive me if it is the case for you. Anyway, in order to try to be as close as possible to an accurate list of acknowledgments, I believe I should slightly organize this chapter:

Family

First of all, all my thanks go to my parents, Nicole and Gilles, who continuously supported me during my life/studies, and for countless other obvious reasons. Also, thanks to my sisters, Pauline and Marion, and to their relatives, Mathilde and Adrien, for being here and for trying to understand at least the summary of my manuscript :)

Then, all my love goes to my wife Lauren, as well as to my newborn daughter Ambre. Thank you Lauren for everything, and thank you Ambre for everything to come. When you're old enough to read this, please note that it was a great decision to prevent me from sleeping the night before the submission deadline for this manuscript. I could find some remaining typos that I corrected in due time, so thanks for your *encouragement*!

Obviously, I'd also like to thank the rest of my family, past or present. May you live far away from the distant lands of Finistère, it is great to have you. Similarly, thanks to Lauren's side of the family, and especially her parents, Arlette and Patrick.

Friends

I would obviously not be the same without all the friends I made across the years. In particular, I would like to thank Pascal (Papy), for once saying to me "You should try computer science. It's just like playing Magic: the Gathering, so you should be okay with it". Well, not sure the argument was the best, but that was definitely a good idea! Not completely unrelated, I'd like to thank all the guys I've been playing Magic (among other games) with since I'm a kid (by the way, I'd like to thank Richard Garfield, eventhough I do not know him personnally). Hard to stay in contact with each and every one of you after all these years, but you had a strong impact on my ability to use my brain.

I would also like to thank all my band mates, and particularly the guys from Tenval and Aboriscor, for sharing so many nice moments during the shows and rehearsals. Continuing with music companions, thanks to the debarkators for the great discoveries and for all the fun, including Michale's B6 menus, Steven's T9 and Benjamophagher's

13th birthday (don't we have a problem with numbers?). See you next year in Tilburg and Glasgow!

I would also like to thank all the friends I made during my studies, and particularly my mates from ENS Rennes. Thanks to Simon and Malfoy for the laughs, and for the best birthday cake of my life. Thanks to Justine, Aurore, Hugo and Gurvan for all the great moments, and sorry for being so bad at playing League of Legends. I hope your future projects will bring more gratitude than ÜberML!

Education

I would like to thank the various schools I studied in during my postgraduate years, in addition to the associated teachers and staff.

In chronological order, thanks to my teachers at Charles de Foucauld, who helped me acquire strong programming skills, and to Cyrille Baudouin, my internship advisor at the Centre Européen de Réalité Virtuelle (CERV), who contributed to my taste for research. More generally, I would like to thank all the researchers from the CERV for the nice environment and help, and to the teachers from École Nationale d'Ingénieurs de Brest (ENIB), where I studied the next year.

Then, I would like to express my appreciation to all the teachers and staff at École Normale Supérieure (ENS) Rennes, formerly ENS Cachan — Antenne de Bretagne. In particular, thanks to Luc Bougé for welcoming me there, and having faith in my ability to succeed. To all of my teachers there, thank you for your patience and for everything I learned from you. Simultaneously, I would like to thank my teachers from Université de Rennes 1, particularly Isabelle Puaut and Steven Derrien for the interesting projects and human aspects.

Finally, I would like to thank IMT Atlantique, formerly Télécom Bretagne, for trusting me so much during my Ph.D. Thanks for the great research context, and for all the responsabilities concerning the PyRat course, which was a great opportunity for a Ph.D. student. In particular, my thanks go to Philippe Picouet (and more generally to all those thanks to whom the project was born), to all the teachers that accepted to participate in this course, and to the students I supervised during these three years.

Research

First of all, I would like to express all my gratitude to my Ph.D. advisor, Vincent Gripon, without whom I would probably not be doing research at this time. Thank you for everything you did for me, may it be during my studies at ENIB, during these three years of Ph.D in which you taught me a lot, or for the PPP/SSS ;) I hope I can pay you back someday for all of this!

More generally, I would like to thank all the researchers and staff from the Electronics department, as well as the interns that passed some time there. In particular, thanks to Nicolas Grelier who made hell of a job during his internship, and to each and every member of the Neucod/Brain teams for the inspiring discussions and great moments. I would also like to thank Catherine Blondé for helping me going through the tangle of administration, and Michel Jézéquel for the continuous support.

Then, I would like to thank Dominique Pastor (and more generally the SC department), and formerly Grégoire Mercier, for directing my Ph.D. This was great working with you. To Grégoire, I hope you are having fun in your new employment!

I would also like to thank Pierre Vandergheynst and his team at École Polytechnique Fédérale de Lausanne (EPFL) for welcoming me there at the beginning of my Ph.D. The ideas that emerged from this visit lead to important parts of this manuscript! Again at EPFL, I would like to address my thanks to Pascal Frossard for accepting to review my manuscript, and for offering me to work in your team next year!

More generally, I would like to thank all the members of the jury. Thanks to Pierre Borgnat for accepting to review my manuscript as well, and for all the nice moments across these years. It is always a pleasure! Thanks to Rémi Gribonval and Christine Guillemot for accepting to participate in my jury. I hope you enjoyed my work!

Finally, I would like to express particular thanks to Michael (Mike) G. Rabbat, for the continuous support and wise suggestions all along my Ph.D. Additionally, thank you and the McGill University for welcoming me in Montréal for two awesome months.

To conclude my acknowledgments, I would like to thank the editors, associate editors and reviewers of my research papers, for helping me improving my work, as well as the GdR ISIS for financially helping me go to Montréal. Also, I would like to thank the organizers of the Graph Signal Processing (GSP) Workshop for putting this up, and Mike Rabbat for the Barbados seminar on GSP, which was a real dream!

Participants to the Barbados meeting. Photo © Benjamin Girault. Standing, from left to right: Jonathan Mei, Augustin-Alexandru Saucan, Siheng Chen, Vincent Gripon, Antonio G. Marques, José M. F. Moura, Michael G. Rabbat, Mark Coates, Benjamin Girault, Gonzalo Mateos. Sitting, from left to right: Nicolas Farrugia, Pierre Borgnat, Bastien Pasdeloup, Paulo Gonçalves.

Résumé

Contexte

Le traitement de signal sur graphe

Il y a quelques années, le *traitement de signal sur graphe* est né de l'observation suivante : un signal défini dans le domaine temporel peut être représenté par un vecteur de réels dont chaque entrée correspond à la valeur du signal à un instant donné. Ce temps, s'il est périodique, peut être modélisé par un graphe prenant la forme d'un anneau (voir la Figure 4), où chaque sommet représente un instant d'échantillonage, et où une arête existe entre deux sommets si les instants représentés sont contigus dans le temps.

Sous ce formalisme, on manipule donc deux types d'objets : un graphe modélisant le support des signaux, et un signal sur ce support. Le traitement de signal sur graphe est le domaine de l'analyse de tels signaux, observés sur des graphes aux structures variées. Entre autres exemples de signaux sur graphes, on trouve des images portées par des grilles de pixels, ou encore des observations de signaux électro-encéphalographiques évoluant sur un graphe modélisant le cerveau ou la surface du crâne.

L'intérêt de passer par le formalisme du traitement de signal sur graphe est qu'il permet de prendre en compte la structure sur laquelle évolue un signal, ce qui apporte donc plus d'information que de considérer le signal seul. En particulier, l'un des résultats marquants du domaine est une analogie entre les vecteurs propres d'une matrice représentant le graphe, le *Laplacien*, et les modes de Fourier. En effet, si l'on considère un graphe anneau, que l'on calcule ses vecteurs propres judicieusement, et qu'on les ordonne par ordre croissant des valeurs propres associées, on remarque que ces vecteurs prennent la forme d'un cosinus, puis d'un sinus, d'un cosinus de fréquence double, etc. La Figure 5 offre une représentation graphique de ces vecteurs propres.

Ainsi, on observe que les vecteurs propres du Laplacien du graphe sont analogues aux modes de Fourier, et les valeurs propres aux fréquences correspondantes. Projeter un signal sur graphe dans la base des vecteurs propres du Laplacien revient donc à effectuer une transformée de Fourier de ce signal. Il est ensuite possible par exemple d'atténuer les composantes associées aux plus hautes valeurs propres, ce qui a pour effet de lisser le signal, à la manière d'un filtre passe-bas en traitement de signal classique. De même, il est possible de définir de nombreux autres opérateurs permettant le traitement de ces signaux, comme la modulation, la convolution, la translation, etc.

Au delà de l'analogie avec le graphe anneau, le formalisme du traitement de signal sur graphe s'applique à tout type de graphe, et les notions présentées précédemment y sont parfaitement définies. Par exemple, la Figure 19 illustre la propriété de lissage du signal par un filtre passe-bas sur un graphe aléatoire. Le traitement de signal sur graphe permet donc l'étude de signaux évoluant sur des structures complexes, apportant de

nouvelles perspectives en termes de compréhension de ces signaux, et permettant des tâches plus concrètes telles que leur classification.

Quel graphe utiliser ?

Comme indiqué précédemment, le traitement de signal sur graphe permet l'étude de signaux grâce à la connaissance de la structure sur laquelle ils évoluent. En particulier, les vecteurs propres de la matrice Laplacienne définissent une transformée de Fourier adaptée aux signaux, à partir de laquelle de nombreux autres opérateurs sont définis.

Toutefois, s'il est facile d'imaginer que des images sont des signaux portés par des graphes prenant la forme de grilles de pixels, ou qu'un signal temporel évolue sur un graphe ligne ou sur un anneau, il est plus difficile de s'accorder sur une structure représentant le cerveau humain, ou de modéliser le support d'observations sismiques. Une première solution pour pouvoir traiter ces signaux serait d'avoir recours aux outils classiques du traitement de signal plutôt qu'à ceux du traitement de signal sur graphe, car non applicables pour cause d'absence d'une représentation du support. Une seconde option, plus satisfaisante, est d'inférer un graphe à partir des données, afin de reconstruire la structure permettant l'étude de tels signaux.

Toutefois, le nombre de graphes pondérés possibles est infini, et il est nécessaire de faire des suppositions sur les signaux ou sur le graphe. Entre autres suppositions classiques, on trouve la parcimonie du graphe, sa régularité, le fait que ses poids soient binaires, ou que les signaux soient lisses sur le graphe, etc. Dans ce manuscrit, nous considérons un modèle de signaux stationnaires, et proposons plusieurs stratégies d'inférence de graphe permettant l'obtention de propriétés choisies sur ce graphe.

Quelle définition pour la translation d'un signal sur un graphe ?

De nombreux opérateurs de manipulation de signaux portés par des graphes découlent de la transformée de Fourier sur graphe, donc du graphe. En particulier, la notion de translation d'un signal n'est pas intuitive lorsque l'on considère une structure abstraite et irrégulière, sur laquelle la notion de distance n'est pas bien définie. L'opérateur de translation sur graphe a donc été défini par analogie avec une propriété de la translation en traitement de signal classique. En effet, translater un signal dans le domaine temporel est équivalent à le convoluer avec une impulsion de Dirac localisée en un instant choisi. La convolution sur graphe étant définie comme un produit dans le domaine spectral, la translation d'un signal sur graphe est introduite comme telle. Ainsi, un signal sur graphe n'est pas réellement translaté dans une direction, mais relocalisé en un endroit par une impulsion de Dirac en un sommet.

Toutefois, la structure du signal dans le domaine du graphe n'est pas préservée par cet opérateur de translation. Ainsi, un signal très localisé dans le domaine du graphe pourra s'étaler sur un grand nombre de sommets une fois convolué avec une impulsion de Dirac localisée en un sommet du graphe. À titre d'exemple, la Figure 41 présente une image, ainsi que sa translation par cet opérateur, démontrant que les motifs de l'image ne sont plus identifiables après translation.

L'un des objectifs de ce manuscrit est de proposer une définition alternative de la translation utilisant uniquement le domaine du graphe, et non le domaine spectral associé. Une telle définition devrait préserver la structure d'un signal lors de sa translation, permettant la préservation de motifs dans le signal.

En quoi le traitement de signal sur graphe peut-il aider à la classification ?

La classification de signaux est un problème classique, aux applications multiples. Par exemple, pour des observations de signaux cardiaques, on aimerait pouvoir classifier le patient comme à risque ou non. Dans le développement des voitures autonomes, établir si un signal fourni par les différents capteurs représente un piéton, une voiture, un feu de signalisation… est une opération capitale.

À l'heure actuelle, les algorithmes les plus efficaces pour effectuer des tâches de classification à partir d'images sont les réseaux de neurones convolutifs. De tels réseaux utilisent la localité des objets dans les images afin de mieux les détecter. Techniquement, ces réseaux fonctionnent grâce à une fenêtre de quelques pixels sur l'image à classifier, qui est translatée en tout point de l'image. Pour cette raison, si l'opérateur de translation n'est pas défini car les signaux sont plus complexes que des images, de tels réseaux ne peuvent être créés.

C'est précisément en cela que le traitement de signal sur graphe peut aider. Dans ce manuscrit, nous proposons une extension des réseaux de neurones convolutifs à des domaines irréguliers de la manière suivante :

1. À partir d'un ensemble de signaux d'apprentissage, nous inférons un graphe ;
2. Ensuite, nous identifions des translations sur ce graphe ;
3. Enfin, nous utilisons ces translations pour définir un réseau de neurones convolutif adapté aux données à classifier.

Principales contributions et résultats

L'objectif de ce manuscrit est de proposer une extension des réseaux de neurones convolutifs à des domaines irréguliers, basée sur les signaux, grâce à des outils issus du traitement de signal sur graphe. Nous procédons en trois étapes :

Inférence de graphes à partir de signaux

Pour un ensemble de signaux à étudier, il est fréquent que l'on ne dispose pas d'un graphe expliquant la structure sur laquelle ils évoluent. Une approche permettant de se ramener au formalisme du traitement de signal sur graphe, malgré cette difficulté, consiste en l'inférence d'un graphe à partir des données. Toutefois, si le nombre de sommets est connu, car correspondant au nombre de variables pour lesquelles on observe des valeurs de signal, définir un ensemble d'arêtes reliant ces sommets implique de faire des choix.

Dans ce manuscrit, nous avons choisi de considérer un modèle de signaux stationnaires. De tels signaux peuvent être générés synthétiquement à partir d'un bruit blanc, que l'on diffuse sur le graphe par une matrice définie à partir de celui-ci. Un tel processus de diffusion tend à modifier les entrées du signal, le colorant par la structure du graphe, tout en conservant sa stationnarité. Les signaux diffusés encodent donc partiellement des informations sur la structure du graphe, que nous voulons extraire afin de retrouver la matrice ayant servi à diffuser le bruit blanc et, par extension, le graphe.

Dans ce manuscrit, nous avons montré la propriété suivante : *les vecteurs propres de la matrice de covariance des signaux sont aussi ceux de la matrice ayant servi à leur diffusion*

sur le graphe. Afin de parfaitement caractériser le graphe, il reste donc à déterminer les valeurs propres à associer à ces vecteurs propres. Nous avons distingué deux cas :

1. Si le processus de diffusion est simple, et que chaque signal est diffusé un nombre connu et constant de fois, nous avons montré qu'il est possible de caractériser les valeurs propres manquantes. Celles-ci peuvent être obtenues par calcul d'une racine des valeurs propres de la matrice de covariance, puis par obtention de leur signes via la résolution d'un problème d'optimisation ;
2. Dans le cas plus générique d'un processus de diffusion quelconque, aucune information sur les valeurs propres n'est disponible.

Dans le second cas, il convient donc d'ajouter un certain nombre d'hypothèses sur la matrice à inférer. Considérant des processus de diffusion, les matrices qui nous intéressent contiennent des entrées positives, car représentant une quantité de signal propagée à chaque étape. Grâce aux vecteurs propres connus, nous pouvons donc définir un ensemble convexe de points, correspondant à des vecteurs de valeurs propres admissibles respectant ce critère. Un exemple d'un tel ensemble convexe est donné en Figure 33.

La recherche de valeurs propres à associer aux vecteurs propres de la matrice de covariance dépend donc ensuite d'un choix, correspondant à des suppositions sur le graphe. Nous avons proposé trois approches pour choisir un point dans cet ensemble convexe :

1. Inférer une matrice simple, c'est-à-dire à la diagonale nulle, qui représente donc un processus maximisant la diffusion des signaux ;
2. Inférer un graphe parcimonieux, contenant peu d'arêtes ;
3. Utiliser une méthode de l'état de l'art, et l'adapter à un a priori de stationnarité en sélectionnant le plus proche point dans l'ensemble convexe.

Nous avons appliqué ces solutions à des données synthétiques, ainsi qu'à des données réelles issues d'observations météorologiques en Bretagne, France. De plus, la troisième stratégie offre des applications potentielles au problème de choisir, pour un ensemble de graphes donnés, lequel est le plus adapté à la représentation d'un ensemble de signaux stationnaires. Enfin, nous avons observé lors de la définition de réseaux de neurones convolutifs à partir de données, que la régularisation d'une solution d'inférence avec a priori de lisseur des signaux permet l'obtention d'un graphe offrant les meilleures performances de classification.

Identification de translations sur graphe

Les définitions existantes pour l'opérateur de translation sur graphe ne préservent pas la structure du signal dans le domaine du graphe. En effet, ces opérateurs sont définis grâce au domaine spectral associé aux vecteurs propres de la matrice Laplacienne. Dans sa définition la plus simple, la translation d'un signal est effectuée par une convolution avec une impulsion de Dirac localisée en un sommet, et non par suite de transferts d'information d'un sommet à l'autre. D'autres opérateurs de translation existent, mais des problèmes similaires sont observés.

Dans ce manuscrit, nous avons proposé une nouvelle définition de l'opérateur de translation sur graphe, analogue à la notion de translation sur des espaces Euclidiens. Nos translations consistent en une orientation d'un sous-ensemble des arêtes du graphe, et se résument par trois propriétés :

1. Une translation est une fonction injective des sommets vers les sommets ;

2. Chaque sommet ne peut être envoyé que sur un sommet adjacent ;
3. Les voisins d'un sommet doivent être envoyés sur les voisins de la destination de ce sommet, *i.e.*, les voisinages doivent être conservés.

Ces propriétés simples permettent de conserver la forme d'un signal défini sur le graphe lors de sa translation. Toutefois, de telles translations n'existent que sur des graphes extrêmement réguliers. Nous avons donc introduit dans ces critères la possibilité qu'une valeur de signal sur un sommet soit *perdue* lors de l'application d'une translation, modélisant ainsi des problèmes tels que les effets de bord lors de la translation d'une image sur un graphe grille non torique.

Considérant un tel graphe grille, nous avons montré que les translations minimisant la perte d'information sont celles envoyant chaque sommet dans une direction correspondant à chacun des points cardinaux, représentées en Figure 48. Cela correspond exactement aux translations Euclidiennes classiques sur des domaines où une telle notion de direction existe. Les critères définissant nos translations n'utilisent pas cet espace métrique, ce qui rend les rend applicables à tout type de graphe.

Toutefois, identifier les meilleures translations sur un graphe (au sens de la perte de la translation) n'est pas une tâche aisée. En effet, nous avons montré que leur identification est un problème NP-complet. Cela nécessite donc la définition d'une stratégie approchée, permettant l'obtention de pseudo-translations, respectant en partie les trois critères précédemment énoncés. En détails, nous avons choisi de sacrifier en partie la préservation des voisinages, et avons formulé notre problème sous la forme d'un problème d'optimisation. Ainsi, nous avons pu obtenir des pseudo-translations intéressantes sur des graphes issus de modèles aléatoires, comme présenté en Figure 50.

Extension des réseaux de neurones convolutifs à des domaines irréguliers

Les réseaux de neurones convolutifs ont montré un fort intérêt dans des tâches telles que la classification d'images. De tels réseaux définissent les connexions entre couches grâce à un noyau convolutif, c'est-à-dire une petite fenêtre de pixels centrée en un point. Ce noyau est translaté en tout point de l'image, et des arêtes sont créées entre les neurones associés aux pixels couverts par le noyau, et celui correspondant à son centre. L'opérateur de translation permet donc de décaler un point de vue local sur l'image, et donc d'y détecter des objets, car généralement portés par des pixels adjacents.

Comme une notion de translation analogue n'était pas définie dans le domaine du graphe pour le cas de graphes irréguliers, il n'était pas possible d'étendre simplement de tels réseaux pour des signaux évoluant sur des structures complexes. L'identification de translations telle que nous l'avons proposée a permis d'effectuer cela, et donc de translater un noyau en différents points d'un graphe, permettant donc la définition de réseaux de neurones convolutifs pour le traitement de signaux portés par des graphes. Combinée à nos travaux sur l'inférence de graphe, cette technique permet donc la définition de réseaux convolutifs adaptés aux signaux à classifier.

Nous avons illustré nos résultats sur la base de données CIFAR-10 d'images de dix classes différentes. Sans utiliser l'information que ces signaux sont des images, la méthode d'inférence nous permet de retrouver un graphe proche d'une grille de pixels, et donc d'obtenir des performances comparables à des réseaux convolutifs utilisant cette information. Inférer le graphe, puis les translations sur celui-ci, nous a permis d'obtenir un gain en performances de 13.11 points sur cette base de données par rapport aux

méthodes de l'état de l'art n'utilisant pas non plus cette information, avec un réseau convolutif simple et facilement améliorable par une recherche de ses paramètres.

Notre méthode est donc une réelle extension des réseaux de neurones convolutifs, qui peuvent être vus comme un cas particulier de notre approche, où un graphe grille est considéré. La méthode proposée dans ce manuscrit offre donc de très nombreuses possibilités d'applications, dans des domaines aussi variés que la vision par ordinateur, les voitures autonomes, l'intelligence artificielle, la détection de problèmes dans des signaux cardiaques, ou encore la compréhension de l'activité cérébrale.

Perspectives

Chacune des trois parties décrites précédemment peut être étendue de nombreuses manières, décrites ci-dessous dans les sections correspondantes :

Inférence de graphes à partir de signaux

Nous avons introduit plusieurs méthodes afin d'inférer un graphe à partir de signaux stationnaires. Si toutes ces méthodes ont en commun de forcer les vecteurs propres de la matrice de covariance, elles diffèrent par la stratégie de sélection des valeurs propres à y associer. De nombreuses autres méthodes pour opérer cette sélection pourraient être développées, afin de reconstruire par exemple une matrice binaire.

Une autre extension possible serait de trouver une caractérisation des contraintes utiles dans la définition de l'ensemble convexe. En effet, de nombreuses contraintes sont redondantes, et leur suppression permettrait d'accélérer la recherche d'un point dans l'ensemble convexe, donc l'inférence d'un graphe.

Dans nos expérimentations sur des données réelles, nous avons choisi de considérer la matrice de covariance empirique comme estimateur de la matrice de covariance. D'autres matrices pourraient être utilisées à la place, offrant possiblement de meilleures propriétés de convergence de leurs vecteurs propres vers ceux de la matrice de covariance, à mesure que le nombre d'observations augmente.

Enfin, la distance à l'ensemble convexe utilisée par notre stratégie de régularisation fournit une méthode permettant de choisir parmi un ensemble de graphes donnés celui le plus adapté à un ensemble de signaux stationnaires. Complémenter ces expériences en considérant des versions bruitées des graphes est une extension intéressante.

Identification de translations sur graphe

Nos travaux sur l'identification de translations sur les graphes ont aussi de nombreuses extensions possibles. Tout d'abord, la NP-complétude du problème implique la possibilité/nécessité de définir de nouvelles heuristiques pour approximer les translations. Dans ce manuscrit, nous avons choisi de sacrifier en partie la conservation des voisinages, mais il serait possible de faire de même avec la restriction aux arêtes par exemple.

Aussi, nous avons observé qu'identifier des translations dans le cas de graphes très irréguliers est un problème difficile. Approximer une telle topologie par une structure plus régulière, sur laquelle inférer les translations, est une piste très prometteuse. Une

solution à ce problème se trouve peut-être dans les propriétés des matrices circulantes, comme expliqué en Section 5.2.5.

Extension des réseaux de neurones convolutifs à des domaines irréguliers

Enfin, si nos travaux ont démontré l'intérêt de l'identification de translations pour la création de réseaux de neurones convolutifs adaptés aux données, de nombreuses extensions sont encore possibles. En effet, les réseaux actuels les plus performants utilisent des mécanismes plus complexes, tels que du *max-pooling* ou des *strides*, qu'il n'est pas naturel de définir quand le support de l'information est irrégulier. Définir de tels opérateurs est une continuité de nos travaux qui améliorera très certainement les performances de classification sur des bases de données de tests, comme sur des cas d'application réels.

De même, nous avons considéré dans nos expériences une architecture de réseau très simple, constituée de quelques couches convolutives seulement, suffisant à illustrer nos résultats. Varier la taille des noyaux, et trouver les meilleurs paramètres définissant le réseau convolutif devrait permettre à nouveau un gain en performances.

Abstract

Context

Graph signal processing

Few years ago, *graph signal processing* emerged from the following observation: a signal defined over time can be represented by a vector of real numbers, of which each entry corresponds to the signal amplitude at a given time. This time, if periodical, can be modeled by a graph taking the form of a ring (see Figure 4), where each vertex represents a sampling instant, and where an edge exists between vertices that represent adjacent instants in time.

Using this formalism, we manipulate two types of objects: a graph modeling a support for signals, and a signal on this support. Graph signal processing is the field that consists in analyzing such signals, observed on graphs with various topologies. Among examples of signals on graphs, images are defined on pixel grids, or observations of electro-encephalographic signals evolve on a graph modeling the brain or the skull.

The interest of using the formalism of graph signal processing is that it allows to take into account the structure on which a signal evolves, which provides more information than considering the signals alone. In particular, one of the main results of the field is an analogy between the eigenvectors of a matrix representing the graph, the *Laplacian*, and the Fourier modes. As a matter of fact, if we consider a ring graph, compute its eigenvectors soundly, and sort them in increasing order of the corresponding eigenvalues, we remark that these eigenvectors are a cosine, then a sine, then a cosine of higher frequency, etc. Figure 5 offers a graphical representation of these eigenvectors.

Therefore, we observe that the eigenvectors of the graph Laplacian are analogous to the Fourier modes, and its eigenvalues to the frequencies. Projecting a signal on a graph into the basis formed by the eigenvectors of the Laplacian thus corresponds to performing the Fourier transform of this signal. It is then possible for instance to reduce the signal entries associated with the highest eigenvalues, which has for effect to smoothen the signal in the manner of a low-pass filter in classical signal processing. Similarly, it is possible to define numerous other operators to process signals on graphs, such as modulation, convolution, translation, etc.

Beyond this analogy with the graph ring, the framework of graph signal processing can be used for any kind of graph, for which the previously introduced notions are perfectly defined. As an example, Figure 19 illustrates the remark on smoothing a signal with a low-pass filter on a random graph. Graph signal processing thus allows the study of signals evolving on complex structures, bringing new perspectives in terms of understanding these signals, and allowing more concrete tasks such as their classification.

Which graph to use?

As indicated previously, graph signal processing allows the study of signals thanks to the knowledge of the structure on which they evolve. In particular, the eigenvectors of the Laplacian matrix define a Fourier transform that is adapted to the signals, from which numerous other operators are defined.

However, if it is easy to imagine that images are signals defined on a graph taking the form of a grid, or that time signals evolve on a line graph or a ring, it is more complicated to agree on a particular topology to model the human brain or the support of seismic observations. A first solution to process these signals is to use the classical signal processing tools in place of the graph signal processing ones, because they are not applicable due to the lack of a graph. A second preferable option is to infer a graph from the data, in order to build a structure that enables the study of these signals.

However, the number of possible weighted graphs is infinite. Therefore, it is necessary to make some assumption, either on the signals or on the graph. Among classical assumptions are the sparsity of the graph, its regularity, the binarity of its weights, smoothness of signals on it, etc. In this manuscript, we consider a model of stationary signals, and we propose multiple graph inference strategies that enforce chosen properties on this graph.

How do we define translation of signals on a graph?

Numerous operators to manipulate signals on graphs are defined thanks to the graph Fourier transform, which depends on the graph. In particular, translation of a signal on an irregular, abstract structure is not intuitive, as the notion of distance is not well defined on graphs. The translation operator on graphs has then been defined analogously with a property in classical signal processing. As a matter of fact, translating a signal in the graph domain is equivalent to convolving it with a Dirac impulse located at a chosen time instant. As convolution of signals on graphs is defined as a product in the spectral domain, translation of a signal is defined the same way. Therefore, a signal is not really translated in a particular direction, but relocated on a chosen vertex using a Dirac signal on this vertex.

However, the signal structure in the graph domain is not preserved when translated this way. As a matter of fact, a signal that is quite localized in the graph domain can spread on a large number of vertices once convolved with a Dirac impulse on a vertex. As an example, Figure 41 depicts an image, and its translation using this operator. It shows that the patterns in the image cannot be found after application of the translation.

One of the objectives of this manuscript is to propose an alternate definition for a translation operator that uses the graph domain instead of the spectral domain. Such a definition should preserve the signal structure, thus allowing the conservation of patterns in a signal as it is translated.

How can graph signal processing help in a classification task?

Classifying signals is a classical problem with many applications. For example, when monitoring heart activity, one may want to be able to classify the patient as at risk or not. In the development of autonomous cars, establishing whether a signal provided by the various sensors represents a pedestrian, a car, a sign... is of paramount importance.

Nowadays, the most efficient algorithms to perform classification tasks of images are convolutional neural networks. Such networks use the locality of objects in images in order to detect them. Technically, these networks work thanks to a window of a few pixels that is shifted to every possible location of an image to classify. For this reason, if the translation operator is not defined, due to signals being more complex than images, then such networks cannot be created.

This is precisely where graph signal processing can help. In this manuscript, we propose an extension to convolutional neural networks to irregular domains as follows:

1. From a set of training signals, we infer a graph;
2. Then, we identify translations on this graph;
3. Finally, we use these translations to define a convolutional neural network that is adapted to the data to classify.

Main contributions and results

The objective of this manuscript is to propose an extension to convolutional neural networks to irregular domains, based on the signals, thanks to graph signal processing tools. We proceed in three steps:

Graph inference from signals

For a given set of signals to study, it is frequent that no graph is available that explains the structure on which they evolve. An approach that allows one to use the graph signal processing tools in spite of this limitation consists in inferring a graph from the data. However, if the number of vertices is known, because corresponding to the number of variables of which we observe realizations, defining a set of edges between these vertices implies to make some choices.

In this manuscript, we have chosen to consider a model of stationary signals. Such signals can be synthetically generated from white noise, by diffusion on the graph using a matrix that is compliant with its structure. Such a diffusion process tends to modify the signal entries, thus coloring it with the graph structure, while preserving its stationarity. Diffused signals therefore partially encode information on the graph, that we want to extract in order to find the matrix that was used to diffuse the white noise and, by extension, the graph.

In this manuscript, we have shown the following property: *the eigenvectors of the covariance matrix of the signals are also those of the matrix that was used to diffuse them on the graph.* Perfect characterization of the graph thus reduces to finding a vector of eigenvalues to associate with those eigenvectors. We have distinguished two situations:

1. If the diffusion process is simple, *i.e.*, if each signal is diffused a constant and known number of times, we have shown that it is possible to find the missing eigenvalues. Their absolute values can be obtained by computing a particular square root of the eigenvalues of the covariance matrix, and the missing signs can be found by solving an optimization problem;
2. In the more general case of any diffusion process, no information on the eigenvalues is available.

In the second case, it is therefore necessary to make some hypotheses on the matrix to infer. Considering diffusion processes, the matrices we are interested in contain positive entries only, as they represent a signal quantity that is propagated at each step. Thanks to the eigenvectors being known, we can define a convex set of elements, corresponding to vectors of eigenvalues that are admissible according to this criterion. An example of such a convex set is given in Figure 33.

The search for eigenvalues to associate with the eigenvectors of the covariance matrix thus depends on a choice, corresponding to some assumptions on the graph. We have proposed three approaches to choose an element from this convex set:

1. Infer a simple matrix, *i.e.*, with an empty diagonal. This corresponds to a process that maximizes the diffusion of signals;
2. Infer a sparse graph, *i.e.*, with few edges;
3. Use an inference method from the literature, and adapt it to have it match a stationarity assumption by selecting the closest element in the convex set.

We have applied these solutions to synthetic data and to a real-life dataset of weather observations in Brittany, France. Moreover, the third strategy offers potential applications to the problem of choosing, from a set of given graphs, which is the most adapted to a set of stationary signals. Finally, we have observed during the creation of convolutional neural networks from data that regularization of an inference solution with a smoothness prior on signals allows acquisition of a graph that offers the best performance in a classification task.

Identification of translations on a graph

Existing definitions for the translation operator on graph do not preserve the structure of the signal in the graph domain. As a matter of fact, these operators are defined using the spectral domain associated with the eigenvectors of the Laplacian matrix. In its simplest definition, translation of a signal is performed by a convolution with a Dirac signal located on a vertex, rather than by a succession of information transfers from a vertex to another. Other translation operators exist, but similar problems are observed.

In this manuscript, we have proposed a novel definition for the translation operator on graphs that is analogous to the notion of translation in Euclidean spaces. Our translations consist in orienting a subset of the edges of the graph, and can be summed up with three properties:

1. A translation is an injective function from the vertices to the vertices;
2. Every vertex can only be sent to one of its neighbors;
3. The neighbors of a vertex must be sent to the neighbors of the destination of this same vertex, *i.e.*, neighborhoods should be preserved.

These simple properties allow the conservation of the shape of a signal defined on the graph as it is translated. However, such translations only exist on very regular graphs. We have thus adapted these criteria to allow the possibility for a signal entry on a vertex to be *lost* during translation. This models problems such as border effects when translating an image on a non-toric grid graph.

Considering such a grid graph, we have shown that translations that minimize the information loss are those that send every vertex in one of the cardinal directions, as depicted in Figure 48. This corresponds exactly to the classical Euclidean translations

on domains where such a notion of direction exists. The criteria that define our translations do not use this metric space, which makes them applicable to any kind of graph.

However, identifying the best translations on a graph (considering the loss of the translations) is not an easy task. As a matter of fact, we have shown that their identification is an NP-complete problem. This therefore requires the definition of an approximate strategy to obtain pseudo-translations, which partly respects the three criteria above. In details, we have chosen to partly sacrifice the neighborhood preservation property. To do so, we have written our problem in the form of an optimization problem. This way, we have been able to identify interesting pseudo-translations on graphs following a random model, as presented in Figure 50.

Extension of convolutional neural networks to irregular domains

Convolutional neural networks have proven to outperform other methods in tasks such as image classification. Such networks define the connections between their layers using a convolutional kernel, *i.e.*, a window of a few pixels centered on a particular location of the image. This kernel is shifted to every point in the image, and edges are created between the neurons that are associated with the pixels covered by the kernel, and the neuron corresponding to its center. The translation operator thus allows to shift a local point of view on the image, thus to detect objects in it, as such objects are generally defined on neighboring pixels.

As an analogous notion of translation was not defined in the graph domain for irregular topologies, it was not possible to simply extend such networks for signals evolving on complex graphs. Identification of translations as we have proposed has allowed us to do this, as we could translate a convolutional kernel to various places of a graph, thus enabling the creation of a convolutional neural network for the processing of signals on graphs. Combined with our work on graph inference, this technique allows the creation of convolutional networks that are adapted to the signals they want to classify.

We have illustrated our results on the CIFAR-10 dataset of images belonging to ten different classes. Without using the knowledge that the signals are images, graph inference has allowed us to find a graph that is close to a grid of pixels, and therefore to obtain performance that are comparable with convolutional neural network using this information. Inferring the graph, then identifying translations on it, allowed us to obtain a correct classification rate 13.11 points higher than state of the art methods that also do not use this information, while using a simple network only, which could easily be improved through a grid search of its parameters.

Our method is a real extension to convolutional neural networks, which can be seen as a particular case of our approach in which a grid graph is considered. The method we propose in this manuscript offers numerous possibilities of applications, in fields including computer vision, autonomous cars, artificial intelligence, heart monitoring or understanding the human brain.

Perspectives

Each of the three parts of our approach can be extended in many ways, described as follows in the corresponding sections:

Graph inference from signals

We have introduced multiple methods to infer a graph from stationary signals. While all these methods have in common to enforce the eigenvectors of the covariance matrix, they differ by the selection strategy for the eigenvalues. Numerous other methods to make this selection could be developed, in order to infer a binary matrix for instance.

Another possible extension would be to characterize which constraints are useful in the definition of the convex set. As a matter of fact, numerous constraints are redundant, and their suppression could accelerate the search for an element in the convex set, thus the graph inference process.

In our experiments on real-life data, we have chosen to consider the sample covariance matrix as an estimate for the covariance matrix. Other matrices could be used in place, thus possibly offering better convergence properties of their eigenvectors to those of the covariance matrix, as the number of observations increases.

Finally, the distance to the convex set we use in our regularization strategy gives us a method for choosing among a set of graphs which one is the most adapted to stationary signals. Complementing these experiments by considering noisy versions of the graphs is an interesting direction for future work.

Identification of translations on a graph

Our work on identifying the translations on graphs also has numerous possible extensions. First, NP-completeness of the problem implies the possibility/necessity to define new heuristics in order to approximate the translations. In this manuscript, we have chosen to partly sacrifice the neighborhood preservation property, but the same could be done for example with the restriction to the edges.

Also, we have observed that identifying translations in the case of very irregular graphs is a difficult problem. Approximating such a topology with a more regular structure on which to infer the translation is a very promising direction. A solution to this problem probably lies in the properties of circulant matrices, as explained in Section 5.2.5.

Extension of convolutional neural networks to irregular domains

Finally, if our work has demonstrated the interest of translations identification for the creation of convolutional neural networks that are adapted to the data, numerous extensions are still possible. As a matter of fact, the best networks use more complex mechanisms, such as *max-pooling* or *strides*, which are not natural to define when the information support is not regular. Defining such operators is a continuity of our work that will certainly improve the classification performance on datasets, as well as on real-life application cases.

Also, we have considered in our experiments a very simple network topology, consisting of a few convolutional layers only, that was sufficient to illustrate our results. Varying the kernels sizes, and finding the best parameters defining the convolutional neural network should again provide a gain in performance.

Chapter 1

Introduction

Contents

1.1 Context

The way we perceive and understand the world surrounding us is mostly a matter of signals and sensors. Information is created by various objects, and propagated in the world by some emitters in the form of signals. It is then perceived by some receptors attached to other objects which can draw their own model of the initial information.

When it comes to human perception, we have access to a certain number of such sensors, traditionally denoted by *senses*. According to Aristotle [1], hearing lets us perceive the sounds, sight allows the physical representation of the world, and touch, taste and smell give information on the nature of the objects. All these senses allow the extraction of information from the surrounding world, which is then analyzed by our brain to perceive and understand our environment. Still, we are only able to perceive what we can sense, making our model of the world an approximate of what it really is.

Figure 1 – *The Five Senses* by Hans Makart. Series of five paintings from 1972 to 1979.

Perception of our world has passionated philosophers, scientists and artists across the ages. Countless theories were developed to try to infer a general model of information. Among these, one in particular had a very strong impact on our understanding of the world: Fourier analysis. During the XIX$^{\text{th}}$century, by observing heat propagation, Fourier drew a generic model of representation of signals as series of sines, allowing the representation of information as a composition of various frequencies [Fou22]. This theory allowed numerous discoveries in fields such as electricity, vibrations, acoustics or observation of the greenhouse effect [Esc+01], and eventually led to the the emergence of a new field of mathematics and physics named *signal processing*.

1. Scientific results suggest that the number of human senses reaches approximately twenty [Dra05].

From there, signal processing developed to allow the rise of new technologies such as modern telecommunications, music or video processing, or computer networks such as the Internet. However, until a few years ago, signal processing tools were only designed to process information defined on regular domains, such as time or Euclidean spaces. In numerous practical domains, signals evolve on more complex topologies, such as information propagating in the human brain. Analyzing such signals remains a challenge, and solutions require an adaptation of the classical signal processing tools.

Graph signal processing then arose as a possible solution to this challenge, proposing an extension of Fourier analysis to signals defined on any particular discrete topology. To understand well the main idea from which this domain developed, let us study a simple example, in which we want to analyze the periodic time signal given in Figure 2:

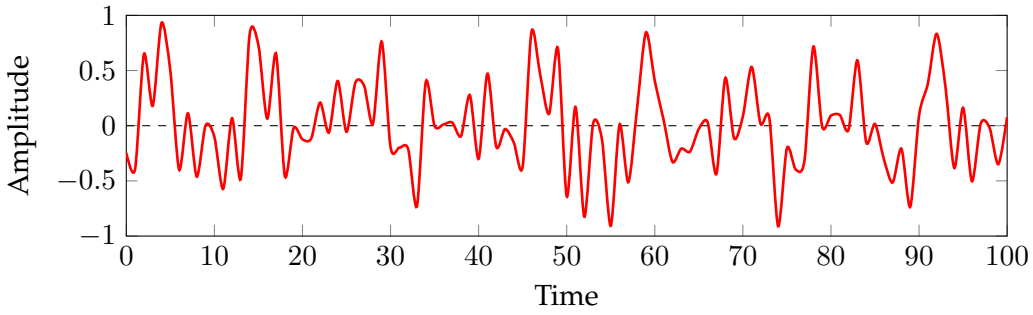


Figure 2 – Example of a periodical signal we want to analyze.

While the theory of Fourier applies to continuous signals, discretizing the signals can be very convenient to process them with computers, or send them through a network as a finite series of bytes. Let us sample the signal in Figure 2 as depicted in Figure 3:

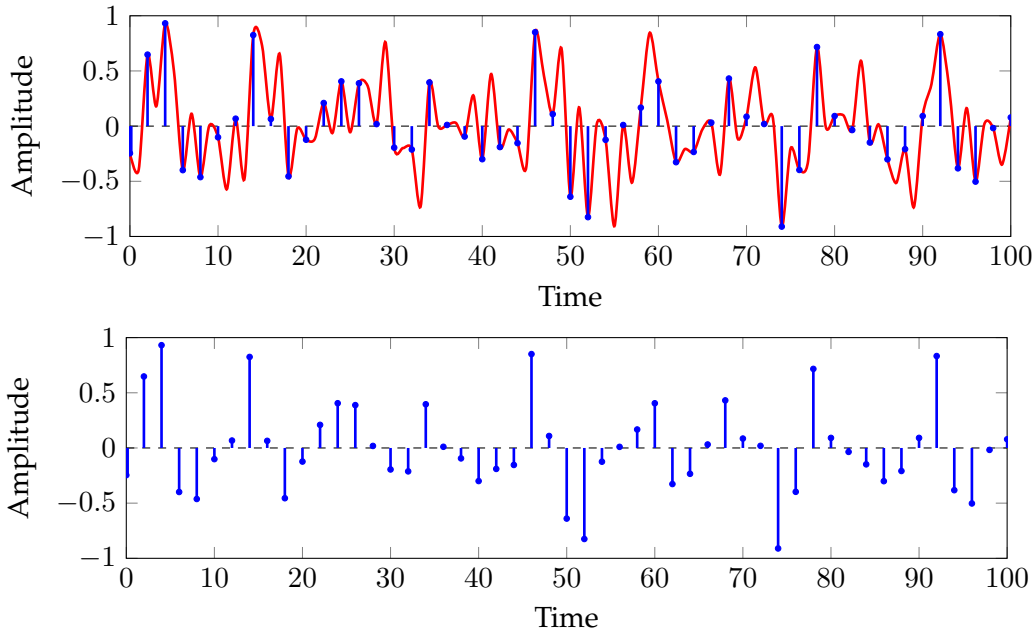


Figure 3 – A possible sampling of the example signal in Figure 2. By keeping the signal entries corresponding to regular moments in time, we obtain a discrete approximate of the continuous signal.

After sampling, the signal becomes discrete, and can be represented by a vector in $\mathbb{R}^N$, where N is the number of samples that approximate the signal. Now, let us represent the periodic time by a graph such that every time instant is represented by a vertex, and is linked to the previous and next instants in time. Figure 4 depicts such a graph, on which the discretized signal can be seen as a single observation for every vertex:

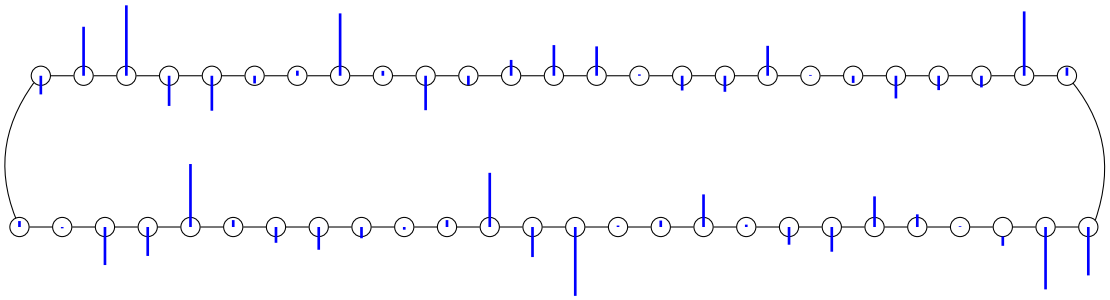
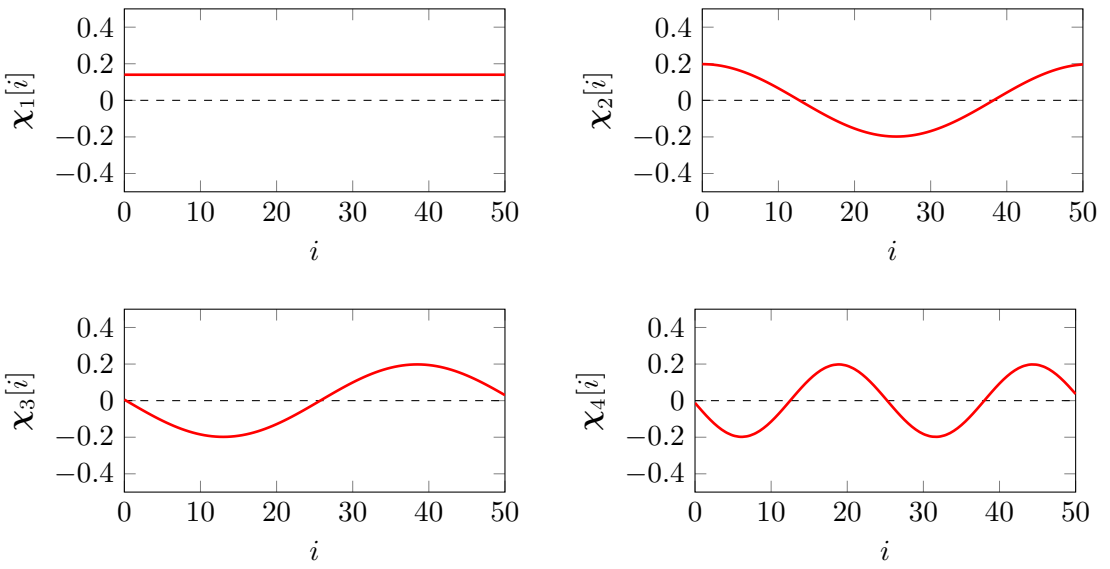


Figure 4 – Representation of the discretized signal in Figure 3 as a signal on a graph. Periodic time is represented by a graph of N vertices, and vertices corresponding to adjacent instants in time are linked by an edge. Note that location of the vertices in the picture has no importance, and only adjacency matters.

Note that, by taking this point of view, we distinguish the signal from the domain on which it evolves. Thus, we manipulate two distinct objects: a graph — modeling the support of information — and some signals evolving on the graph.

Without entering too much into details, graph structures as in Figure 4 can be represented by their adjacency matrix $\mathbf{A}$, *i.e.*, by a two-dimensional array in which the entry at the intersection of row i and column j is equal to 1 if the i^{th} vertex is linked to the j^{th} one by an edge in the graph, and 0 otherwise. From this particular matrix, one can compute the *Laplacian matrix* of the graph, which is a real and symmetric matrix, thus diagonalizable in an orthonormal basis. Let us consider the eigenvectors $\{\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_N\}$ of the Laplacian matrix associated with the graph in Figure 4. Figure 5 depicts the first few of these eigenvectors:



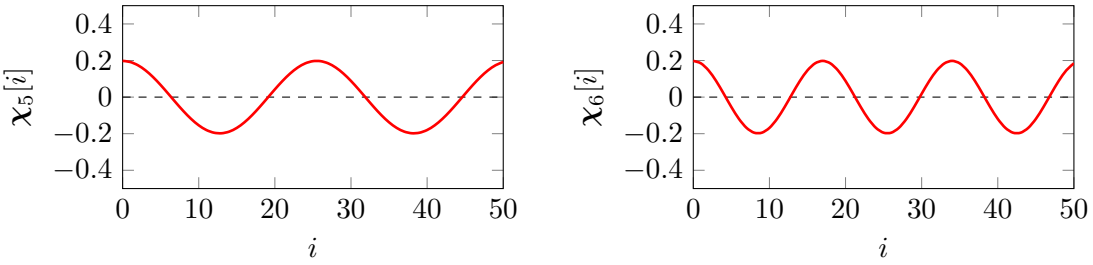
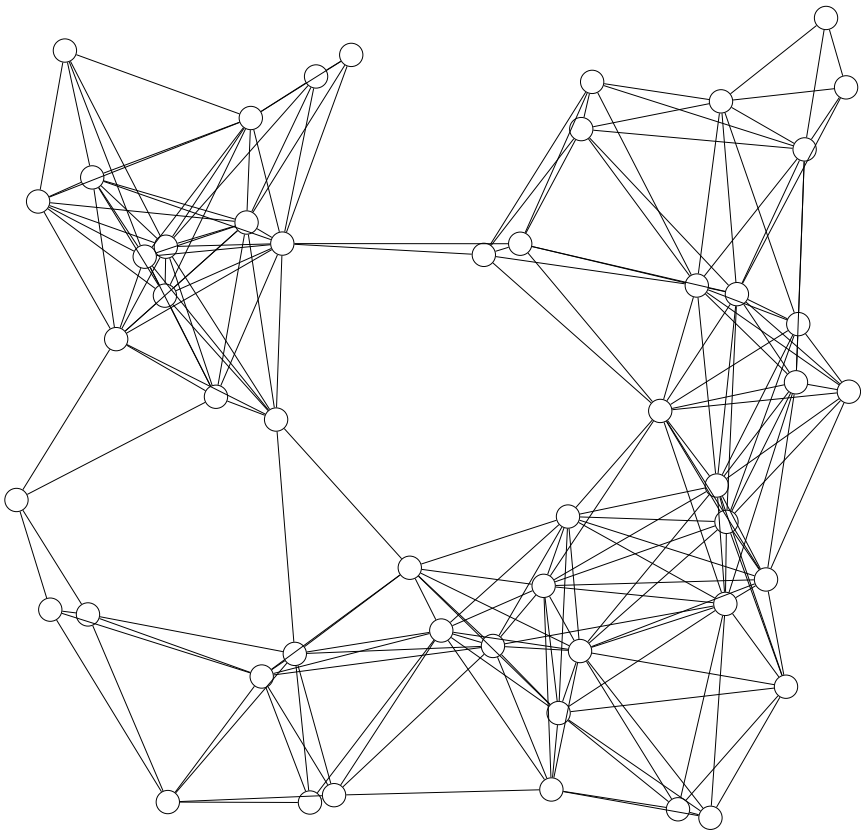


Figure 5 – First eigenvectors of the Laplacian of the graph in Figure 4.

Interestingly, the eigenvectors of the Laplacian matrix are exactly the sines that decompose time signals in classical Fourier analysis. Therefore, projecting a time signal to the basis formed by the eigenvectors of Laplacian matrix corresponds to projecting it to a basis of sines. This motivating example provides a comprehensive link between the time domain in which signals are defined, and the frequency domain in which they can be decomposed.

The entire field of graph signal processing developed from this observed analogy between the classical Fourier basis and the eigenvectors of the Laplacian matrix. By representing the support of signals under study by a graph, one can determine an adapted basis of frequencies in which the signals can be represented.

As an example, Figure 6 depicts a complex graph, representing a domain on which signals we want to study are evolving. This graph can represent a network of sensors, measuring for instance seismic activity. Such seismic signals can then be decomposed into frequencies through the eigenvectors of the Laplacian matrix, and can be analyzed as if they were simple time signals:



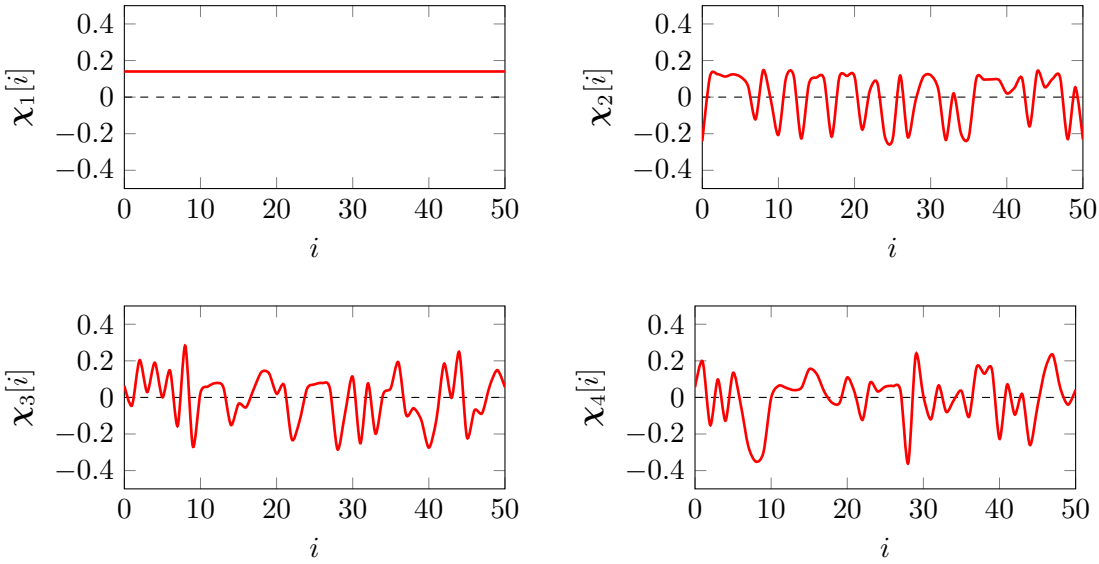


Figure 6 – Example of a complex topology on which signals we want to study are observed (top). The first few eigenvectors $\{\chi_1, \ldots, \chi_N\}$ of the associated Lapacian matrix are depicted (bottom). They represent the equivalent for this graph to the Fourier sines depicted in Figure 4.

Using the paradigm of graph signal processing, it is therefore possible to study signals evolving on complex domains. Researchers have successfully developed tools for graph signal processing, inspired from those existing in classical Fourier analysis. Operations such as filtering of signals, convolution or wavelets are now possible, and have already proven to be efficient in image denoising or analysis of heat diffusion [Shu+13].

1.2 Problems: F.A.Q.

1.2.1 Which graph corresponds to my signals?

As stated before, graph signal processing is a powerful generalization of classical Fourier analysis, that allows the study of signals evolving on very complex structures, modeled by graphs. This opens the way to applications such as comprehension of signals evolving in the brain, or analysis of signals observed by sensor networks, which was not possible before. However, while it is quite intuitive to represent the time by a graph as in Figure 4, it is harder to imagine which graph correctly represents things such as the human brain.

In fact, in most situations in which we want to study signals, the underlying graph topology is not available, leading to the impossibility to determine the adapted basis providing a frequency representation of signals. This is a real problem, since without the eigenvectors of the Laplacian matrix, it is just impossible to study signals using the graph signal processing framework.

A possible solution is graph inference from signals. Obviously, signals defined on the vertices of a particular domain have a certain connection with this domain. For example, when considering time signals, it is in general the case that two adjacent instants in time — *i.e.*, two vertices in the graph representing time — share similar signal entries.

Therefore, signals partially encode information on the underlying topology on which they evolve, and extracting this information can help finding the graph.

Graph inference from signals is one of the problems we study in this document. More particularly, we consider the specific model of stationary signals, and propose strategies to infer an adapted graph to process them.

1.2.2 I know how to move signals in time, but how does it work on a graph?

Shifting signals in time is a cornerstone of classical signal processing. This allows operations such as convolution with a filter, which in turn contribute to applications such as signal denoising or detection of anomalies.

When considering time signals, translation is quite easy to understand. If a signal needs to be advanced by τ seconds, one can just replace every signal entry at time t by the signal entry at time $t - \tau$. Similarly, to translate images, one can just move every pixel in a 2D direction.

In the general context of graphs, translating a signal from a vertex to another is not as natural. This is due to the absence of an underlying Euclidean space as for time signals or images. Efficient solutions to translate filters on graphs have been developed, allowing the localization of such signals to various places of the graph. However, these classical solutions do not apply to signals that have a certain organization, like images for instance in which a pixel has a link with its surroundings. As a matter of fact, these solutions consider translation as a particular case of convolution, which has the effect to modify the signal entries while translating it.

To be able to retain the overall organization of the signal, another possible solution we propose is to infer translations from the graph. When considering the case of the graph representing time as in Figure 4, a natural translation on such graph consists of an orientation of the edges as in Figure 7:

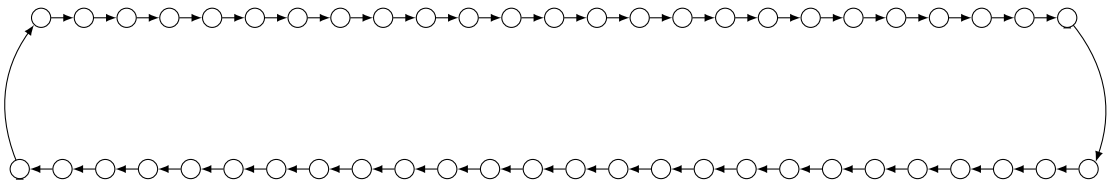


Figure 7 – Natural translation on a graph representing time. Advancing a signal in time using this graph can be done by replacing every signal entry by the one on the previous vertex, following the edges orientations.

The same approach can be taken in the case of any graph, by considering a translation on a graph as an orientation of a carefully chosen subset of the edges of the graph.

1.2.3 Can graph signal processing help me classify signals?

Classification of signals consists in saying, for a given signal, whether it belongs or not to a certain class. For example, if the signal to classify is a photo of a dog, an expected classification for this signal would be the class *dog*.

There are numerous techniques to classify signals, among which the now well-known neural networks, on which *deep learning* is based. A particular type of neural network is of interest for us. *Convolutional neural networks* are very efficient to detect objects in images, and have the very interesting property to be *invariant by translation* of these objects, which through a slight misuse of language indicates that an object location in the image does not alter the performance of the network.

If we are able to identify translations on underlying graphs, then we can adapt these networks to locate some signal patterns on very complex topologies. Therefore, convolutional neural networks can then be used to classify signals on the graph, even if observed on an irregular domain.

1.3 Outline

In this document, we propose solutions to the problems introduced in Section 1.2. We start from a set of signals that we assume to evolve on an unknown graph. From these signals, we infer a graph using some priors. Then, we propose an approach to infer translations on this graph, and use these translations to define an adapted convolutional neural network. This network is then trained to classify the signals from which the whole process started.

This Ph.D. dissertation is organized as follows:

- First, Chapter 2 introduces all the notions that are necessary for a complete understanding of our work. This includes, among others, elements of graph theory, signal processing, and graph signal processing;
- Then, Chapter 3 studies the problem of graph inference from signals, as presented in Section 1.2.1. More specifically, we present multiple methods, and propose strategies to infer a graph from stationary signals;
- Chapter 4 then presents the notion of translations on graphs, and provides an in-depth study of the problem introduced in Section 1.2.2. We show that intuitive translations of time signals or images can be modeled by an orientation of some edges of a particular graph, and extend our observations to any graph, allowing the definition of translations that preserve the signal shape when translating it;
- Once a graph has been inferred from signals, and translations have been found on the graph, Chapter 5 proposes an application of our work to signals classification. In particular, we consider a dataset of images — without using the information that such signals are images — and show that inferring a graph and translations allows us to improve the classification rate of these images by 13.11 points compared to state of the art methods;
- Finally, Chapter 6 summarizes the various contributions of this document, and concludes this Ph.D. dissertation.

Chapter 2

Signal processing on graphs

Contents

This chapter introduces numerous notions that are necessary for a complete understanding of the contributions presented in this manuscript. The context of this work being the field of graph signal processing, this chapter is divided into three parts, corresponding to the three words naming the domain: Section 2.1 provides elements of graph theory, Section 2.2 gives notions of classical signal processing, and finally Section 2.3 presents the field of graph signal processing.

2.1 Elements of graph theory

2.1.1 What is graph theory?

Graph theory is a subfield of discrete mathematics and computer science, that consists in the study of *graphs*, which are mathematical objects that can be used to model problems. The first contribution to graph theory is an article by Euler dating from 1736, in which he studies the following problem:

Problem 1: The seven bridges of Königsberg

There are seven bridges in the city of Königsberg. Is there a walk in this city that crosses every bridge once and only once?

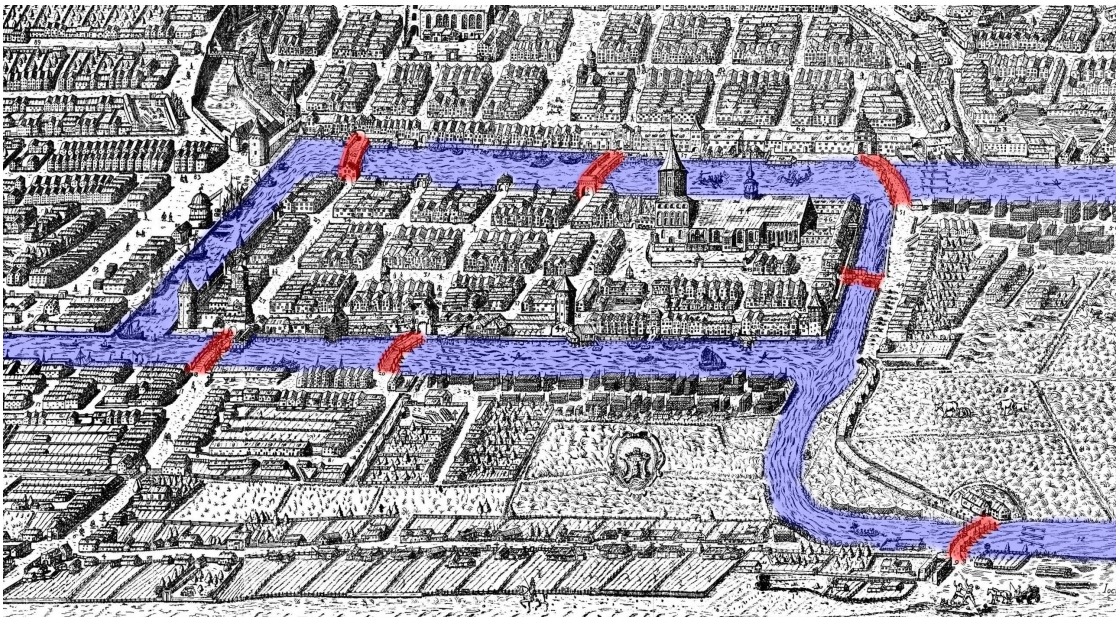


Figure 8 – Map of the city of Königsberg. The Pregel River crossing the city is highlighted in blue, and the bridges are highlighted in red.

Using a formalism that led to the emergence of graph theory, Euler showed that Problem 1 cannot be solved. A first step in his reasoning consisted in simplification of the problem by considering only the elements that have importance to model the problem:

- The four parts of the city of Königsberg that are separated by the Pregel River;
- The bridges that link these parts.

This formalism of modeling a problem by two sets — a set of elements of interest, and a set of relations between them — forms the basis of graph theory. More precisely, a *graph* is defined as follows:

Definition 1: Graph

A ***graph*** is a tuple $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where:

- $\mathcal{V}$ is a set of ***vertices***;
- $\mathcal{E}$ is a multiset of pairs of vertices, called ***edges***.

Remark 1

Throughout this document, we call ***pair*** a set of two elements, given in no particular order, noted $\{v_1, v_2\}$. Contrary to pairs, we term ***couple*** a set of two elements, in which order matters, noted (v_1, v_2).

The number of vertices in the graph is called its ***order***, and the number of edges is called its ***size***. Additionally, for a given vertex $v \in \mathcal{V}$, the set of vertices it is linked to by an edge in $\mathcal{E}$ is called its ***neighborhood***, noted $\mathcal{N}(v)$.

This model allows a simplified visual representation of the problem. Graphical representations of graphs have already been provided in Figure 4 or Figure 6. When considering Problem 1, the graph associated with the city of Königsberg is given in Figure 9:

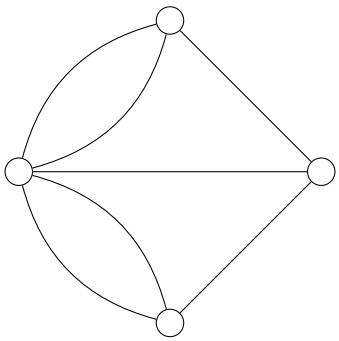
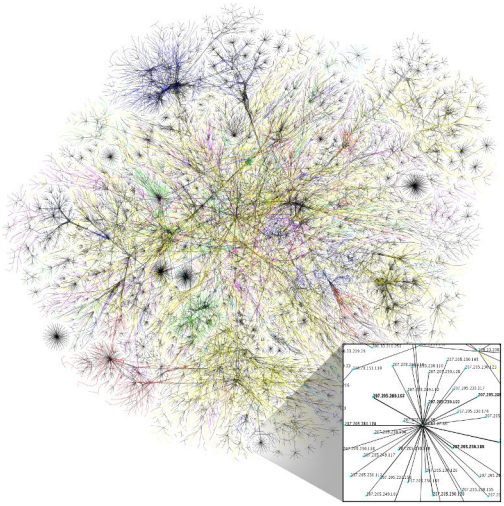
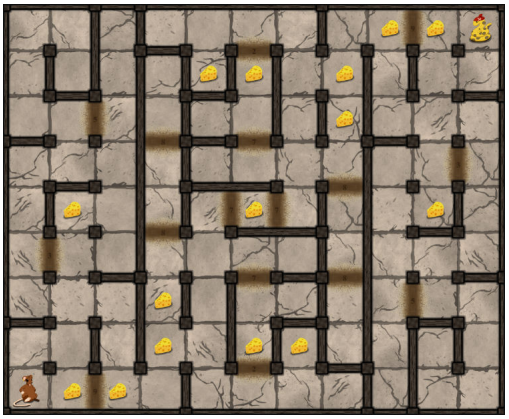


Figure 9 – Graph associated with the city of Königsberg. Vertices represent the locations of interest, *i.e.*, the four parts of the city. Edges link two vertices if there is a bridge allowing to go from one to another.

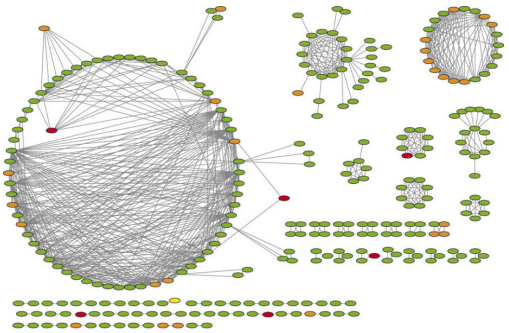
Graph theory developed around this modelization, providing an abstract model of problems on which numerous results have been found. Algorithms on graphs have then been applied to countless domains, allowing for instance development of GPS systems, resolution of games, networking, optimization of trajectories for production elements, social media... Figure 10 shows a few examples in which graph theory has been used:



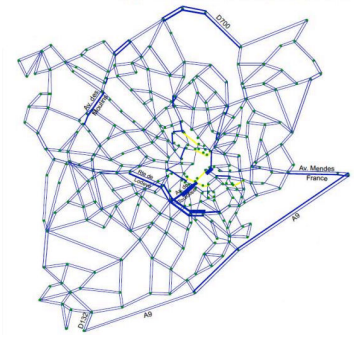
(a) Partial graph of the Internet, on the 15th of January, 2005 [Opt]. Every vertex represents a terminal, and edges model connections between these terminals.



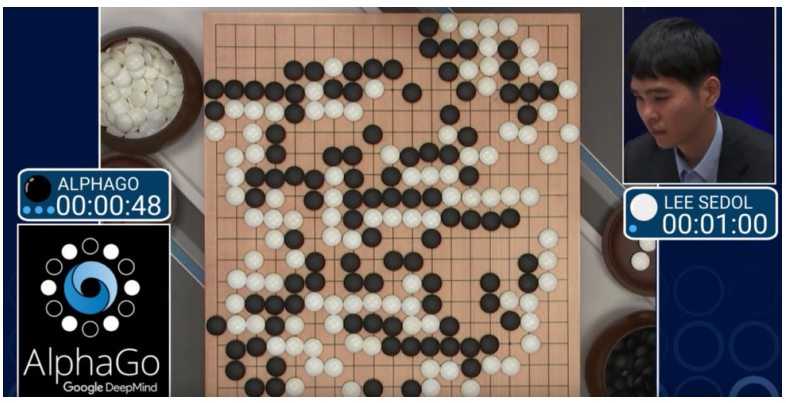
(b) PyRat, a serious game to learn programming, in which two characters controlled by programs compete to grab more pieces of cheese than the opponent [PK17]. Every vertex represents a location in the maze, and edges model accessibility between these locations.



(c) In genetics, upstream similarity network is a graph used to analyze the possible regulatory role of genes [RMC08]. Every vertex represents a gene, and edges model similarity between these genes.



(d) Road graph of the city of Montpellier, used to analyze traffic blockages [AL07]. Every vertex represents a crossroad, and edges model streets going from one crossroad to another.



(e) AlphaGo had a strong impact on society, when researchers from Google DeepMind managed to defeat the professional players Fan Hui (in 2015), Lee Sedol (in 2016) and Ke Jie (in 2017) at the game of Go [Sil+16]. AlphaGo makes extensive use of graph theory, in particular through the use of deep learning and Monte-Carlo tree searches.

Figure 10 – Examples of applications of graph theory. Abstracting details to model them in the framework of graph theory allows the application of similar algorithms in all these cases.

2.1.2 Extensions of the graph model and operations on graphs

A graph in Definition 1 provides a minimal model for problems, which can be enriched depending on the information to model. In this section, we first present a few classical extensions and properties of graphs as defined in Definition 1. Then, we introduce the notion of paths on graphs that we will need in this manuscript.

Classical extensions and properties of graphs

A first classical extension of graphs are *weighted graphs*.

Let us consider for example the modelization of the roads of a country by a graph, in which vertices represent cities, and edges link two cities if a road exists between them. An important information that is not encoded in Definition 1 is the distance between these cities. Weighted graphs are given a weighting function, that associates a certain quantity with the edges:

Definition 2: Weighted graph

A ***weighted graph*** is a tuple $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, f \rangle$, where:

- $\mathcal{V}$ is a set of vertices;
- $\mathcal{E}$ is a multiset of pairs of vertices, called edges;
- $f : \left\{ \begin{array}{ccc} \mathcal{E} & \to & \mathcal{W} \\ \{v_1, v_2\} & \mapsto & w \end{array} \right.$ is a function associating a weight with every edge.

Frequently, $\mathbb{Z}$, $\mathbb{R}$ or $\mathbb{C}$ are used for the codomain $\mathcal{W}$ of f, depending on the problem to model. Note that unweighted graphs can be considered as particular cases of weighted graphs, in which the weight function is disregarded.

Graphically, the edge weight is indicated along the corresponding edge, and is often dropped when it is equal to 1. An example of a weighted graph is given in Figure 11:

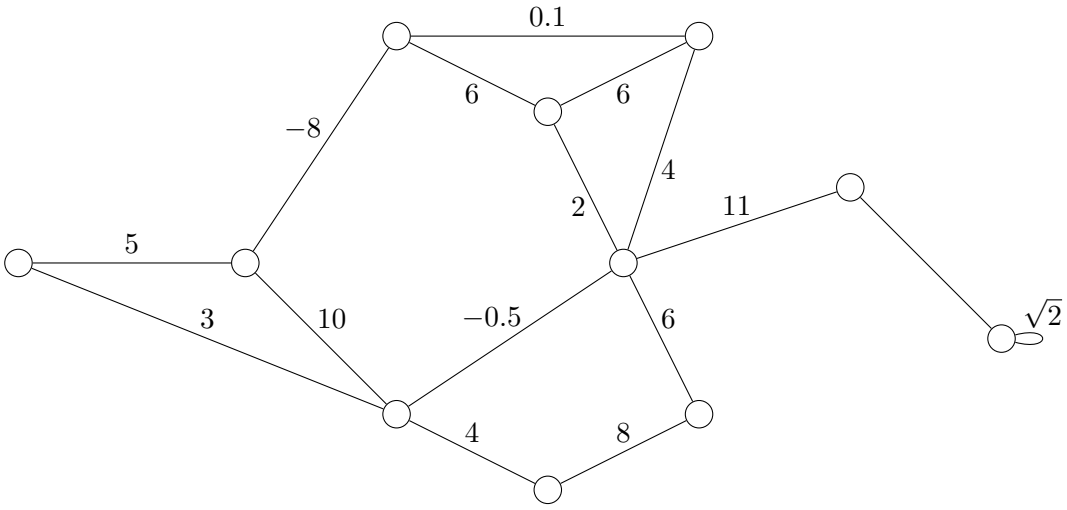


Figure 11 – Example of a weighted graph.

A second classical extension of graphs are *directed graphs*. Such models encode the information that access to a vertex from another one can be unidirectional, which can be useful for instance to represent one direction roads:

Definition 3: Directed graph

A ***directed graph***, or ***digraph***, is a tuple $\overrightarrow{\mathcal{G}} = \langle \mathcal{V}, \overrightarrow{\mathcal{E}} \rangle$, where:

- $\mathcal{V}$ is a set of vertices;
- $\overrightarrow{\mathcal{E}}$ is a multiset of couples of vertices, called ***directed edges***, or ***diedges***.

Note that, contrary to Definition 1, the order in which vertices are given in the diedges has a lot of importance. For this reason, elements of $\overrightarrow{\mathcal{E}}$ are couples (v_1, v_2), meaning that there is a relation from v_1 to v_2. Again, note that undirected graphs can be considered as particular cases of digraphs, in which $(v_1, v_2) \in \overrightarrow{\mathcal{E}}$ if and only if $(v_2, v_1) \in \overrightarrow{\mathcal{E}}$.

Graphically, diedges are depicted with an arrow on one side of the edge, indicating the direction. However, when both $(v_1, v_2) \in \overrightarrow{\mathcal{E}}$ and $(v_2, v_1) \in \overrightarrow{\mathcal{E}}$, the arrows are generally dropped. An example of a directed graph is given in Figure 12:

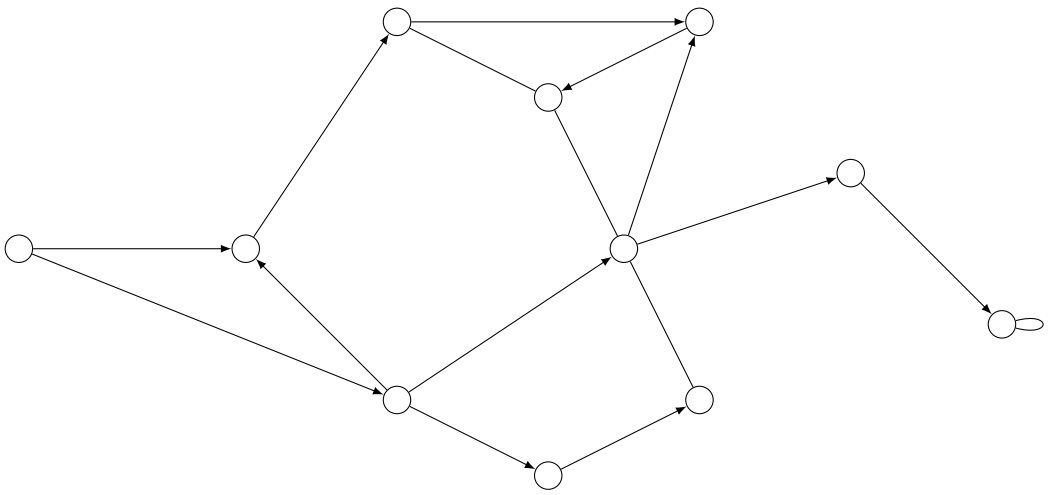


Figure 12 – Example of a directed graph.

Of course, graphs can be both weighted and directed if the problem under study requires so. Numerous additional extensions of the graph model exist, but we are only interested in these two extensions in this manuscript.

To conclude with this section, we introduce a few properties of graphs. A first one is *simplicity* of the graph:

Definition 4: Simple graph

A graph is said to be ***simple*** if no vertex is linked to itself by an edge — *i.e.*, there are no ***self-loops*** — and if no edge has a duplicate in the set of edges.

Remark 2

In the remaining of this document, we will restrict our study to graphs that do not have duplicates in the set of edges. Therefore, we can simplify the definitions so that $\mathcal{E}$ is a set of pairs, and $\overrightarrow{\mathcal{E}}$ is a set of couples.

Another property of interest is *sparsity* of the graph, that measures the number of edges it contains:

Definition 5: Sparse graph

A graph is said to be ***sparse*** if the number of edges it contains is low relatively to the number of edges it could contain. In the case of an undirected graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, this is measured by the following quantity:

$$\frac{2|\mathcal{E}|}{|\mathcal{V}|(|\mathcal{V}|-1)}\,, \tag{1}$$

where $|\cdot|$ is the cardinality operator. In the case of a directed graph $\overrightarrow{\mathcal{G}} = \langle \mathcal{V}, \overrightarrow{\mathcal{E}} \rangle$, this is measured by the following quantity:

$$\frac{\left|\overrightarrow{\mathcal{E}}\right|}{|\mathcal{V}|(|\mathcal{V}|-1)}\,. \tag{2}$$

Finally, we call an *induced subgraph* of a graph a subset of its vertices, and the edges that link two vertices in this subset:

Definition 6: Induced subgraph

An ***induced subgraph*** $\mathcal{G}^- = \langle \mathcal{V}^-, \mathcal{E}^- \rangle$ of a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is defined as follows:

- $\mathcal{V}^- \subseteq \mathcal{V}$;
- $\forall \{v_1, v_2\} \in \mathcal{E} : \{v_1, v_2\} \in \mathcal{E}^- \Leftrightarrow v_1 \in \mathcal{V}^- \wedge v_2 \in \mathcal{V}^-$.

Paths on graphs

Once a problem is modeled in the form of a graph, algorithms on graphs can be used to answer some questions about the graph, hence about the problem. Numerous of such questions involve computation of *paths* on the graph. Paths on graphs represent sequences of vertices, and can represent for example a process, or a trajectory, depending on the modeled problem:

Definition 7: Path

A ***path*** on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is a (possibly empty) sequence $p = (v_i)_i, v_i \in \mathcal{V}$ such that:

- $\forall i \in [\![1, |p| - 1]\!] : \{v_i, v_{i+1}\} \in \mathcal{E}$, where $|\cdot|$ is the cardinality operator, indicating the number of vertices composing the path;
- Every vertex of $\mathcal{V}$ appears at most once in p, with the possible exception of the first and the last vertices of p;

- Every unordered pair of two vertices $\{v_i, v_{i+1}\}$ that are consecutive in p appears at most once in p.

In this manuscript, we choose to consider paths as sequences of vertices. Equivalently, a path can be defined as a sequence of edges. We denote the set of all paths in a graph by $\mathcal{P}(\mathcal{G})$. Additionally, we call ***length*** of a path $p \in \mathcal{P}(\mathcal{G})$ the number of edges that compose it, *i.e.*, $|p| - 1$.

Definition 7 tells us that vertices must appear at most once along a path, with the possible exception of the first one, that can be identical to the last one. Paths verifying this latter property are called ***cycles***. We denote the set of cycles in the graph by $\mathcal{C}(\mathcal{G})$.

Using the notion of paths, we can introduce the notion of *connected component* as follows:

Definition 8: Connected component

A ***connected component*** in a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is a subset $\mathcal{V}^- \subseteq \mathcal{V}$ of its vertices such that:

$$\forall v_1, v_2 \in \mathcal{V}^- : \exists (v_1, \dots, v_2) \in \mathcal{P}(\mathcal{G}) \ . \tag{3}$$

A graph that has only one connected component is said to be ***connected***.

Remark 3

In the remaining of this document, we consider connected graphs only.

As for graphs, paths have numerous properties. In particular, the notion of *shortest path* is useful in multiple situations. For example, consider a graph representing roads in a city. Finding a shortest path from a vertex representing a source location to another vertex representing a target location is something of real interest in the development of GPS systems:

Definition 9: Shortest path

A ***shortest path*** from $v_1 \in \mathcal{V}$ to $v_2 \in \mathcal{V}$ is a path $p_1 \in \mathcal{P}(\mathcal{G})$ of first element v_1 and last element v_2 such that no other path $p_2 \in \mathcal{P}(\mathcal{G})$ of first element v_1 and last element v_2 has a strictly smaller length than p_1.

Length of the shortest paths between two vertices v_1 and v_2 is called the ***geodesic distance*** between them, noted $d_{\text{geo}}(v_1, v_2)$.

Algorithms to find shortest paths in graphs exist, depending on some graph properties. Knowing how such algorithms perform is not necessary to understand our work. Still, the interested reader may consult [Cor09] for more details.

Remark 4

In the rest of this manuscript, we are only interested in graphs with positive weights. A shortest path between two vertices can then be found using a *breadth first search* algorithm [Cor09].

When determining translations on graphs later in this manuscript, we will be interested in *Hamiltonian paths*:

Definition 10: Hamiltonian path

A path $p \in \mathcal{P}(\mathcal{G})$ on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is said to be ***Hamiltonian*** if all vertices of $\mathcal{V}$ appear in p.

Finally, let us consider a small adaptation of Definition 7 in which we allow vertices to be repeated in the path (we call this a ***trail***). We say that a trail is ***Eulerian*** if every edge in the graph appears as an unordered pair of two adjacent vertices in the trail. The name of this property comes from Problem 8, since Eulerian trails are the expected solutions of the problem introduced by Euler.

2.1.3 Matrix representation of graphs

Modeling a graph by a matrix

It is often convenient to index vertices from 1 to $N = |\mathcal{V}|$, with $|\cdot|$ being the cardinality operator. We note $[\![C_1, C_2]\!]$ the set of all integers between $C_1 \in \mathbb{Z}$ and $C_2 \in \mathbb{Z}$, both included. Using indexation of vertices, we can write $\mathcal{V} = [\![1, N]\!]$, thus making no distinction between the vertices and their index. Therefore, a graph can be modeled by a two-dimensional array — a matrix — in which the entry at the intersection of row v_1 and column v_2 ($v_1, v_2 \in \mathcal{V}$) indicates if an edge exists between v_1 and v_2.

Remark 5

In order not to make a confusion between variable $v_i \in \mathcal{V} = [\![1, N]\!]$ and the vertex of index i, we choose to represent the latter by $\textcircled{i}$.

The matrix representing the graph structure is called its *adjacency matrix*:

Definition 11: Adjacency matrix of a graph

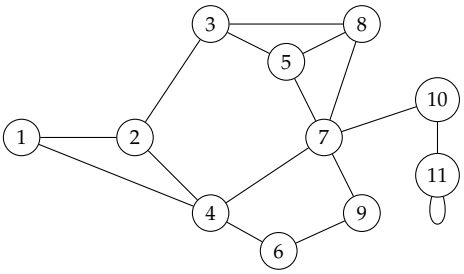
The ***adjacency matrix*** $\mathbf{A}$ of a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ as defined in Definition 1 is an $N \times N$ matrix with:

$$\forall v_1, v_2 \in \mathcal{V} : \mathbf{A}[v_1, v_2] = \begin{cases} 1 & \text{if } \{v_1, v_2\} \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} . \tag{4}$$

In this manuscript, we note $\mathbf{A}[v_1, v_2]$ the entry of matrix $\mathbf{A}$ at the intersection of row v_1 and column v_2. Additionally, we use the notations $\mathbf{A}[v_1, :]$ and $\mathbf{A}[:, v_2]$ to represent the complete ${v_1}^{\text{th}}$ row and ${v_2}^{\text{th}}$ column of $\mathbf{A}$, respectively.

The adjacency matrix of a graph in Definition 1 has the following interesting properties:

- It is ***binary***, *i.e.*, every entry $\mathbf{A}[v_1, v_2]$ is either 0 or 1;
- It is ***symmetric***, *i.e.*, $\forall v_1, v_2 \in \mathcal{V} : \mathbf{A}[v_1, v_2] = \mathbf{A}[v_2, v_1]$;
- If $\mathcal{G}$ is simple, then the diagonal entries — those at the intersection of row v and column v — of $\mathbf{A}$ are all 0. The example graph in Figure 13 is not simple:



$$\begin{array}{c} \\ \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{4} \\ \textcircled{5} \\ \textcircled{6} \\ \textcircled{7} \\ \textcircled{8} \\ \textcircled{9} \\ \textcircled{10} \\ \textcircled{11} \end{array} \begin{array}{c} \begin{array}{ccccccccccc} \textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} & \textcircled{5} & \textcircled{6} & \textcircled{7} & \textcircled{8} & \textcircled{9} & \textcircled{10} & \textcircled{11} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

Figure 13 – Example of a graph (left), and associated adjacency matrix (right).

Extensions of this graph model introduced in Section 2.1.2 can also be represented by matrices. In the case of a weighted graph, we call this matrix *weights matrix*:

Definition 12: Weights matrix

The ***weights matrix*** $\mathbf{W}$ of a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, f \rangle$ is an $N \times N$ matrix with:

$$\forall v_1, v_2 \in \mathcal{V} : \mathbf{W}[v_1, v_2] = \begin{cases} f(\{v_1, v_2\}) & \text{if } \{v_1, v_2\} \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} . \tag{5}$$

Contrary to the adjacency matrix, the weights matrix is in general not binary. However, remember that graphs are particular cases of weighted graphs. For this reason, the adjacency matrix of an unweighted graph is sometimes written $\mathbf{W}$.

Considering the second extension introduced in Section 2.1.2, the adjacency matrix of a directed graph is defined as follows:

Definition 13: Adjacency matrix of a digraph

The ***adjacency matrix*** $\mathbf{A}$ of a digraph $\overrightarrow{\mathcal{G}} = \langle \mathcal{V}, \overrightarrow{\mathcal{E}} \rangle$ is an $N \times N$ matrix with:

$$\forall v_1, v_2 \in \mathcal{V} : \mathbf{A}[v_1, v_2] = \begin{cases} 1 & \text{if } (v_1, v_2) \in \overrightarrow{\mathcal{E}} \\ 0 & \text{otherwise} \end{cases} . \tag{6}$$

As we can see, breaking the bidirectionality of edges by using a digraph corresponds to breaking the symmetry of the adjacency matrix.

Figure 14 gives the matrix representations of the graphs in Figure 11 and Figure 12:

$$\begin{array}{c} \\ \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{4} \\ \textcircled{5} \\ \textcircled{6} \\ \textcircled{7} \\ \textcircled{8} \\ \textcircled{9} \\ \textcircled{10} \\ \textcircled{11} \end{array} \begin{array}{c} \begin{array}{ccccccccccc} \textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} & \textcircled{5} & \textcircled{6} & \textcircled{7} & \textcircled{8} & \textcircled{9} & \textcircled{10} & \textcircled{11} \end{array} \\ \begin{bmatrix} 0 & 5 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & -8 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 & 6 & 0 & 0 & 0.1 & 0 & 0 & 0 \\ 3 & 10 & 0 & 0 & 0 & 4 & -0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 2 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & -0.5 & 2 & 0 & 0 & 4 & 6 & 11 & 0 \\ 0 & 0 & 0.1 & 0 & 6 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 11 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \sqrt{2} \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{4} \\ \textcircled{5} \\ \textcircled{6} \\ \textcircled{7} \\ \textcircled{8} \\ \textcircled{9} \\ \textcircled{10} \\ \textcircled{11} \end{array} \begin{array}{c} \begin{array}{ccccccccccc} \textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} & \textcircled{5} & \textcircled{6} & \textcircled{7} & \textcircled{8} & \textcircled{9} & \textcircled{10} & \textcircled{11} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Figure 14 – Weights matrix associated with the weighted graph in Figure 11 (left), and adjacency matrix associated with the digraph in Figure 12 (right). Vertices are labeled in the same order as in Figure 13.

Again, note that both representations are not incompatible, and that a weighted digraph can easily be represented by a non-symmetric, non-binary matrix.

Some operations on matrices

Matrices are very classical mathematical tools, on which numerous operations can be performed. In this section, we recall the ones we will need in this manuscript. For more operations and results on matrices, the reader may be interested in [Ber05].

Let us first start with simple arithmetic operations on matrices:

Definition 14: Addition of matrices

Let $\mathbf{M}_1$ and $\mathbf{M}_2$ be two $C_1 \times C_2$ matrices. ***Addition of matrices*** $\mathbf{M}_1 + \mathbf{M}_2$ is performed as follows:

$$\forall i \in [\![1, C_1]\!], \forall j \in [\![1, C_2]\!] : (\mathbf{M}_1 + \mathbf{M}_2)[i, j] = \mathbf{M}_1[i, j] + \mathbf{M}_2[i, j] \ . \tag{7}$$

Addition of matrices is a commutative and associative operation. Subtraction of matrices is defined the same way, and multiplying a matrix by a scalar is also performed entrywise. Product of two matrices, however, is defined as follows:

Definition 15: Product of matrices

Let $\mathbf{M}_1$ be a $C_1 \times C_2$ matrix, and $\mathbf{M}_2$ be a $C_2 \times C_3$ matrix. The result of the ***product of matrices*** $\mathbf{M}_1\mathbf{M}_2$ is a $C_1 \times C_3$ matrix, obtained as follows:

$$\forall i \in [\![1, C_1]\!], \forall j \in [\![1, C_3]\!] : (\mathbf{M}_1\mathbf{M}_2)[i, j] = \sum_{k=1}^{C_2} \mathbf{M}_1[i, k]\mathbf{M}_2[k, j] \ . \tag{8}$$

This is also an associative operation but, contrary to addition, it is in general not commutative. The identity element for matrix product is the *identity matrix*. We say that a matrix is ***diagonal*** if non-null values can only be found in its diagonal entries:

Definition 16: Identity matrix

The ***identity matrix*** $\mathbf{I}_C$ is a $C \times C$ diagonal matrix defined as follows:

$$\forall v_1, v_2 \in \mathcal{V} : \mathbf{I}_C[v_1, v_2] = \begin{cases} 1 & \text{if } v_1 = v_2 \\ 0 & \text{otherwise} \end{cases} \ . \tag{9}$$

Now, here are a few operations on a single matrix that will also be needed:

Definition 17: Transposition of a matrix

Let $\mathbf{M}$ be a $C_1 \times C_2$ matrix. Its ***transposed matrix*** $\mathbf{M}^\top$ is a $C_2 \times C_1$ matrix, defined as follows:

$$\forall i \in [\![1, C_2]\!], \forall j \in [\![1, C_1]\!] : \mathbf{M}^\top[i, j] = \mathbf{M}[j, i] \ . \tag{10}$$

Definition 18: Matrix inverse

We call ***inverse*** of an $N \times N$ matrix $\mathbf{M}$, a matrix $\mathbf{M}^{-1}$ such that:

$$\mathbf{M}\mathbf{M}^{-1} = \mathbf{I}_N \ . \tag{11}$$

Note that if the columns of $\mathbf{M}$ are orthonormal, then $\mathbf{M}^{-1} = \mathbf{M}^\top$.

An operation that will be extensively used in this manuscript is matrix diagonalization:

Definition 19: Matrix diagonalization

Let $\mathbf{M}$ be a $C \times C$ matrix. ***Diagonalization*** of $\mathbf{M}$ consists in its decomposition in the following form:

$$\mathbf{M} = \boldsymbol{\mathcal{X}} \mathrm{diag}(\boldsymbol{\lambda}) \boldsymbol{\mathcal{X}}^{-1} \ , \tag{12}$$

where $\boldsymbol{\mathcal{X}}$ is an orthonormal matrix, $\boldsymbol{\lambda} \in \mathbb{C}^N$ is a vector, and $\mathrm{diag}(\cdot)$ is a function that creates a diagonal matrix with entries in the order of the given vector:

$$\mathrm{diag} : \begin{cases} \mathbb{C}^N & \to & \mathbb{C}^{N \times N} \\ \mathbf{v} & \mapsto & \begin{bmatrix} \mathbf{v}[1] & 0 & \dots & 0 \\ 0 & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \mathbf{v}[N] \end{bmatrix} \end{cases} . \tag{13}$$

The matrix $\boldsymbol{\mathcal{X}}$ and vector $\boldsymbol{\lambda}$ are respectively denoted ***eigenvectors*** and ***eigenvalues*** of $\mathbf{M}$. Eigenvectors of a $C \times C$ matrix are vectors $(\boldsymbol{\chi}_1, \dots, \boldsymbol{\chi}_C)$ in $\mathbb{C}^C$, and eigenvalues are associated scalars $(\lambda_1, \dots, \lambda_C)$ such that $\forall i \in [\![1, C]\!] : \mathbf{M}\boldsymbol{\chi}_i = \lambda_i \boldsymbol{\chi}_i$. They have the following properties:

- The eigenvalues are the same for any diagonalization of $\mathbf{M}$;
- If the eigenvalues are pairwise distinct then, in absolute value, the eigenvectors are the same for any diagonalization of $\mathbf{M}$.

Diagonalization of a matrix can be seen as a transformation of its representation to represent a vector of coordinates (its eigenvalues) in a particular basis (its eigenvectors). In this manuscript, we consider without loss of generality that eigenvalues are sorted by increasing value, *i.e.*, $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_C$.

Remark 6

In all the situations we will consider in this manuscript, matrices are real-valued and symmetric, thus diagonalizable by an orthonormal matrix of eigenvectors in $\mathbb{R}^C$. This result is known as the *spectral theorem* [Wei58]. Still, note that there exist solutions to decompose non-symmetric matrices similarly using Jordan decomposition [Gan98].

Additionally, we only consider matrices such that every eigenvalue is unique in $\boldsymbol{\lambda}$. Repetition of eigenvalues leads to numerous difficulties, as there can be multiple diagonalizations of the same matrix. For this reason, we do not study this particular case.

Diagonalization allows us to write matrix power as follows:

Definition 20: Matrix power

Let $\mathbf{M}$ be a square matrix. Putting $\mathbf{M}$ at ***power*** C is equivalent to the following:

$$\mathbf{M}^C = \boldsymbol{\mathcal{X}}\mathrm{diag}(\boldsymbol{\lambda})^C\boldsymbol{\mathcal{X}}^\top \ . \tag{14}$$

Since the matrix is diagonal, the power distributes on all entries of $\mathrm{diag}(\boldsymbol{\lambda})$.

Similarly, multiplication of a matrix by a constant is equivalent to multiply its eigenvalues by this constant.

Then, let $\mathbf{M}_1$ and $\mathbf{M}_2$ be two square matrices with same dimensions, with eigenvalues $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$, respectively. If $\mathbf{M}_1$ and $\mathbf{M}_2$ have the same eigenvectors $\boldsymbol{\mathcal{X}}$, then we have the following properties:

- $\mathbf{M}_1\mathbf{M}_2 = \boldsymbol{\mathcal{X}}(\mathrm{diag}(\boldsymbol{\lambda}_1)\mathrm{diag}(\boldsymbol{\lambda}_2))\boldsymbol{\mathcal{X}}^\top = \boldsymbol{\mathcal{X}}(\mathrm{diag}(\boldsymbol{\lambda}_2)\mathrm{diag}(\boldsymbol{\lambda}_1))\boldsymbol{\mathcal{X}}^\top = \mathbf{M}_2\mathbf{M}_1$;
- $\mathbf{M}_1 + \mathbf{M}_2 = \boldsymbol{\mathcal{X}}(\mathrm{diag}(\boldsymbol{\lambda}_1) + \mathrm{diag}(\boldsymbol{\lambda}_2))\boldsymbol{\mathcal{X}}^\top$.

Following from these results, an interesting property we will use later in this manuscript is that linear combinations of matrices with the same eigenvectors yields a matrix with these eigenvectors.

To conclude with eigenvalues-related properties, we notice that the ***trace*** of a matrix — *i.e.*, the sum of its diagonal entries, noted $\mathrm{Tr}(\cdot)$ — is equal to the sum of its eigenvalues.

As for vectors, matrices are given some norms. In particular, let $\mathbf{M}$ be a $C_1 \times C_2$ matrix. Throughout this manuscript, we are interested in the following matrix norms:

- ***Matrix*** $L_{0,1}$ ***norm***: $\|\mathbf{M}\|_{0,1} = |\{(i,j) \mid \mathbf{M}[i,j] > 0, i \in [\![1, C_1]\!], j \in [\![1, C_2]\!]\}|$;
- ***Matrix*** $L_{1,1}$ ***norm***: $\|\mathbf{M}\|_{1,1} = \sum_{i \in [\![1, C_1]\!]} \sum_{j \in [\![1, C_2]\!]} |\mathbf{M}[i,j]|$;
- ***Matrix Frobenius norm***: $\|\mathbf{M}\|_F = \sum_{i \in [\![1, C_1]\!]} \sum_{j \in [\![1, C_2]\!]} |\mathbf{M}[i,j]|^2$.

A few interesting matrices

Using matrix notation to represent graphs allows the definition of other interesting matrices, that will have a lot of importance later in this manuscript. A first matrix we are interested in is the *indegrees matrix*, which indicates for each vertex the total weight of incident edges:

Definition 21: Indegrees matrix

The ***indegrees matrix*** $\mathbf{D}_{\text{in}}$ of a weighted digraph $\overrightarrow{\mathcal{G}} = \langle \mathcal{V}, \overrightarrow{\mathcal{E}}, f \rangle$ of order N and weights matrix $\mathbf{W}$ is an $N \times N$ diagonal matrix with:

$$\forall v_1, v_2 \in \mathcal{V} : \mathbf{D}_{\text{in}}[v_1, v_2] = \begin{cases} \sum_{v_3 \in \mathcal{V}} \mathbf{W}[v_3, v_1] & \text{if } v_1 = v_2 \\ 0 & \text{otherwise} \end{cases} . \tag{15}$$

Similarly, the ***outdegrees matrix*** $\mathbf{D}_{\text{out}}$ of a graph can be obtained by summing the weights of diedges that have v_1 as first element, *i.e.*, by replacing $\mathbf{W}[v_3, v_1]$ by $\mathbf{W}[v_1, v_3]$ in Definition 21. In the case of undirected graphs, these matrices are the same, and are generally denoted ***degrees matrix***, written $\mathbf{D}$.

Now, let us consider a simple graph, positively weighted and directed. The *Laplacian matrix* of this graph is a very important matrix, which can be used to provide a lot of information on the graph. The term *Laplacian* comes from the similarity between this matrix and the Laplacian operator used in multivariate calculus. This matrix measures how much a vertex differs from the vertices it is linked to. It is defined as follows:

Definition 22: Laplacian matrix

The ***Laplacian matrix*** $\mathbf{L}$ of a simple weighted digraph of order N, weights matrix $\mathbf{W}$ and degrees matrix $\mathbf{D}$ is an $N \times N$ matrix computed as follows:

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \ . \tag{16}$$

If the graph is undirected, then $\mathbf{L}$ is symmetric and real-valued, thus diagonalizable by an orthonormal matrix. Let us denote its eigenvalues $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N)$, sorted by increasing value. This matrix has numerous interesting properties, among which the following ones [Chu97]:

- $\forall i \in [\![1, N]\!] : \lambda_i \geq 0$;
- The number of occurrences of eigenvalue 0 is called ***algebraic multiplicity*** of the graph, and corresponds to the number of connected components in the graph;
- λ_2 is called ***algebraic connectivity*** of the graph, and measures how connected a graph is.

The Laplacian matrix can also be given in its normalized version as follows:

Definition 23: Normalized Laplacian matrix

The ***normalized Laplacian matrix*** $\boldsymbol{\mathcal{L}}$ of a simple weighted digraph of order N, weights matrix $\mathbf{W}$ and degrees matrix $\mathbf{D}$ is an $N \times N$ matrix computed as follows:

$$\boldsymbol{\mathcal{L}} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}} \ . \tag{17}$$

Let us denote its eigenvalues $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N)$, sorted by increasing value. As for its non-normalized version, the normalized Laplacian has numerous interesting properties, among which the following ones (see [Chu97] for more properties and details):

- $\forall i \in [\![1, N]\!] : \lambda_i \in [0, 2]$;
- $\lambda_N = 2$ if and only if the graph is ***bipartite***, *i.e.*, if its vertices can be partitioned in two disjoint sets such that all edges link vertices in distinct sets.

As presented in the introduction, these two matrices will be extensively used later in this manuscript, when considering graph signal processing.

2.1.4 Some families of graphs

Until now, most graphs that were presented in the various figures were either arbitrary, for the sake of demonstration, or were modeling some particular domains. Considering some particular graphs can be very useful in many situations, for example to take profit of their properties, or to model some situations using certain random models. In this section, we introduce the families of graphs that we will use in this manuscript.

Deterministic graphs

To ease the definition of some graphs, let us introduce the ***quotient ring*** $\mathbb{Z}/C\mathbb{Z}$. In this ring, addition is performed modulo C. To distinguish operators on this ring from classical addition or subtraction, we will denote them $+_C$ and $-_C$. Therefore, when considering a set of N vertices, we can write $\textcircled{N} +_N 1 = \textcircled{1}$ or $\textcircled{1} -_N 1 = \textcircled{N}$, indicating a difference of 1 between indices of the first and last vertex.

The first graph we introduce was already presented in the introduction, without naming it. We define a *K-ring graph* as follows:

Definition 24: K-ring graph

A ***K-ring graph*** $\mathcal{G}_r = \langle \mathcal{V}_r, \mathcal{E}_r \rangle$ of order N is a graph such that every vertex is linked to the K following and K previous vertices, following indices order, *i.e.*:

$$\forall i \in [\![1, N]\!], \forall k \in [\![1, K]\!] : \{\textcircled{i}, \textcircled{i} +_N k\} \in \mathcal{E}_r \wedge \{\textcircled{i}, \textcircled{i} -_N k\} \in \mathcal{V}_r \,. \tag{18}$$

When $K = 1$, we classically call the corresponding graph a ***ring graph***. As presented in Figure 4, a ring graph offers a good model for periodical time.

It is interesting to notice that the adjacency matrices of K-ring graph are ***circulant***, *i.e.*, are such that every row is equal to the following one, shifted by one column.

Generalizing the idea of the ring graph to higher dimensions, we introduce the *torus graph* in D dimensions as follows:

Definition 25: Torus graph

Let $\mathbf{d} \in \mathbb{N}^{*D}$. The ***torus graph*** $\mathcal{G}_t = \langle \mathcal{V}_t, \mathcal{E}_t \rangle$ yielded by the dimensions vector $\mathbf{d}$ is the graph such that:

- $\mathcal{V}_t = [\![1, \mathbf{d}[1]]\!] \times [\![1, \mathbf{d}[2]]\!] \times \cdots \times [\![1, \mathbf{d}[D]]\!]$;
- $\forall \mathbf{v}_1, \mathbf{v}_2 \in \mathcal{V}_t : \{\mathbf{v}_1, \mathbf{v}_2\} \in \mathcal{E}_t \Leftrightarrow$
  $\exists i \in [\![1, D]\!] : \begin{cases} \text{and} & |\mathbf{v}_1[i] -_{\mathbf{d}[i]} \mathbf{v}_2[i]| = 1 \\ & \forall j \in [\![1, D]\!], j \neq i : \mathbf{v}_1[j] = \mathbf{v}_2[j] \end{cases}$.

Note that elements in $\mathcal{V}_t$ are vectors of D integers, representing the Euclidean coordinates of the vertices. When $D = 2$, the torus graph offers a good model for the domain of periodic images. Each pixel is represented by a vertex, and edges link adjacent pixels.

Similar to the torus graph, we define the *grid graph* as follows:

Definition 26: Grid graph

Let $\mathbf{d} \in {\mathbb{N}^*}^D$. The ***grid graph*** $\mathcal{G}_g = \langle \mathcal{V}_g, \mathcal{E}_g \rangle$ yielded by the dimensions vector $\mathbf{d}$ is the graph such that:

- $\mathcal{V}_g = [\![1, \mathbf{d}[1]]\!] \times [\![1, \mathbf{d}[2]]\!] \times \cdots \times [\![1, \mathbf{d}[D]]\!]$;
- $\forall \mathbf{v}_1, \mathbf{v}_2 \in \mathcal{V}_g : \{\mathbf{v}_1, \mathbf{v}_2\} \in \mathcal{E}_g \Leftrightarrow$
  $\exists i \in [\![1, D]\!] : \left\{ \text{and} \quad \begin{array}{l} |\mathbf{v}_1[i] - \mathbf{v}_2[i]| = 1 \\ \forall j \in [\![1, D]\!], j \neq i : \mathbf{v}_1[j] = \mathbf{v}_2[j] \end{array} \right.$.

The only difference with the torus graph is that the subtraction of coordinates to define the edges is not performed modulo the dimension. The resulting graph is therefore a good model for non-periodic images.

Figure 15 depicts an example of a two-dimensional torus graph, and a two-dimensional grid graph, both yielded by the vector of dimensions $\mathbf{d} = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$:

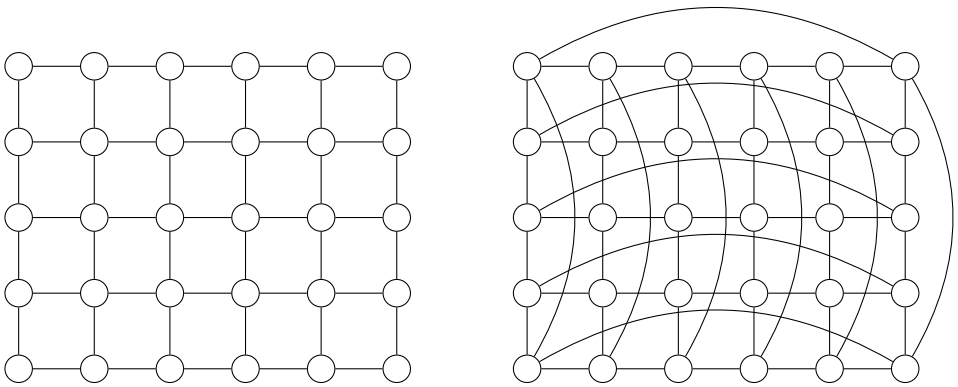


Figure 15 – Example of a grid graph (left) and torus graph (right). Each of these graphs is described by the dimensions vector $\mathbf{d} = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$.

Interestingly, the ring graph can be seen as a unidimensional torus graph. Similarly, a unidimensional grid graph is generally called a ***line graph***.

Another graph we will use at some points in this manuscript is the *complete graph*. This graph can be useful to prove some properties, as any graph has its edges included in the set of edges of a complete graph of the same order:

Definition 27: Complete graph

The ***complete graph*** $\mathcal{G}_c = \langle \mathcal{V}_c, \mathcal{E}_c \rangle$ of N vertices is the graph such that each vertex is connected to every other one, *i.e.*:

$$\forall v_1, v_2 \in \mathcal{V}_c : \{v_1, v_2\} \in \mathcal{E}_c \, . \tag{19}$$

Finally, a last graph we consider in this manuscript is the *star graph*:

Definition 28: Star graph

The ***star graph*** $\mathcal{G}_s = \langle \mathcal{V}_s, \mathcal{E}_s \rangle$ of N vertices is the graph such that one single vertex $\mathbf{v}_1 \in \mathcal{V}$ is connected to every other vertex, *i.e.*:

$$\exists v_1 \in \mathcal{V}, \forall v_2, v_3 \in \mathcal{V}_s : \{v_2, v_3\} \in \mathcal{E}_s \Leftrightarrow v_2 = v_1 \ . \tag{20}$$

As examples, Figure 16 depicts a complete graph and a star graph, both of order $N = 5$:

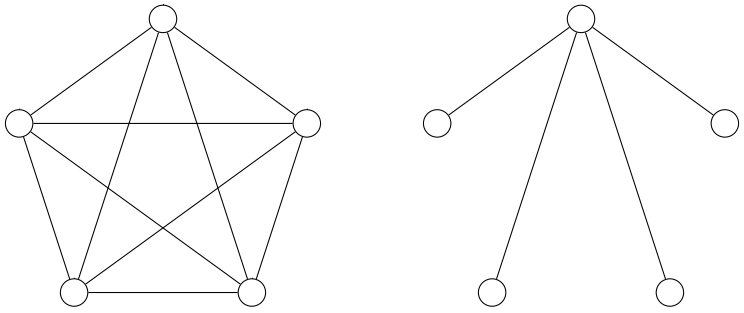


Figure 16 – Example of a complete graph (left) and star graph (right).

Random graphs

Additionally, random graphs can be useful to make simulations, allowing evaluation of an algorithm on controlled examples. In this manuscript, we consider three models of random graphs. The first one is the *Erdős-Rényi model* [ER59]:

Definition 29: Erdős-Rényi graph

A graph $\mathcal{G}_{er} = \langle \mathcal{V}_{er}, \mathcal{E}_{er} \rangle$ following an ***Erdős-Rényi model*** of parameter P is a random graph in which every edge has a probability P of existence, *i.e.*:

$$\forall v_1, v_2 \in \mathcal{V}_{er} : \begin{cases} \{v_1, v_2\} \in \mathcal{E}_{er} & \text{with probability } P \\ \{v_1, v_2\} \notin \mathcal{E}_{er} & \text{with probability } 1 - P \end{cases} . \tag{21}$$

In such graph, the edges exist or not independently from each other. The following two models we consider take the vertices into consideration when creating the edges. In particular, a *random geometric* graph is built from a set of randomly generated locations:

Definition 30: Random geometric graph

A graph $\mathcal{G}_{rg} = \langle \mathcal{V}_{rg}, \mathcal{E}_{rg} \rangle$ of order N following a ***random geometric model*** of parameter R is a graph such that every vertex is given a random location in a two-dimensional unit square, and is connected to other vertices that are located under a certain radius, *i.e.*:

- $\mathcal{V}_{rg} = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$, with $\forall i \in [\![1, N]\!] : \mathbf{v}_i \in [0, 1]^2$;
- $\forall \mathbf{v}_1, \mathbf{v}_2 \in \mathcal{V}_{rg} : \{\mathbf{v}_1, \mathbf{v}_2\} \in \mathcal{E}_{rg} \Leftrightarrow \|\mathbf{v}_1 - \mathbf{v}_2\|_2 < R$, where $\|\cdot\|_2$ measures the ℓ_2 norm of a vector.

Note that random geometric graphs are sometimes defined using coordinates on a two-dimensional unit torus. In this case, distance between them is computed accordingly. Since edges between vertices indicate a certain geographic proximity between them, such graphs can be very convenient to model networks [Nek07].

Finally, the third model of random graphs we consider is the *Watts-Strogatz model* [WS98]:

Definition 31: Watts-Strogatz graph

A graph $\mathcal{G}_{ws} = \langle \mathcal{V}_{ws}, \mathcal{E}_{ws} \rangle$ following a ***Watts-Strogatz model*** of parameters K and P is built iteratively from a K-ring graph $\mathcal{G}_r = \langle \mathcal{V}_r, \mathcal{E}_r \rangle$, by rewiring edges with probability P and avoiding self-loops and duplicates, *i.e.*:

- $\mathcal{V}_{ws} = \mathcal{V}_r$;
- $\forall \{v_1, v_2\} \in \mathcal{E}_r : \begin{cases} \{v_1, v_2\} \in \mathcal{E}_{ws} & \text{with probability } 1-P \\ \exists v_3 \in \mathcal{V}_{ws}, v_1 \neq v_3, \{v_1, v_3\} \notin \mathcal{E}_{ws} : \{v_1, v_3\} \in \mathcal{E}_{ws} & \text{with probability } P \end{cases}$.

In this model, K controls the original neighborhood of every vertex, and P controls the quantity of *disorder* in the graph. For $P = 0$, the graph is a K-ring, and for $P = 1$, it is a completely randomized simple graph.

All the random graph models introduced in this section are unweighted. However, it is sometimes interesting to consider weighted graphs. When the random graph is built from vertices that have Euclidean coordinates, weights can be added to edges considering the distance separating them.

From Definition 12, note that absence of an edge is represented by a 0 in the corresponding entry in the weights matrix. However, a low value indicates existence of an edge with a small weight in the graph. If weights represent distances, this causes a discontinuity in the weights function, since 0 represents infinite distance, and $\varepsilon > 0$ represents high similarity between vertices.

To correct this discontinuity, a classical approach to add weights to a random graph is to consider a function that decreases with the distance. The most common such function is a decreasing exponential of the distance, as follows:

$$\forall v_1, v_2 \in \mathcal{V} : \mathbf{W}[v_1, v_2] = \begin{cases} \exp\left(-\frac{d(v_1,v_2)^2}{2\theta^2}\right) & \text{if } \mathbf{A}[v_1, v_2] \neq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{22}$$

where $d(v_1, v_2)$ is the Euclidean distance using the coordinates associated with the vertices, and θ is a parameter controlling the decrease.

Numerous other solutions exist, some of which can be found in [GP10]. Also, when considering graphs for which vertices are not associated with coordinates, the weights can for example be drawn randomly using a chosen distribution.

2.2 Elements of signal processing

As presented in the introduction of this manuscript, signal processing on graphs developed as an extension of classical signal processing, with the aim to generalize it to more complex domains than classical tools could handle. In this section, we review a few notions of signal processing. Their equivalents from a graph signal processing point of view will be introduced in the next section.

2.2.1 What is signal processing?

The Fourier transform of signals

Hearing music, or making a phone call for instance, are operations that involve time. Every time instant is associated with a signal quantity, corresponding in these cases to the level of sounds that can be heard at this particular instant. As time is a continuous domain, such signals can be defined as continuous functions as follows:

Definition 32: Continuous signal

A ***continuous signal*** over $\mathbb{R}$ is a function $s : \left\{ \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ t & \mapsto & s(t) \end{array} \right.$.

Signal value at instant t is called ***amplitude*** of the signal.

Analysis of such objects is the objective of continuous signal processing. One of the key tools allowing this is the Fourier transform of signals, allowing the representation of time signals as a continuous sum of sines, or complex exponentials:

Definition 33: Fourier transform

The ***Fourier transform*** $\widehat{s} = \mathrm{FT}(s)$ of a continuous, integrable, signal $s(t)$ is:

$$\mathrm{FT}(s) : \left\{ \begin{array}{ccc} \mathbb{R} & \to & \mathbb{C} \\ \omega & \mapsto & \widehat{s}(\omega) = \int_{\mathbb{R}} s(t)\, \mathrm{e}^{-i2\pi\omega t}\, \mathrm{d}t \end{array} \right. , \tag{23}$$

with t in seconds, and ω the frequency in Hz.

The Fourier transform of a signal gives a frequency representation for it called its ***spectrum***, allowing its manipulation in a different context. Spectrum value at frequency ω is called ***amplitude*** of the spectrum.

Using *inverse Fourier transform*, it is then possible to bring the signal back to the time domain as follows:

Definition 34: Inverse Fourier transform

The ***inverse Fourier transform*** $s = \mathrm{FT}^{-1}(\widehat{s})$ of an integrable spectrum $\widehat{s}(\omega)$ is:

$$\mathrm{FT}^{-1}(\widehat{s}) : \left\{ \begin{array}{ccc} \mathbb{R} & \to & \mathbb{C} \\ t & \mapsto & s(t) = \int_{\mathbb{R}} \widehat{s}(\omega)\, \mathrm{e}^{+i2\pi\omega t}\, \mathrm{d}\omega \end{array} \right. . \tag{24}$$

From continuous signals to discrete signals

As introduced before, spectral analysis of continuous signals can be done thanks to the Fourier transform. However, computers can only manipulate discrete objects, and processing signals with such tools requires to *sample* the signals:

Definition 35: Signal sampling

Signal sampling is the process of approximating a continuous signal by a discrete signal with entries taking their values in a finite set $\mathcal{S}$. It is done in two steps:

1. ***Discretization*** of the signal consists in dividing it into equal intervals of time, each interval being represented by a single amplitude;
2. ***Quantization*** of the discretized signal consists in approximating the amplitude by a finite value in $\mathcal{S}$.

Discretized signals are then vectors — potentially infinite — taking their values in $\mathcal{S}$. Manipulating discrete signals can then be done similarly to continuous signals, using the *discrete Fourier transform* and the *inverse discrete Fourier transform*:

Definition 36: Discrete Fourier transform

The ***discrete Fourier transform*** $\widehat{\mathbf{t}} = \mathrm{DFT}(\mathbf{t})$ of a discrete signal $\mathbf{t}[t]$ of N samples is defined by:

$$\mathrm{DFT}(\mathbf{t}) : \begin{cases} \mathbb{R} & \to & \mathbb{C} \\ \omega & \mapsto & \widehat{\mathbf{t}}[\omega] = \sum_{t=1}^{N} \mathbf{t}[t]\, \mathrm{e}^{-i2\pi\omega t} \end{cases}, \tag{25}$$

where $\widehat{\mathbf{t}}$ can be seen as a vector with non-integer indices associated with the various frequencies.

Definition 37: Inverse discrete Fourier transform

The ***inverse discrete Fourier transform*** $\mathbf{t} = \mathrm{DFT}^{-1}(\widehat{\mathbf{t}})$ of a spectrum $\widehat{\mathbf{t}}[\omega]$ is:

$$\mathrm{DFT}^{-1}(\widehat{\mathbf{t}}) : \begin{cases} \mathbb{R} & \to & \mathbb{C} \\ t & \mapsto & \mathbf{t}[t] = \sum_{\omega=1}^{N} \widehat{\mathbf{t}}[\omega]\, \mathrm{e}^{+i2\pi\omega t} \end{cases}, \tag{26}$$

where $\mathbf{t}$ can be seen as a vector with non-integer indices associated with the various sampling instants.

Manipulating discrete signals as an approximate for continuous signals suggests a loss in details. However, the Nyquist-Shannon sampling theorem [Sha49] states that a signal can be fully determined by a correct subsampling:

Theorem 1: Nyquist-Shannon sampling theorem

A signal $s(t)$ of highest frequency $\omega_{\max}$Hz is fully determined by subsampling it with a period of $\frac{1}{2\omega_{\max}}$.

This theorem justifies subsampling as a very powerful tool to manipulate signals. However, in the case when the signal ***bandwidth*** — *i.e.*, the difference between its maximal and minimal frequencies — is infinite, then Theorem 1 does not apply, and reconstruction of the initial signal from its sampled version necessarily exhibits imperfections.

2.2.2 Operations on signals

Using Fourier transform and its inverse simplifies manipulation of signals. In this section, we introduce some classical tools to process time signals. The tools we introduce here are defined for continuous signals. However, the results also apply for discrete signals by replacing integrals with discrete sums.

Filtering signals

Filtering signals allows to remove unwanted components from them such as noise, or to extract information located in particular frequencies. One of the cornerstone tools allowing this manipulation of signals is *convolution*, defined as follows:

Definition 38: Convolution of signals

Convolution of two signals, noted $s_1 * s_2$, is a function that associates with each time instant the sum of one signal, weighted by the other around origin, *i.e.*:

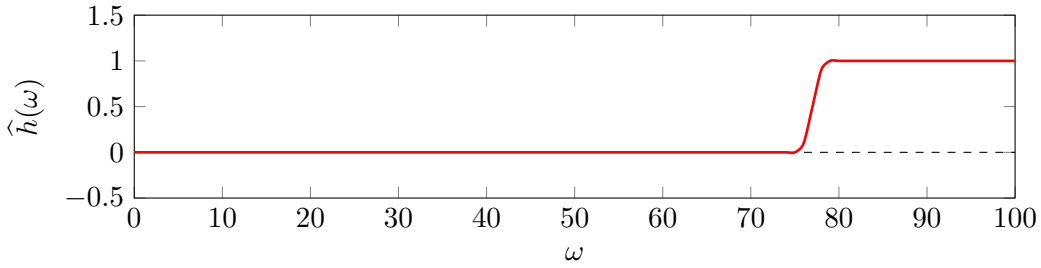
$$(s_1 * s_2)(t) = \int_{\mathbb{R}} s_1(t-\tau)s_2(t)\,\mathrm{d}\tau = \int_{\mathbb{R}} s_1(t)s_2(t-\tau)\,\mathrm{d}\tau\ . \tag{27}$$

What makes convolution an essential tool to filter signals is the fact that convolving signals in the time domain is equivalent to multiplying their spectrums:

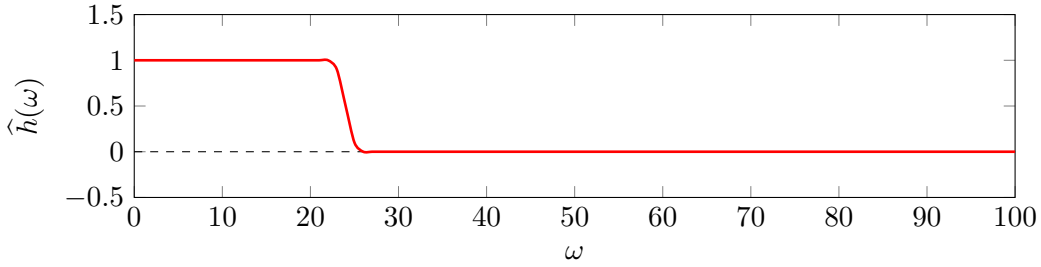
$$s_1 * s_2 = \mathrm{FT}^{-1}(\mathrm{FT}(s_1)\mathrm{FT}(s_2)) = \mathrm{FT}^{-1}(\widehat{s_1}\widehat{s_2})\ . \tag{28}$$

Using this property, one can then build a signal in the time domain such that it has desirable properties in the spectral domain, thus creating a filter to convolve with another signal of interest. The most common such signals are the following, each given with an example spectral representations:

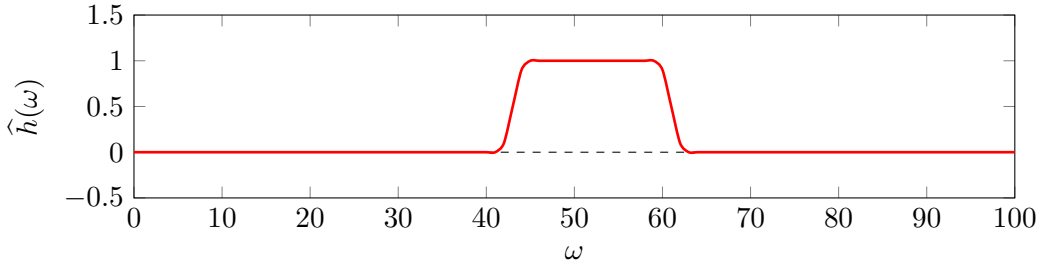
- ***High-pass filter***: The highest frequencies are kept, and the others are attenuated:



- ***Low-pass filter***: The lowest frequencies are kept, and the others are attenuated:



- ***Band-pass filter***: An interval of frequencies is kept, and the others are attenuated:



To illustrate filtering, let us consider the following example. It is well known that noise in a signal is located in the high frequencies. Therefore, it is easier to identify — and remove — noise on a signal when considering its spectrum. Figure 17 depicts an example of a noisy signal from which we have removed the 25% highest frequencies:

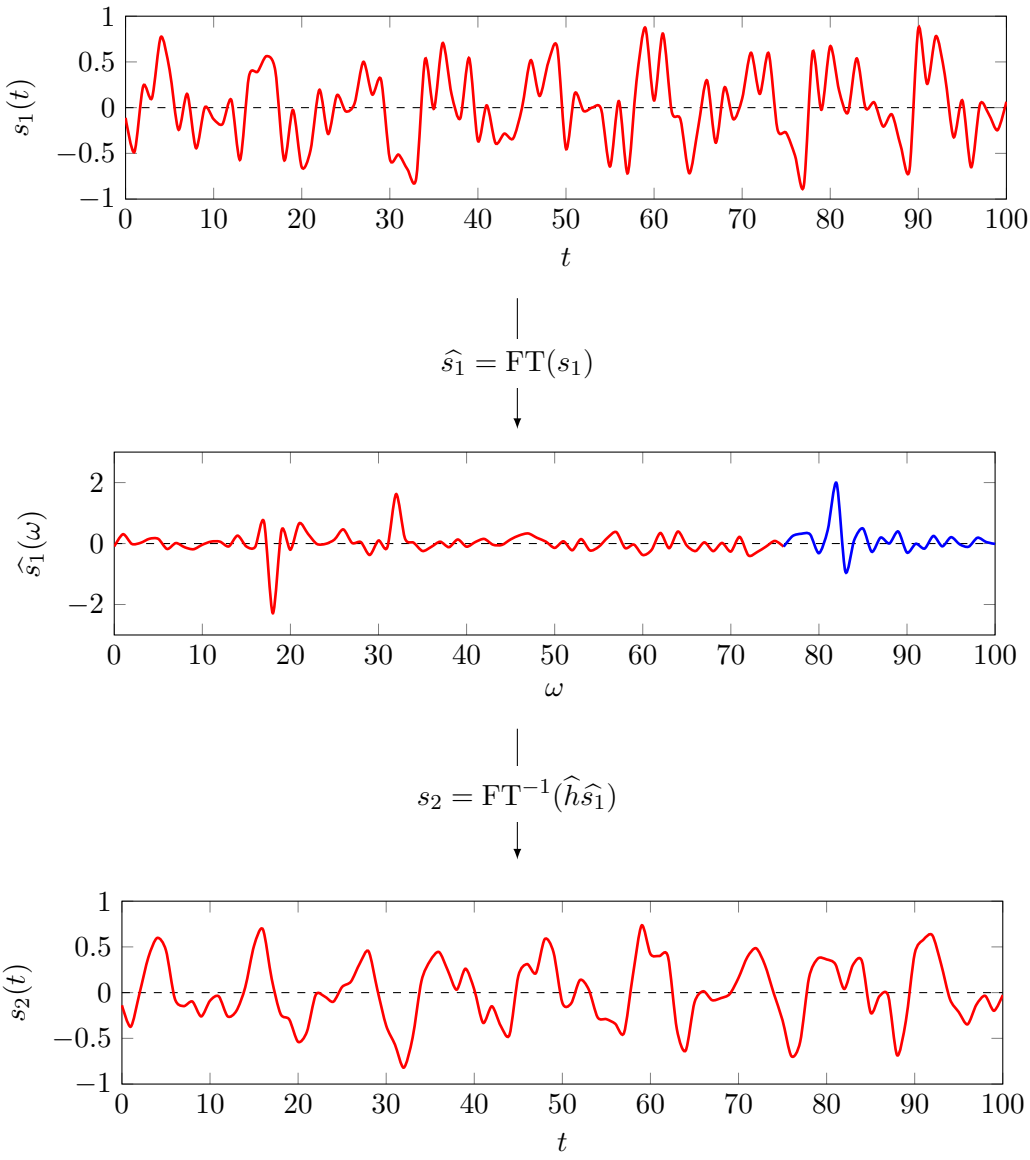


Figure 17 – Example of denoising of a signal, represented both in the time domain (top) and in the frequency domain (middle). Frequencies that are depicted in blue are removed using a low-pass filter h. Inverse Fourier transform of the corrected signal shows that it is now more regular (bottom).

Additionally, a particular signal is of interest for convolution:

Definition 39: Dirac delta signal

A ***Dirac delta signal*** at time t is a signal $\delta(t)$ such that:

$$\delta(t) = \begin{cases} +\infty & \text{if } t = 0 \\ 0 & \text{otherwise} \end{cases} . \tag{29}$$

This particular signal has the interesting property to have an equal contribution of all its frequencies. Therefore, convolving it with an unknown filter allows to identify the spectrum of this filter.

A few other operators on signals

Filtering is only one possible operation on signals. Signal processing defines a lot of other operators that can help process such objects. One particular operation that was implicitly used in Definition 38 is *translation*:

Definition 40: Signal translation

Translation of a signal $s(t)$ by a quantity τ is performed by a change of variable, *i.e.*, by replacing $s(t)$ with $s(t-\tau)$.

This operator allows to shift a signal in time. The corresponding operation in the spectral domain is *modulation*:

Definition 41: Signal modulation

Modulation of a signal $s(t)$ by a frequency ξ is performed as follows:

$$\mathrm{M}_\xi s(t) = \mathrm{e}^{i2\pi\xi t} s(t) . \tag{30}$$

This operation corresponds in the spectral domain to the following:

$$\widehat{\mathrm{M}_\xi s}(\omega) = \widehat{s}(\omega - \xi) . \tag{31}$$

Numerous other operations on signals exist, allowing for instance to dilate, scale or reverse signals. For more information on such operators, the interested reader may consult [Pri90] for instance.

2.3 Graph signal processing

As indicated in the introduction of this manuscript, the objective of graph signal processing is to generalize the classical signal analysis framework in order to make it applicable to more complex signals, evolving on elaborate topologies. In this section, we introduce in more details the link between the Laplacian matrix — normalized or not — and show how classical tools from signal processing can be transported to graphs. Then, we introduce some properties of the underlying graph used to provide a frequency representation of signals.

2.3.1 From signal processing to graph signal processing

Signals and graphs

As shown before, a discretized time signal can be seen as a vector of real entries, carried by vertices of a ring graph modeling time. We therefore manipulate two distinct objects: a graph, and some *signals* on this graph:

Definition 42: Signal on a graph

A ***signal*** $\mathbf{x}$ on a graph of order N is a column vector in $\mathbb{R}^N$.

When considering a ring graph, we have seen in Figure 5 that the eigenvectors $\boldsymbol{\mathcal{X}} = (\boldsymbol{\chi}_1, \dots, \boldsymbol{\chi}_N)$ of the Laplacian matrix correspond to the sines that are used to provide a spectral representation for signals in classical Fourier analysis. Additionally, the frequencies of these sines increase as the eigenvalues associated with the corresponding eigenvectors increase.

As a consequence, projection of a signal $\mathbf{x}$ to the basis formed by $\boldsymbol{\mathcal{X}}$ corresponds to projecting it to a basis of sines. This leads to the following definition of the *graph Fourier transform*, introduced in [Shu+13]:

Definition 43: Graph Fourier transform

The ***graph Fourier transform*** $\widehat{\mathbf{x}} = \mathrm{GFT}(\mathbf{x})$ of a signal $\mathbf{x}$ on a graph is:

$$\widehat{\mathbf{x}} = \boldsymbol{\mathcal{X}}^\top \mathbf{x} \,. \tag{32}$$

Conversely, retrieving the graph signal from its spectrum can be done by projecting it back to the graph domain using the *inverse graph Fourier transform*:

Definition 44: Inverse graph Fourier transform

The ***inverse graph Fourier transform*** $\mathbf{x} = \mathrm{GFT}^{-1}(\widehat{\mathbf{x}})$ of the spectrum $\widehat{\mathbf{x}}$ of a signal on a graph is:

$$\mathbf{x} = \boldsymbol{\mathcal{X}} \widehat{\mathbf{x}} \,. \tag{33}$$

Note that orthonormality of the eigenvectors makes this function a valid inverse of the graph Fourier transform:

$$\mathrm{GFT}^{-1}(\mathrm{GFT}(\mathbf{x})) = \boldsymbol{\mathcal{X}} \boldsymbol{\mathcal{X}}^\top \mathbf{x} = \mathbf{x} \,. \tag{34}$$

Note that alternative definitions exist for the graph Fourier transform and its inverse. In particular, it is possible to use another basis to provide a spectral representation of signals. Classical alternatives to the Laplacian matrix are its normalized version [Shu+13], or the adjacency matrix of the graph [SM13; SM14].

To illustrate graph Fourier transform, Figure 18 depicts a random geometric graph, on which a signal is observed. Using the eigenvectors of the Laplacian matrix, the spectrum of the signal is provided:

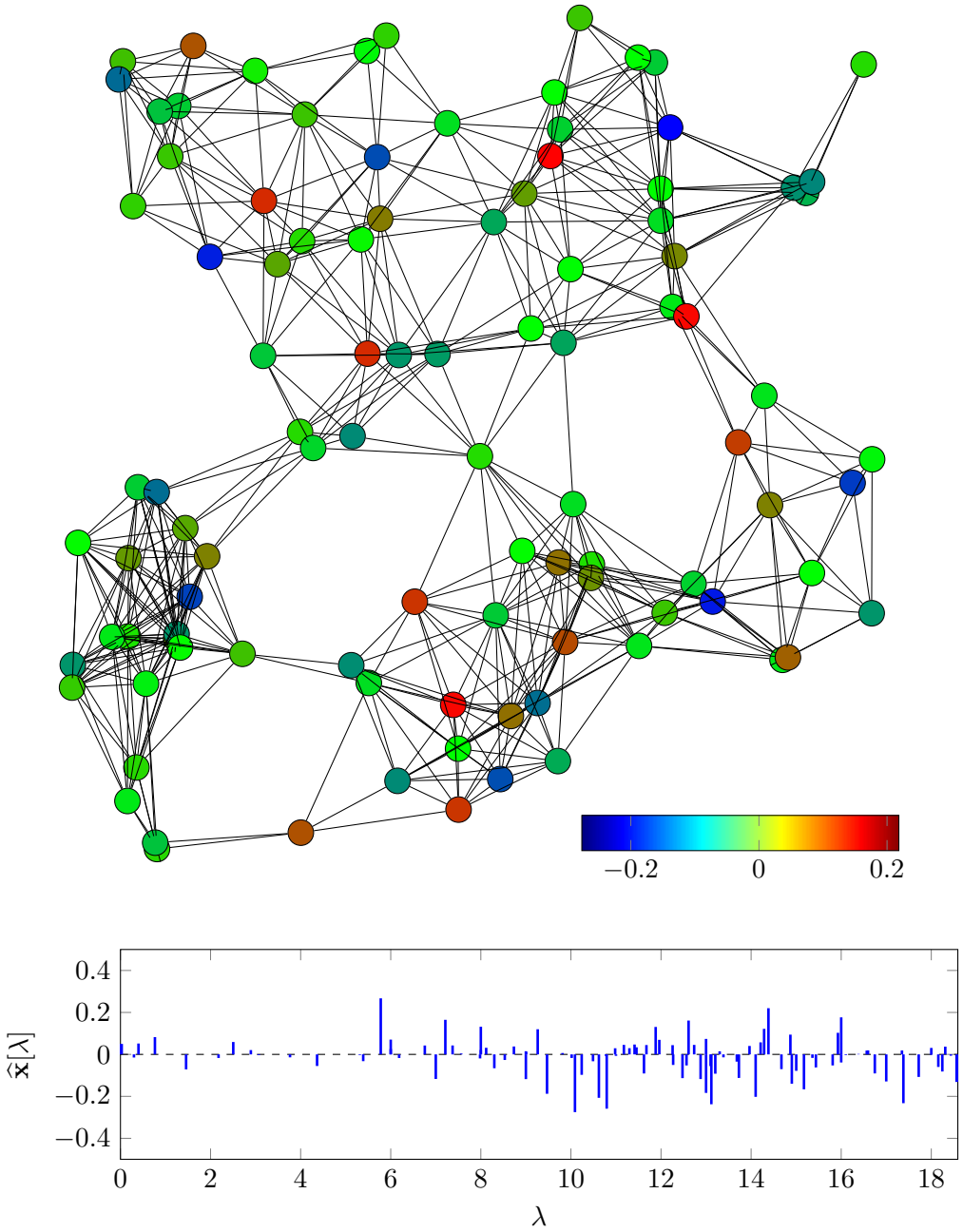


Figure 18 – Example of a random geometric graph, on which a signal $\mathbf{x}$ is observed (top). Entries of the signal on the vertices are represented with colors according to the given color map. The spectral representation of the signal is given by its graph Fourier transform using the Laplacian matrix (bottom). As for discrete signals in classical signal processing, we use the notation $\widehat{\mathbf{x}}[\lambda]$ for the amplitude of the spectrum at eigenvalue λ.

An interesting observation is that the spectrum of the signal is not defined for every frequency, and that spacing of these frequencies is not regular. This is due to the distribution of the eigenvalues of the Laplacian matrix of the graph. More information on the distributions of random graphs can be found for example in [DJ+10].

This irregular spacing is characteristic of the problems one may encounter when trying to define tools on such graphs to process signals. This is also true in the graph domain, in which finding regular patterns is not an easy task. For these reasons, definition of operators such as translation or modulation is not straightforward.

Operators on signals on graphs

Porting the convolution operator in Definition 38 to signals defined on a graph is not an evident operation. As a matter of fact, its definition requires being able to shift a signal by a quantity τ, which has no real meaning in the irregular domain of the graph. In the spectral domain, however, we do not have this particular limitation, as two signals on a graph share the same domain for their spectrum. Therefore, multiplying the corresponding entries can be done easily.

Definition 45: Convolution of signals on a graph

Convolution of two signals on a graph is performed in the spectral domain analogously to classical convolution. The corresponding signal in the time domain is then obtained by inverse graph Fourier transform:

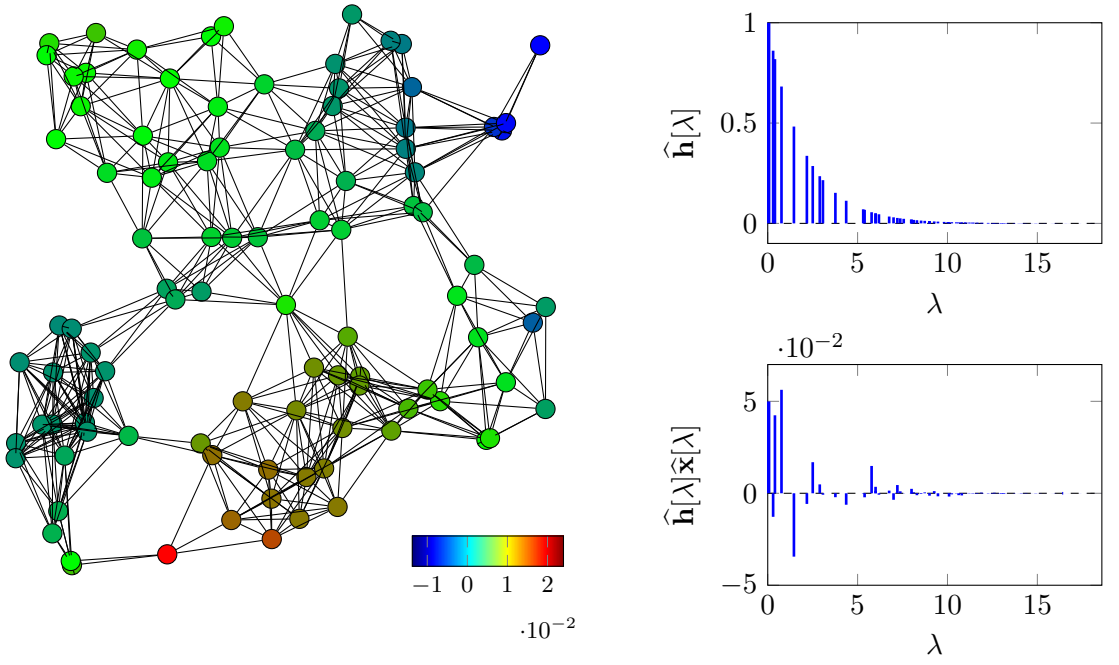
$$\mathbf{x}_1 * \mathbf{x}_2 = \mathrm{GFT}^{-1}(\mathrm{GFT}(\mathbf{x}_1) \odot \mathrm{GFT}(\mathbf{x}_2)) = \mathrm{GFT}^{-1}(\widehat{\mathbf{x}_1} \odot \widehat{\mathbf{x}_2}) \ , \tag{35}$$

where $\odot$ is the ***entrywise product*** of vectors.

Filtering of signals on graphs can then be done as for classical ones, by convolving them with a filter that has interesting spectral properties. As an example, let us consider again the signal $\mathbf{x}$ in Figure 18. Figure 19 depicts the same signal, convolved with a signal $\mathbf{h}$ which spectrum is defined as follows:

$$\widehat{\mathbf{h}}[\lambda] = \mathrm{e}^{-C\lambda} \ , \tag{36}$$

where C is a controlling parameter, for all eigenvalue λ of the Laplacian matrix of the graph in Figure 18. As all eigenvalues of the Laplacian matrix are positive (see Section 2.1.3), this corresponds to a low-pass filter on the graph:



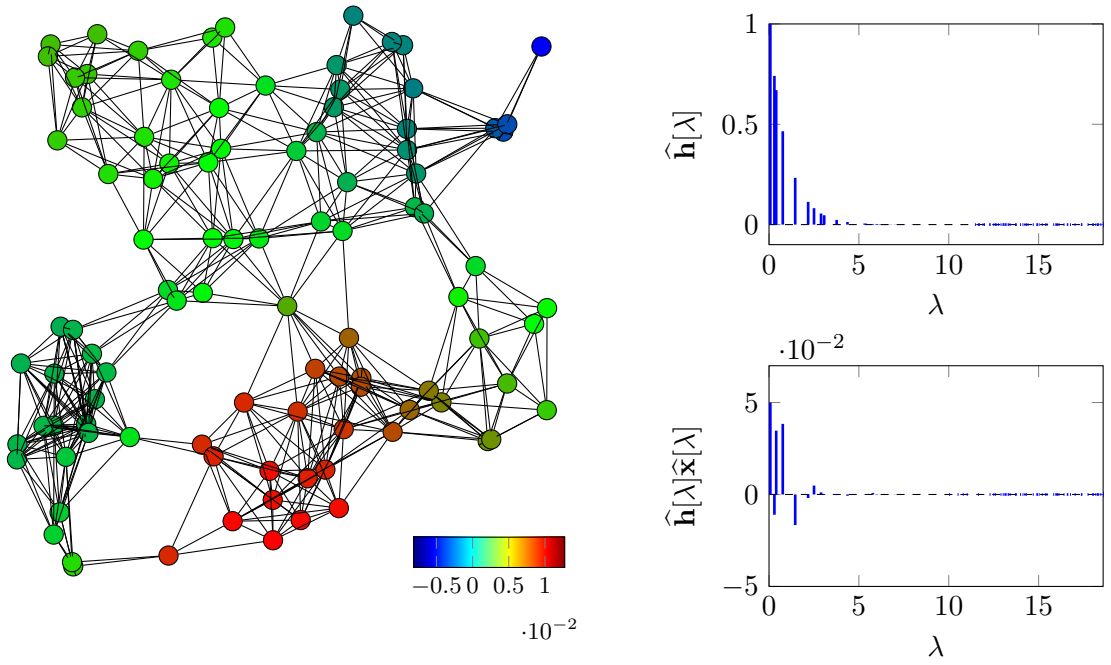


Figure 19 – Convolution of the signal $\mathbf{x}$ from Figure 18 with a low-pass filter $\mathbf{h}$ defined for $C = 0.5$ (top) and $C = 1$ (bottom) in (36). The filtered signal is represented on the graph (left), and details on its spectrum and on the filter used are also given (right).

As it can be seen from this figure, applying a low-pass filter to a signal on a graph also tends to make it more regular, as it was the case for time signals in Figure 17.

This process of finding equivalences with classical signal processing — either in the graph domain or in the spectral domain — is the key idea to define most operators for signal processing on graphs.

In particular, translation of signals on a graph cannot simply be defined by associating an index to each vertex and replacing the signal entry at vertex v by the signal entry at vertex $v -_N 1$, except for the ring graph. Therefore, translation of signals on graphs is defined using the property in classical signal processing that translation of a time signal corresponds to convolution with a Dirac delta signal at a target time instant. To apply this to graphs, let us first introduce the *Dirac delta signal on the graph*:

Definition 46: Dirac delta signal on a graph

A ***Dirac delta signal*** on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ of N vertices at vertex $v_1 \in \mathcal{V}$ is a signal $\boldsymbol{\delta}_{v_1} \in \mathbb{R}^N$ such that:

$$\boldsymbol{\delta}_{v_1}[v_2] = \begin{cases} 1 & \text{if } v_2 = v_1 \\ 0 & \text{otherwise} \end{cases} . \tag{37}$$

Using this particular signal, we can translate a signal to a particular vertex as follows:

Definition 47: Translation of a signal on a graph

Let us consider a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ of order N. ***Translation*** of a signal $\mathbf{x}$ to a particular vertex $v_1 \in \mathcal{V}$, as introduced by Shuman *et al.* in [Shu+13], is defined as follows:

$$\mathrm{T}_{v_1}\mathbf{x}[v_2] = \sqrt{N}(\mathbf{x} * \boldsymbol{\delta}_{v_1})[v_2] \; . \tag{38}$$

In this equation, $\sqrt{N}$ ensures preservation of the mean of the signal.

Note that this translation operator moves the whole signal around a target vertex, rather than reaching it with a series of small shifts. As an example, Figure 20 depicts a graph, on which a signal is translated to various locations:

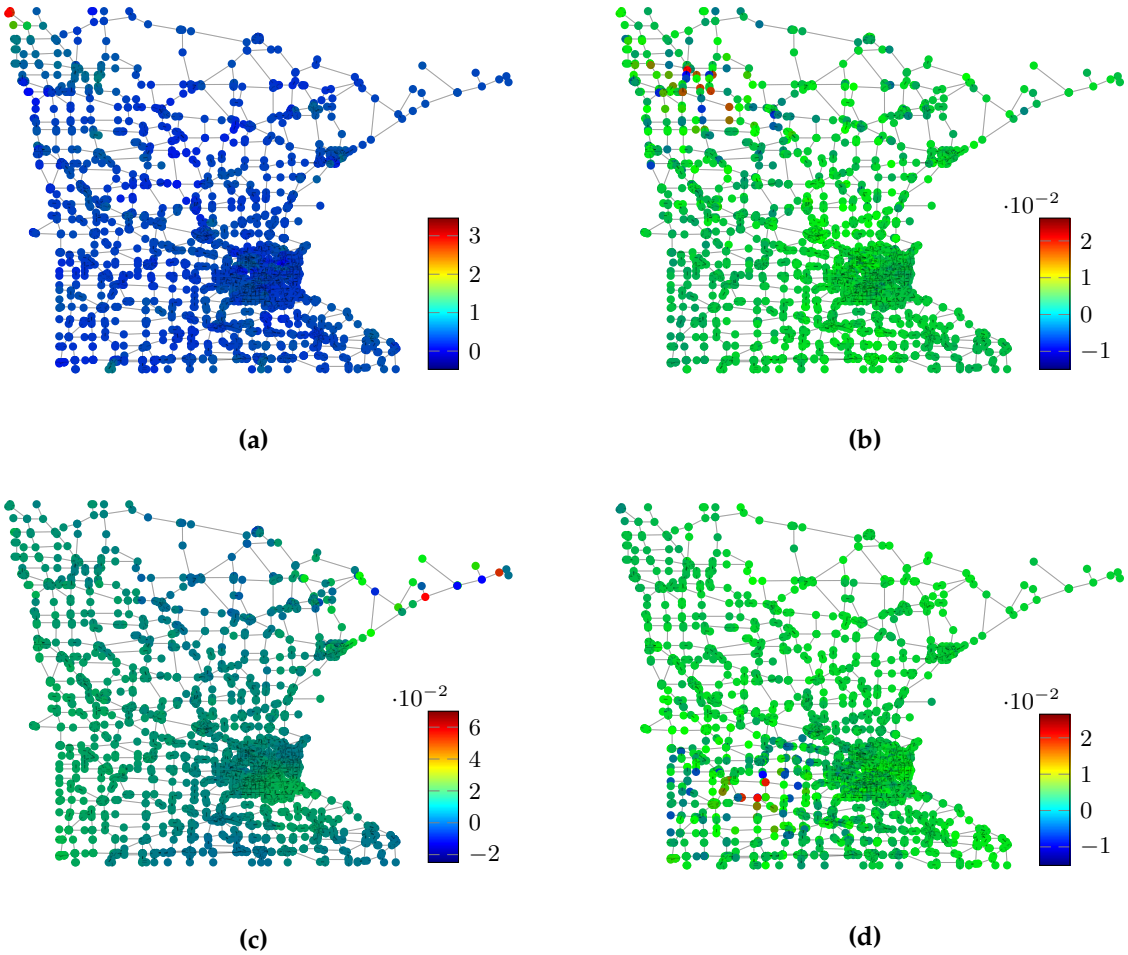


Figure 20 – Reproduction of an experiment performed in [Shu+13]. This graph represents the roads of Minnesota. An initial signal **(a)**, defined in the spectral domain using (36) with $C = 5$, is translated to vertices v_{100} **(b)**, v_{200} **(c)**, and v_{2000} **(d)**.

Also, this operator does not preserve the ℓ_2 norm of the translated signal in the general case. For this reason, other definitions for translation on graphs have been proposed. Since translation of signals is an important part of this manuscript, these methods will be presented in Chapter 4.

Similarly to translation, that was not straightforward to define due to irregularity of

the graph domain, modulation suffers from the irregularity of the spectrum, *i.e.*, the variable space between consecutive eigenvalues of the Laplacian matrix. As introduced in Definition 41, modulation can be defined in the graph domain, by multiplying the signal to modulate with a complex exponential of a certain frequency. Noticing that this exponential is in fact an eigenvector of the Laplacian matrix of the ring graph, *modulation* of a signal on a graph is defined in the graph domain as follows:

Definition 48: Modulation of a signal on a graph

Let us consider a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ of order N, and let $(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_N)$ be the eigenvectors of its Laplacian matrix. ***Modulation*** of a signal $\mathbf{x}$ on a graph by a frequency λ_i (of associated eigenvector $\boldsymbol{\chi}_i$) is performed as follows:

$$\mathrm{M}_{\lambda_i}\mathbf{x}[v] = \sqrt{N}\boldsymbol{\chi}_i[v]\mathbf{x}[v] \ . \tag{39}$$

Contrary to time signals, modulation of a signal on a graph is not exactly a translation in the spectral domain. However, Shuman *et al.* have shown in [SRV12] that modulating a signal which spectrum is localized at eigenvalue 0 by an eigenvalue λ causes the spectrum of the modulated signal to be localized around λ.

Other tools from classical signal processing such as modulation of a signal also have their equivalents in the framework of graph signal processing. Examples include dilatation of a signal, or wavelets on graphs. Since they will not be used in this manuscript, we refer the interested reader to [HVG11] or [Shu+13].

2.3.2 The underlying graph and the signals

Obviously, the underlying graph topology has a lot importance in graph signal processing. We already established that the eigenvectors of its Laplacian matrix define a graph Fourier transform, and that the distribution of its eigenvalues impacts the design of filters for signals defined on the graph. In this section, we review some additional properties of signals defined on the graph.

Smoothness of a signal on a graph

In Figure 19, we have implicitly established that keeping only the lowest frequencies of a signal on a graph tends to make it more regular on the graph, analogously to time signals. More formally, this property of regularity is measured by the *smoothness* of the signal on the graph as follows:

Definition 49: Smoothness of a signal on a graph

Smoothness of a signal $\mathbf{x}$ on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, of weights matrix $\mathbf{W}$ and Laplacian matrix $\mathbf{L}$, measures the total difference between signal entries on adjacent vertices, as follows:

$$\begin{aligned} \mathrm{S}(\mathbf{x}) &= \tfrac{1}{2} \sum_{v_1 \in \mathcal{V}} \sum_{v_2 \in \mathcal{N}(v_1)} \mathbf{W}[v_1, v_2](\mathbf{x}[v_1] - \mathbf{x}[v_2])^2 \\ &= \sum_{\{v_1, v_2\} \in \mathcal{E}} \mathbf{W}[v_1, v_2](\mathbf{x}[v_1] - \mathbf{x}[v_2])^2 \\ &= \mathbf{x}^\top \mathbf{L} \mathbf{x} \end{aligned} \ . \tag{40}$$

Since $\mathbf{L}$ is diagonalizable by an orthonormal matrix of eigenvectors in $\mathbb{R}^C$, we can rewrite this equality as follows:

$$\begin{aligned} \mathrm{S}(\mathbf{x}) &= \mathbf{x}^\top \boldsymbol{\mathcal{X}} \mathrm{diag}(\boldsymbol{\lambda}) \boldsymbol{\mathcal{X}}^\top \mathbf{x} \\ &= \widehat{\mathbf{x}}^\top \mathrm{diag}(\boldsymbol{\lambda}) \widehat{\mathbf{x}} \\ &= \sum_{\lambda \in \boldsymbol{\lambda}} \lambda \widehat{\mathbf{x}}[\lambda]^2 \end{aligned} , \tag{41}$$

where $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\lambda}$ are the eigenvectors and eigenvalues of the Lalacian matrix.

Considering the first part of the computation, note that $\mathrm{S}(\mathbf{x})$ can only be null when the signal $\mathbf{x}$ is constant. Also, this quantity is low when the signal entries on vertices that are linked by an edge with a strong weight are close. For this reason, a low smoothness value indicates more regularity on the graph.

Additionally, the second part of the computation in Definition 49 tells us that for a signal to be smooth, most of the signal energy should be located on the smaller eigenvalues. To illustrate this, let us consider again the signal $\mathbf{x}$ in Figure 18, and its filtered versions in Figure 19. For these signals, we obtain the following smoothness values:

- $\mathrm{S}(\mathbf{x}) = 12.037$;
- For $C = 0.5$: $\mathrm{S}(\mathbf{h} * \mathbf{x}) = 7.81 \cdot 10^{-3}$;
- For $C = 1$: $\mathrm{S}(\mathbf{h} * \mathbf{x}) = 2.13 \cdot 10^{-3}$.

This confirms the observation that application of a low-pass filter, as in the case of time signal, make signals on graphs more regular.

Stationarity of signals on a graph

In statistics, a *stationary process* is defined as follows:

Definition 50: Stationary process

A stochastic process is said to be ***stationary*** if its joint probability distribution does not change when shifted, *i.e.*, if its moments are invariant by translation.

This strict definition of stationarity — also called ***strong-sense stationarity*** — is quite restrictive, and is generally relaxed to consider a weaker form of stationarity, defined as follows:

Definition 51: Weak-sense stationary process

A stochastic process is said to be ***weak-sense stationary*** if its two first moments — mean and autocovariance — are invariant by translation.

As an example, Figure 21 depicts two signals $s_1(t)$ and $s_2(t)$. The first one is an instantiation of a strong-sense stationary process, and corresponds to ***white noise***, *i.e.*, *i.i.d.* realizations of a zero-mean, fixed variance random process. The second one is an instantiation of a process that is not stationary, and represents the selling price of the *Magic: the Gathering* card *Serra Angel*, in edition *Alpha* [Mag]:

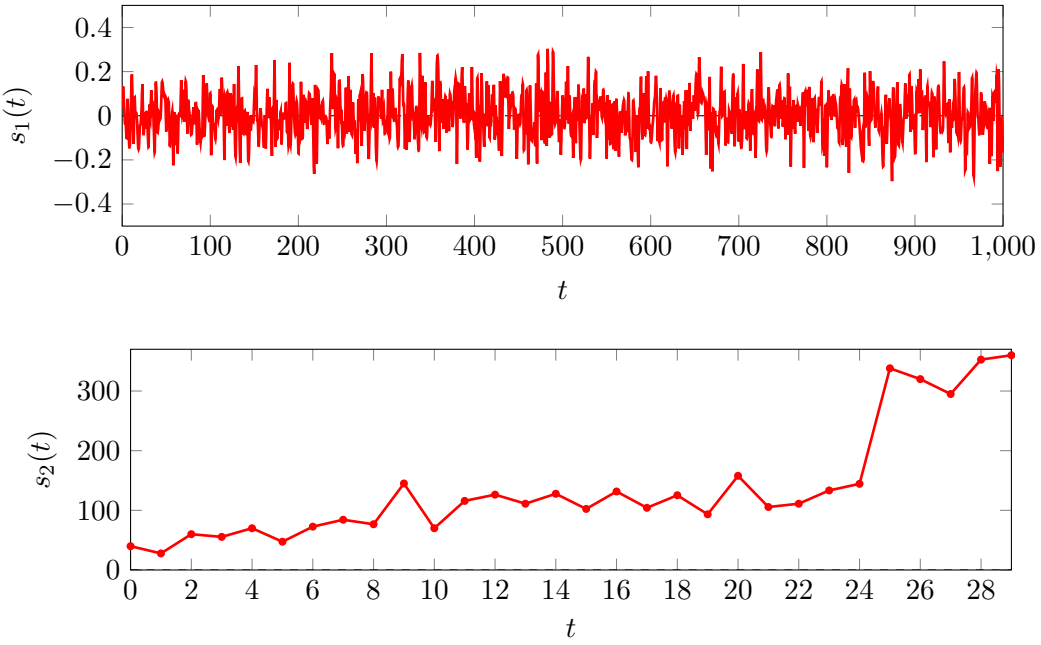


Figure 21 – Example of a stationary signal $s_1(t)$ corresponding to white noise obtained from *i.i.d.* realizations of a $\mathcal{N}(0, 0.1)$ distribution (top). Additionally, a non-stationary signal $s_2(t)$ is provided (bottom), corresponding to the average selling price (in Euro) of the *Magic: the Gathering* card *Serra Angel*, in edition *Alpha* between the months of February 2013 and May 2017.

Considering (weak-sense) stationary processes is a frequent assumption in signal processing, as it simplifies numerous operations. Therefore, it has applications to fields such as detection theory, reliability or filtering [Lin12].

When considering graphs, stationarity of processes is defined analogously to Definition 50 [Gir15b; PV17; Mar+16], but replacing invariance by translation with invariance by application of a *graph shift operator*:

Definition 52: Graph shift operator

A ***graph shift operator*** on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ of adjacency matrix $\mathbf{A}$ is a matrix $\mathbf{S}$ such that:

$$\forall v_1, v_2 \in \mathcal{V} : \mathbf{S}[v_1, v_2] \neq 0 \Leftrightarrow \mathbf{A}[v_1, v_2] \neq 0 \ . \tag{42}$$

Examples of such graph shift operators have already been encountered in this manuscript, in the form of the adjacency matrix, the Laplacian matrix, or its normalized version. Application of a graph shift operator $\mathbf{S}$ to a signal $\mathbf{x}_1$ to obtain a signal $\mathbf{x}_2$ is therefore done as follows:

$$\mathbf{x}_2 = \mathbf{S}\mathbf{x}_1 \ . \tag{43}$$

Intuitively, performing this computation replaces any entry v in $\mathbf{x}_2$ by a linear combination of the entries $\mathcal{N}(v)$ of $\mathbf{x}_2$. More generally, the various powers of a graph shift operator can be considered to model combination of values at more distant neighborhoods. For this reason, and by analogy with classical filtering, the corresponding matrix is often called *graph filter*:

Definition 53: Graph filter

A ***graph filter*** on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, for which we consider a graph shift operator $\mathbf{S}$, is a polynomial of $\mathbf{S}$:

$$\mathbf{H} = \sum_{i \in \mathcal{I} \subseteq \mathbb{Z}} C_i \mathbf{S}^i \ , \tag{44}$$

for some constant values C_i.

Note from Section 2.1.3 that all the powers of a given matrix share the same eigenvectors. Therefore, a graph filter has the same eigenvectors as the graph shift operator it is built from.

Using the definitions above, we define *stationarity* on a graph as follows:

Definition 54: Strong-sense stationary process on a graph

Using the definition in [Mar+16], a stochastic process on a graph is said to be ***strong-sense stationary*** if its joint probability distribution does not change when shifted in the graph, *i.e.*, if its moments are invariant by application of a graph shift operator on the graph.

Similarly to Definition 51, a ***weak-sense stationary process on a graph*** is defined by considering only its two first moments.

Again, ***white noise on a graph***, defined as a process of zero-mean and controlled variance, is a strong-sense stationary process. Also, for a graph filter $\mathbf{H}$ and a white noise signal $\mathbf{x}$, the signal $\mathbf{Hx}$ is stationary, since $\mathbf{H}$ is a polynomial of a graph shift operator.

Diffusion of signals

Diffusion of a signal on a graph is the process of propagating a signal across the edges of the graph. Numerous diffusion models can be considered, but all have in common to be representable by *graph shift operators*. A particular graph shift operator we are interested in is the one that represents the diffusion process measured by the normalized Laplacian matrix:

Definition 55: Diffusion matrix associated with the normalized Laplacian

Let us consider a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ of weights matrix $\mathbf{W}$ and degrees matrix $\mathbf{D}$. The ***diffusion matrix*** associated with the normalized Laplacian of this graph is:

$$\boldsymbol{\mathcal{T}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}} \ . \tag{45}$$

Note that this matrix differs from the normalized Laplacian in Definition 23 by an identity matrix. Therefore, it can be understood as a diffusion operator which impact is measured by the graph Laplacian. In more details, let us consider a signal $\mathbf{x}$ on the graph. We have the following equality:

$$\boldsymbol{\mathcal{L}}\mathbf{x} = \left(\mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}} \right) \mathbf{x} = \mathbf{I}_N \mathbf{x} - \boldsymbol{\mathcal{T}} \mathbf{x} \ . \tag{46}$$

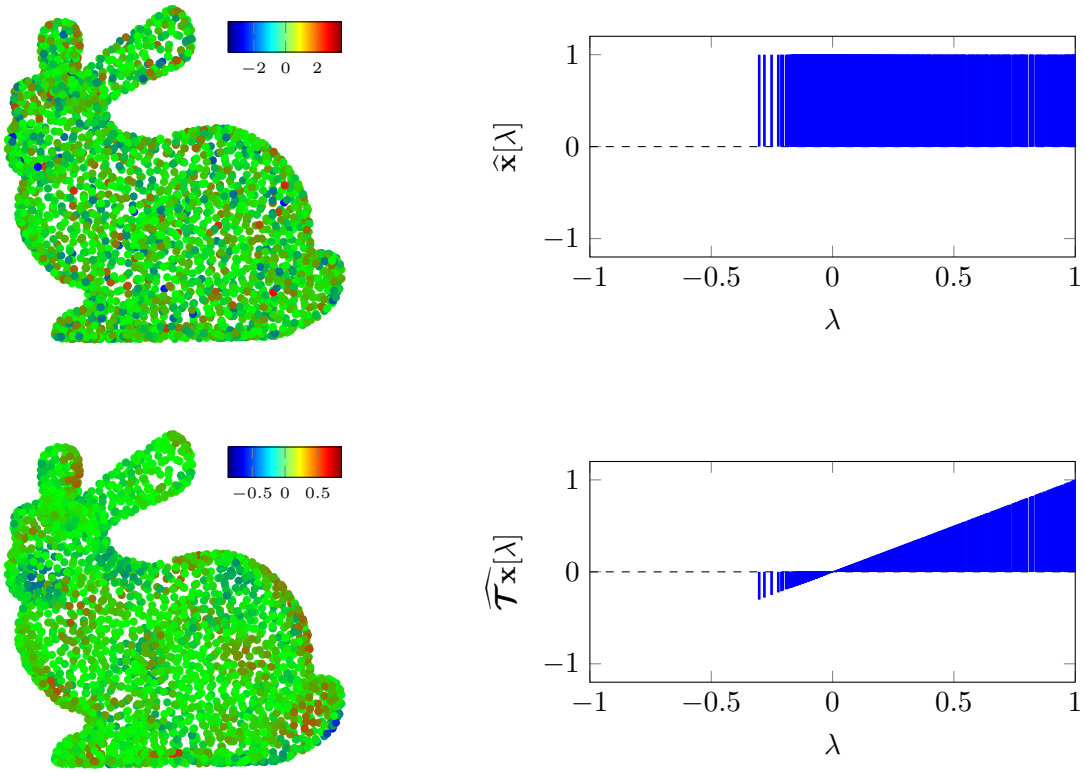
Considering the properties of the normalized Laplacian matrix introduced in Section 2.1.3, and noticing that the eigenvalues of $\mathbf{I}_N$ are all 1, we obtain the following properties:

- The eigenvalues of $\boldsymbol{\mathcal{T}}$ are in $[-1, 1]$, with -1 being an eigenvalue if and only if the graph is bipartite;
- The eigenvectors of $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{\mathcal{L}}$ are the same;
- The eigenvector associated with eigenvalue 0 of $\boldsymbol{\mathcal{L}}$ is also associated with eigenvalue 1 of $\boldsymbol{\mathcal{T}}$.

For these reasons — and considering a non-bipartite graph — diffusion of a signal n times using $\boldsymbol{\mathcal{T}}^n$ shrinks the contribution of the eigenvalues that are not 1. Therefore, as n grows to infinity, any signal eventually converges to a stable state corresponding to the eigenvector $\boldsymbol{\chi}$ associated with eigenvalue 1, which is equal to:

$$\forall i \in [\![1, N]\!] : \boldsymbol{\chi}[i] = \sqrt{\frac{\mathbf{D}[i,i]}{\sum_{j=1}^{N} \mathbf{D}[j,j]}}. \tag{47}$$

Since this eigenvector is associated with the lowest frequency of the graph Laplacian, diffusion of a signal with $\boldsymbol{\mathcal{T}}$ eventually converges to a highly regular signal on the graph. As a consequence, $\boldsymbol{\mathcal{T}}$ corresponds to the matrix that models heat propagation on the graph. Figure 22 illustrates this process on a graph corresponding to the 3D mesh of a bunny:



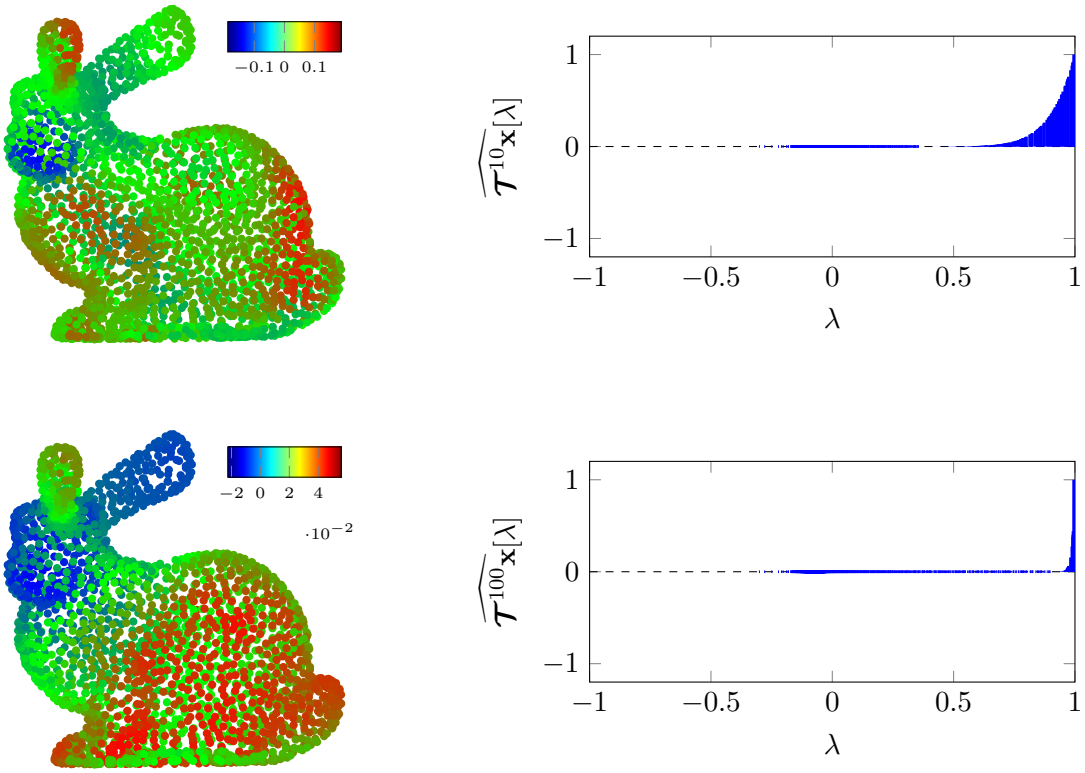


Figure 22 – Graph of a 3D mesh representing a bunny. For clarity, edges — linking neighboring angles of the mesh — are not represented. An initial signal, defined in the spectral domain such that every frequency contributes equally, is then diffused on the graph with $\mathcal{T}$, $\mathcal{T}^{10}$, and $\mathcal{T}^{100}$.

In the general case, diffusion of a signal does not eventually converge to a stable value, since the eigenvalues of the graph shift operator may not respect the criteria above. In particular, if one eigenvalue is larger than 1 in ***magnitude***, *i.e.*, in absolute value of its amplitude, then the signal will diverge. Still, as signals are diffused on the graph using a graph filter $\mathbf{H}$, the eigenvalue of $\mathbf{H}$ with the highest magnitude takes more importance relatively to the others. Therefore, $\frac{\mathbf{H}^n\mathbf{x}}{\|\mathbf{H}^n\mathbf{x}\|_2}$ eventually converges to the eigenvector associated with the eigenvalue of $\mathbf{H}$ with the highest magnitude, as n grows to infinity.

This observation is important for later work in this manuscript. It is interesting to notice that a random signal on a graph is not linked to its support at all. Contrary, once it is diffused upon convergence, it becomes the eigenvector of the graph filter used for diffusion associated with the eigenvalue of largest magnitude. For intermediary numbers of diffusions, the signal thus adapts progressively to the graph on which it is diffused, through the graph filter that is used.

2.3.3 The uncertainty principle on graphs

Heisenberg's uncertainty principle applied to signal on graphs

In classical signal processing, there is well-known result that states that a signal cannot be localized both in the time and in the spectral domain. This result is known as *Heisen-*

berg's uncertainty principle [Hei25]. Before defining this principle, we need to introduce the notion of *spread* of a signal, measuring how localized the signal is in the time or spectral domain:

Definition 56: Time spread of a signal

Let us consider a signal $s(t)$ of unitary ℓ_2 norm. The ***time spread*** around time instant $t_\bullet \in \mathbb{R}$, noted $\Delta_{t_\bullet}(s)$, is a local quantity measuring the localization of a signal in the time domain:

$$\Delta_{t_\bullet}(s) = \int_{\mathbb{R}} (t - t_\bullet)^2 |s(t)|^2 \, \mathrm{d}t \; . \tag{48}$$

From this quantity, the time spread of the signal, noted $\Delta_t(s)$, measures its localization in the time domain:

$$\Delta_t(s) = \min_{t_\bullet \in \mathbb{R}} \Delta_{t_\bullet}(s) \; . \tag{49}$$

The smaller this quantity, the more concentrated $s(t)$. The spectral version of the spread is defined similarly as follows:

Definition 57: Spectral spread of a signal

Let us consider a signal $s(t)$ of unitary ℓ_2 norm. The ***spectral spread*** around frequency $\omega_\bullet \in \mathbb{R}$, noted $\Delta_{\omega_\bullet}(s)$, is a local quantity measuring the localization of a signal in the spectral domain:

$$\Delta_{\omega_\bullet}(s) = \frac{1}{2\pi} \int_{\mathbb{R}} (\omega - \omega_\bullet)^2 |\widehat{s}(\omega)|^2 \, \mathrm{d}\omega \; . \tag{50}$$

From this quantity, the spectral spread of the signal, noted $\Delta_\omega(s)$, measures its localization in the spectral domain:

$$\Delta_\omega(s) = \min_{\omega_\bullet \in \mathbb{R}} \Delta_{\omega_\bullet}(s) \; . \tag{51}$$

Again, the smaller this quantity, the more concentrated $\widehat{s}(\omega)$. Using these two quantities, the uncertainty principle is given as follows:

Theorem 2: Uncertainty principle

For any unit-norm signal $s(t)$, the following inequality holds:

$$\Delta_t(s) \Delta_\omega(s) \geq \frac{1}{4} \; . \tag{52}$$

An extreme case for this uncertainty principle is the Dirac delta signal $\delta(t)$, which is completely localized in time. When considering its Fourier transform, we obtain the following development:

$$\widehat{\delta}(\omega) = \int_{\mathbb{R}} \delta(t) \, \mathrm{e}^{-i2\pi\omega t} \, \mathrm{d}t = 1 \; . \tag{53}$$

As a consequence, the most localized function in the time domain has a spectrum in which every frequency has amplitude 1.

In the general case, not all signals have the same value $\Delta_t(s)\Delta_\omega(s)$, and we can find some signals that minimize this product and are therefore Pareto optima in terms of concentration. Finding such signals has applications for example in analysis of phase synchrony of brain signals, in which interesting signals are localized in time and are bandlimited [Lac+99].

Porting this uncertainty principle to graph signal processing was done by Agaskar and Lu in [AL12; AL13]. By analogy with the definitions above, they define notions of *spreads* in the graph and in the spectral domains, as follows:

Definition 58: Graph spread of a signal on a graph

Let us consider a signal $\mathbf{x}$ of unitary ℓ_2 norm on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. The ***graph spread*** around vertex $v_\bullet \in \mathcal{V}$, noted $\Delta_{\mathcal{G},v}(\mathbf{x})$, is a local quantity measuring the localization of a signal in the graph domain:

$$\Delta_{\mathcal{G},v_\bullet}(\mathbf{x}) = \sum_{v \in \mathcal{V}} d_{\text{geo}}(v, v_\bullet)^2 |\mathbf{x}[v]|^2 \ . \tag{54}$$

From this quantity, the graph spread of the signal, noted $\Delta_{\mathcal{G},v}(\mathbf{x})$, measures its localization in the graph domain:

$$\Delta_{\mathcal{G},v}(\mathbf{x}) = \min_{v_\bullet \in \mathcal{V}} \Delta_{\mathcal{G},v_\bullet}(\mathbf{x}) \ . \tag{55}$$

Using this definition, if we note $\mathbf{P}_{v_\bullet}$ the diagonal matrix of geodesic distances to vertex $v_\bullet$, the graph spread can be rewritten as follows:

$$\Delta_{\mathcal{G},v}(\mathbf{x}) = \min_{v_\bullet \in \mathcal{V}} \mathbf{x}^\top \mathbf{P}_{v_\bullet}^2 \mathbf{x} \ . \tag{56}$$

Definition 59: Spectral spread of a signal on a graph

Let us consider a signal on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ with spectrum $\widehat{\mathbf{x}}$ of unitary ℓ_2 norm. The ***spectral spread*** around eigenvalue $\lambda_\bullet \in \boldsymbol{\lambda}$ of the normalized Laplacian matrix, noted $\Delta_{\mathcal{G},\lambda_\bullet}(\mathbf{x})$, is a local quantity measuring the localization of a signal in the spectral domain of the graph:

$$\Delta_{\mathcal{G},\lambda_\bullet}(\mathbf{x}) = \sum_{\lambda \in \boldsymbol{\lambda}} (\lambda - \lambda_\bullet)^2 |\widehat{\mathbf{x}}[\lambda]|^2 \ . \tag{57}$$

From this quantity, the spectral spread of the signal, noted $\Delta_{\mathcal{G},\lambda}(\mathbf{x})$, measures its localization in the spectral domain of the graph:

$$\Delta_{\mathcal{G},\lambda}(\mathbf{x}) = \min_{\lambda_\bullet \in \boldsymbol{\lambda}} \Delta_{\mathcal{G},\lambda_\bullet}(\mathbf{x}) \ . \tag{58}$$

Definition 59 is not exactly the spectral spread as introduced in [AL12]. In particular, noticing that all eigenvalues of the normalized Laplacian matrix are positive, and considering only the spread around eigenvalue 0 — by analogy with the symmetry of the spectrum in classical signal processing — the authors simplify the definition as follows:

$$\begin{aligned} \Delta_{\mathcal{G},\lambda}(\mathbf{x}) &= \sum_{\lambda \in \boldsymbol{\lambda}} \lambda |\widehat{\mathbf{x}}[\lambda]|^2 \\ &= \mathbf{x}^\top \boldsymbol{\mathcal{L}} \mathbf{x} \end{aligned} \ . \tag{59}$$

Using these definitions of spreads in the graph domain and in the spectral domain, Agaskar and Lu then propose a characterization of the *feasibility region*:

Definition 60: Feasibility region

The ***feasibility region*** $\mathcal{D}_{\mathcal{G},v_\bullet}$ of a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is defined as follows:

$$\mathcal{D}_{\mathcal{G},v_\bullet} = \{(g,s) \mid \Delta_{\mathcal{G},v_\bullet}(\mathbf{x}) = g \wedge \Delta_{\mathcal{G},\lambda}(\mathbf{x}) = s\} \,, \tag{60}$$

for any unit-norm signal $\mathbf{x}$ on the graph.

This region characterizes the possible compromises between graph spread and spectral spread. It has the following noticeable properties, for which proofs are given in [AL13]:

- $\mathcal{D}_{\mathcal{G},v_\bullet}$ is a closed subset of $[0,\lambda_N] \times \left[0, \max\limits_{v\in\mathcal{V}} d_{\text{geo}}(v_\bullet, v)\right]$;
- $\mathcal{D}_{\mathcal{G},v_\bullet}$ intersects the vertical axis at a single point $(1,0)$;
- $\mathcal{D}_{\mathcal{G},v_\bullet}$ intersects the horizontal axis at a single point $\left(0, \boldsymbol{\chi}_1^\top \mathbf{P}_{v_\bullet}^2 \boldsymbol{\chi}_1\right)$;
- The points $\left(1, \max\limits_{v\in\mathcal{V}} d_{\text{geo}}(v_\bullet, v)\right)$ and $\left(\lambda_N, \boldsymbol{\chi}_N^\top \mathbf{P}_{v_\bullet}^2 \boldsymbol{\chi}_N\right)$ belong to $\mathcal{D}_{\mathcal{G},v_\bullet}$;
- If $N \geq 3$, then $\mathcal{D}_{\mathcal{G},v_\bullet}$ is a convex set.

To summarize this properties, Figure 23 depicts the feasibility region of a graph:

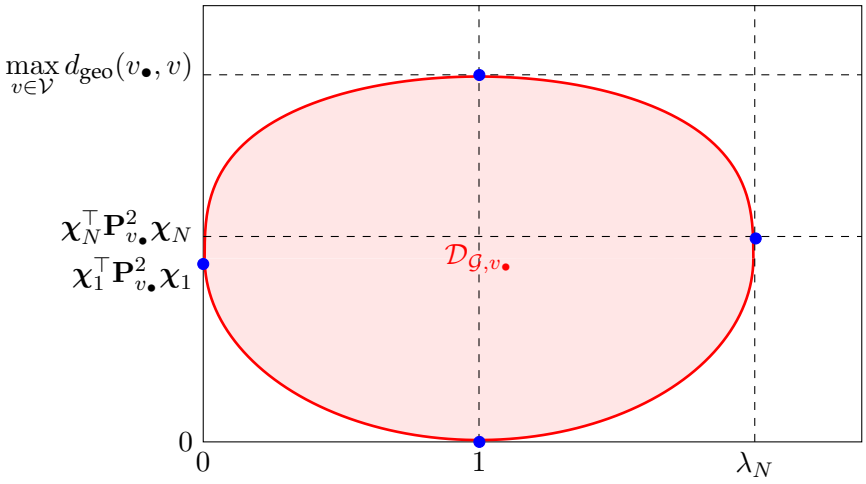


Figure 23 – Example of a feasibility region of a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, for a vertex $v_\bullet \in \mathcal{V}$. In this figure, eigenvectors and eigenvalues are those of the normalized Laplacian matrix. Every point of the region in red corresponds to the spreads of a signal, and therefore delimits $\mathcal{D}_{\mathcal{G},v_\bullet}$. The noticeable points of the domain are highlighted with blue dots.

Considering Figure 23, the points we are the most interested in are those that lie in the lower left part of the convex set, as they correspond to signals that are Pareto optima in terms of locality. This part of the curve is denoted *uncertainty curve*:

Definition 61: Uncertainty curve

The ***uncertainty curve*** $\gamma_{v_\bullet}(s)$, of a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, for a vertex $v_\bullet \in \mathcal{V}$, is:

$$\gamma_{v_\bullet}(s) = \min_{\substack{\mathbf{x}\in\mathbb{R}^N \\ \|\mathbf{x}\|_2=1}} \Delta_{\mathcal{G},v_\bullet}(\mathbf{x}) \;\; s.t. \;\; \Delta_{\mathcal{G},\lambda}(\mathbf{x}) = s \,. \tag{61}$$

Obviously, this curve depends on the graph topology, as well as on the vertex that is chosen for reference. For example, Figure 24 depicts the uncertainty curve we obtain for a complete graph, and for a star graph for which $v_\bullet$ is chosen to be the most connected vertex. In particular, it was shown by Rabbat and Gripon in [RG14] that this latter graph is the one that minimizes the graph spread for $s = 0$:

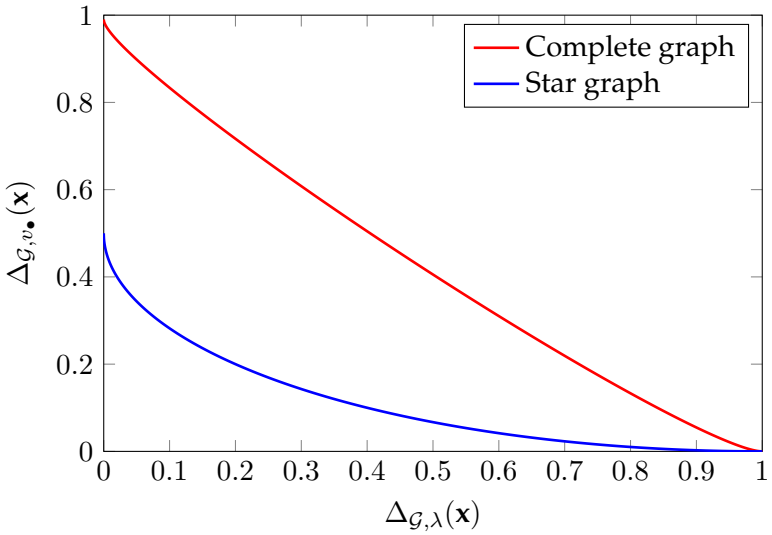


Figure 24 – Example of uncertainty curves for a complete graph, and for a star graph for which $v_\bullet$ is chosen to be the most connected vertex. Both graphs consist of $N = 100$ vertices. Interestingly, it is possible to show that the uncertainty curve of this latter graph is the same for every N, contrary to most graphs.

Considering the particular cases of the complete graph and star graph with $v_\bullet$ chosen as the most connected vertex, Agaskar and Lu have shown in [AL13] that for such configurations, signals $\mathbf{x}^*$ reaching the uncertainty curve are of the following form:

$$\mathbf{x}^* = \begin{bmatrix} \mathbf{x}^*[1] \\ \mathbf{x}^*[2] \\ \dots \\ \mathbf{x}^*[2] \end{bmatrix} . \tag{62}$$

Since we are only considering unit-norm signals, such Pareto optima can be translated to polar coordinates as follows:

$$\mathbf{x}^* = \begin{bmatrix} \cos(\theta) \\ \dfrac{\sin(\theta)}{\sqrt{N-1}} \\ \dots \\ \dfrac{\sin(\theta)}{\sqrt{N-1}} \end{bmatrix} , \tag{63}$$

for some parameter θ. As a consequence, by developing from the following equality:

$$\cos^2(\theta) + \sin^2(\theta) = 1 , \tag{64}$$

we obtain that the uncertainty curve for the complete graph is part of an ellipse defined as follows:

$$(2\Delta_{\mathcal{G},v_\bullet}(\mathbf{x}) - 1)^2 + (N-1)\left(\Delta_{\mathcal{G},\lambda}(\mathbf{x}) + \frac{N-2}{N-1}\Delta_{\mathcal{G},v_\bullet}(\mathbf{x}) - 1\right)^2 = 1 . \tag{65}$$

Similarly, for the star graph with $v_\bullet$ chosen as the most connected vertex, the uncertainty curve is part of the following ellipse:

$$(\Delta_{\mathcal{G},\lambda}(\mathbf{x}) - 1)^2 + (2\Delta_{\mathcal{G},v_\bullet}(\mathbf{x}) - 1)^2 = 1 \ . \tag{66}$$

For more complex graphs, Agaskar and Lu introduce an approximate algorithm, called the *sandwich algorithm*. More information on this can be found in [AL13].

To conclude with this section, it is worth noting that in the general case, an uncertainty principle is defined by some compromise between quantities defined in the graph domain, and in its spectrum. The uncertainty principle introduced by Agaskar and Lu is inspired by Heisenberg's results. Other definitions have been introduced, but are not studied in this manuscript. The interested reader may consult [TBDL16] or [TV16] for additional uncertainty principles.

Contribution: characterization of signals reaching the uncertainty curves

In the previous section, we introduced numerous definitions about the uncertainty principle, based on the work by Agaskar and Lu in [AL12; AL13]. Now, we review our contributions in relation with their work.

First, we have proposed in [Pas+16] a method to extend the class of graphs for which it is possible to formally characterize the signals that reach the uncertainty curve, in the manner of (65) or (66). To propose such a characterization, we extended the work of Agaskar and Lu in [AL13], Appendix C. This latter work was originally made to show the results in (62). In [Pas+16], we have shown the following result:

Proposition 1

Let us consider a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, and a vertex $v_\bullet \in \mathcal{V}$. Let $\mathbf{P}_{v_\bullet}$ be the diagonal matrix of geodesic distances to vertex $v_\bullet$, and let $\boldsymbol{\mathcal{L}}$ be the normalized Laplacian matrix of the graph. Let $\mathbf{M}(s) = \mathbf{P}_{v_\bullet}^2 - s\boldsymbol{\mathcal{L}}$ be a matrix defined for a fixed s.

If $\mathbf{M}(s)$ is of the form:

$$\mathbf{M}(s) = \left(\begin{array}{c|c} \mathbf{M}_1 & \mathbf{M}_2 \\ \hline \mathbf{M}_3 & \mathbf{M}_4 \end{array}\right) \ , \tag{67}$$

where:

- $\mathbf{M}_1$ is a $C \times C$ square matrix;
- $\mathbf{M}_2$ is a matrix that is constant by line, *i.e.*, $\mathbf{M}_2 = \mathbf{x}\mathbf{1}_C^\top$, for $\mathbf{x} \in \mathbb{R}^C$, and for $\mathbf{1}_C$ a vector of C entries all equal to 1;
- $\mathbf{M}_3$ is any matrix;
- $\mathbf{M}_4$ is an $(N - C) \times (N - C)$ circulant matrix.

then any signal $\mathbf{x}^*$ reaching the uncertainty curve is of the form:

$$\mathbf{x}^* = \begin{bmatrix} \mathbf{x}^*[1] \\ \dots \\ \mathbf{x}^*[C] \\ \mathbf{x}^*[C+1] \\ \dots \\ \mathbf{x}^*[C+1] \end{bmatrix} . \tag{68}$$

For the need of the proof, we recall the following result from [AL13], Proposition 2:

[AL13], Proposition 2

Every signal $\mathbf{x}^*$ reaching uncertainty curve is the eigenvector associated with the lowest eigenvalue of a matrix $\mathbf{M}(s) = \mathbf{P}_{v_\bullet}^2 - s\boldsymbol{\mathcal{L}}$.

Proof: Proposition 1

Since $\mathbf{M}_4$ is circulant, we have that $\mathbf{1}_{N-C}$ is an eigenvector for $\mathbf{M}_4$. By construction of $\mathbf{M}(s)$, $\mathbf{M}_4$ is symmetric, and can thus be diagonalized. Let $(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{N-C-1})$ be the eigenvectors of $\mathbf{M}_4$ orthogonal to $\mathbf{1}_{N-C}$, associated with the eigenvalues $(\lambda_1, \ldots, \lambda_{N-C-1})$. By construction, we have $\forall i \in [\![1, N-C-1]\!] : \boldsymbol{\chi}_i^\top \mathbf{1}_{N-C} = 0$.

For all $i \in [\![1, N-C-1]\!]$, we build a vector as follows:

$$\boldsymbol{\chi}_i^+ = \begin{bmatrix} \mathbf{0}_C \\ \boldsymbol{\chi}_i \end{bmatrix}, \tag{69}$$

for $\mathbf{0}_C$ a vector of C entries all equal to 0. Using the fact that $\mathbf{M}_2$ is constant by line, we obtain that $\forall i : \mathbf{M}(s)\boldsymbol{\chi}_i^+ = \lambda_i \boldsymbol{\chi}_i^+$. Therefore, every $\boldsymbol{\chi}_i^+$ is an eigenvector of $\mathbf{M}(s)$. Using the methodology of [AL13], appendix C (Rayleigh inequality), we obtain that the eigenvector associated with the smallest eigenvalue of $\mathbf{M}(s)$ must be orthogonal to $\boldsymbol{\chi}_i^+$ for all $i \in [\![1, N-C-1]\!]$.

By noting that the C first entries of vectors $\boldsymbol{\chi}_i^+$ are null, and by application of [AL13], Proposition 2, we obtain that every signal $\mathbf{x}^*$ reaching the uncertainty curve is of the form in (68).

It is interesting to remark that the application of Proposition 1 can be made recursively on $\mathbf{M}_1$, allowing one to refine the characterization of the entries $\mathbf{x}^*[1] \ldots \mathbf{x}^*[C]$ of signals that reach the uncertainty curve, and thus to reduce the search space of solutions.

To illustrate the characterization of signals $\mathbf{x}$ reaching the uncertainty curve, let us consider a star graph, but this time with $v_\bullet$ taken as one of the least connected vertices of the graph. We obtain that $\mathbf{M}(s)$ can be decomposed in a way such that an $(N-2) \times (N-2)$ circulant submatrix $\mathbf{M}_4$ appears. Using Proposition 1, we obtain that, for this graph and this choice of $v_\bullet$, all signals $\mathbf{x}^*$ are of the following form:

$$\mathbf{x}^* = \begin{bmatrix} \mathbf{x}^*[1] \\ \mathbf{x}^*[2] \\ \mathbf{x}^*[3] \\ \ldots \\ \mathbf{x}^*[3] \end{bmatrix}. \tag{70}$$

In order to iterate over the solution signals $\mathbf{x}^*$, we can consider the set of unit-norm signals defined on an hypersphere of dimension equal to the number of distinct entries in the characterization of $\mathbf{x}^*$. In the considered example, we can thus reduce the set of

potential solutions to signals characterized by two parameters θ and ϕ as follows:

$$\mathbf{x}^* = \begin{bmatrix} \cos(\theta) \\ \sin(\theta)\cos(\phi) \\ \frac{\sin(\theta)\sin(\phi)}{\sqrt{N-2}} \\ \dots \\ \frac{\sin(\theta)\sin(\phi)}{\sqrt{N-2}} \end{bmatrix}. \tag{71}$$

Figure 25 represents the pairs $(\Delta_{\mathcal{G},v_\bullet}(\mathbf{x}), \Delta_{\mathcal{G},\lambda}(\mathbf{x}))$ for signals $\mathbf{x}$ obtained by iterating over possible values of θ and ϕ in the interval $[0, 2\pi]$, with a step of 0.05. We also depict the uncertainty curves obtained using the sandwich algorithm, and observe that they match the frontier of the set of explored signals:

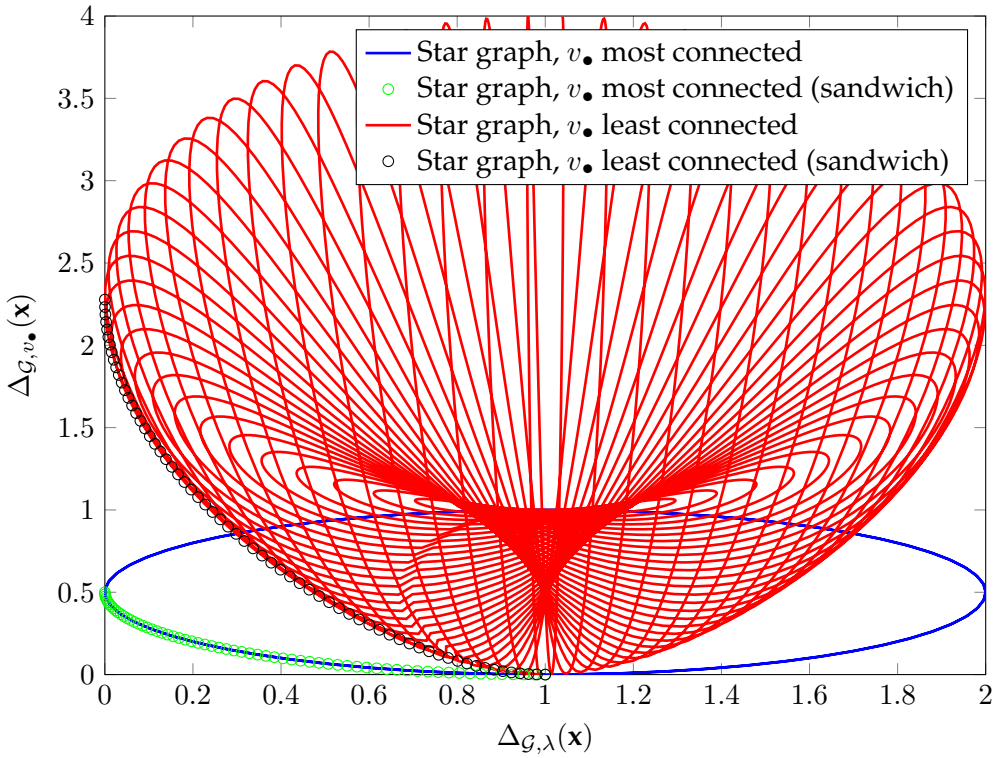


Figure 25 – Sub-sampling of the space of potential solutions for a star graph, for different choices of $v_\bullet$. The lower left frontier of the set of explored signals is the uncertainty curve. Approximation through the sandwich algorithm matches the obtained results.

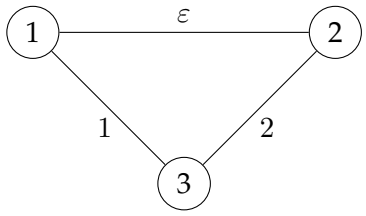
Since the signals verifying the uncertainty curve are part of an hypersphere, it is interesting to notice that Figure 25 is a 3D plot, for which we have a 2D interpretation.

Contribution: extending the uncertainty principle to weighted graphs

The uncertainty principle introduced by Agaskar and Lu in [AL12] was defined for unweighted graphs only. In [Pas+15b], we have proposed an extension of their definition to weighted graphs.

While it is possible to compute a graph spread in the manner of Definition 58, and a spectral spread as in Definition 59, thus allowing the computation of an uncertainty

curve as in Definition 61, we have shown that a simple application of theses definitions leads to a discontinuity problem. To illustrate the problem, let us consider the weighted graph $\mathcal{G}_1 = \langle \mathcal{V}_1, \mathcal{E}_1, f_1 \rangle$ in Figure 26:



$$\mathbf{W} = \begin{array}{c} \\ \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \end{array} \begin{array}{c} \begin{array}{ccc} \textcircled{1} & \textcircled{2} & \textcircled{3} \end{array} \\ \begin{bmatrix} 0 & \varepsilon & 1 \\ \varepsilon & 0 & 2 \\ 1 & 2 & 0 \end{bmatrix} \end{array}$$

Figure 26 – Example of a weighted graph (left), and associated weights matrix (right). In this graph, ε represents a very small weight.

Let us choose $v_\bullet = \textcircled{1}$. We obtain the following matrix of geodesic distances to $v_\bullet$:

$$\mathbf{P}_{v_\bullet} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{72}$$

Additionally, the normalized Laplacian matrix of this graph is given as follows:

$$\boldsymbol{\mathcal{L}} = \begin{bmatrix} 1 & -\frac{\varepsilon}{\sqrt{(1+\varepsilon)(2+\varepsilon)}} & -\frac{1}{\sqrt{(1+\varepsilon)3}} \\ -\frac{\varepsilon}{\sqrt{(2+\varepsilon)(1+\varepsilon)}} & 1 & -\frac{2}{\sqrt{(2+\varepsilon)3}} \\ -\frac{1}{\sqrt{3(1+\varepsilon)}} & -\frac{2}{\sqrt{3(2+\varepsilon)}} & 1 \end{bmatrix} . \tag{73}$$

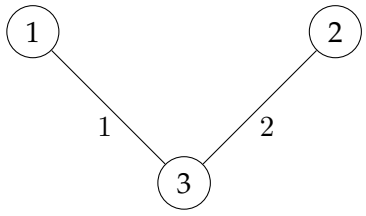
Let us consider a unit norm signal $\mathbf{x}$ such that every entry equals $\frac{1}{\sqrt{3}}$. Using Definition 58 and Definition 59, we compute the graph and spectral spreads of this signal on the graph:

- $\Delta_{\mathcal{G},v_\bullet}(\mathbf{x}) = \mathbf{x}^\top \mathbf{P}_{v_\bullet}^2 \mathbf{x} = \frac{\varepsilon^2+1}{3}$;
- $\Delta_{\mathcal{G},\lambda}(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\mathcal{L}} \mathbf{x} = \frac{1}{3}\left(3 - \frac{2\varepsilon}{\sqrt{(1+\varepsilon)(2+\varepsilon)}} - \frac{2}{\sqrt{3(1+\varepsilon)}} - \frac{4}{\sqrt{3(2+\varepsilon)}}\right)$.

It seems reasonable to expect that as ε tends to 0, both spreads tend to the limit case where $\varepsilon = 0$. In particular, these quantities should be robust to measurement noise in scenarios where the weights matrix is not known. Let us now compute these limits:

- $\Delta_{\mathcal{G},v_\bullet}(\mathbf{x}) = \frac{\varepsilon^2+1}{3} \underset{\varepsilon\to 0}{\longrightarrow} \frac{1}{3}$;
- $\Delta_{\mathcal{G},\lambda}(\mathbf{x}) = \frac{1}{3}\left(3 - \frac{2\varepsilon}{\sqrt{(1+\varepsilon)(2+\varepsilon)}} - \frac{2}{\sqrt{3(1+\varepsilon)}} - \frac{4}{\sqrt{3(2+\varepsilon)}}\right) \underset{\varepsilon\to 0}{\longrightarrow} \frac{1}{3}\left(3 - \frac{2}{\sqrt{3}} - \frac{4}{\sqrt{6}}\right)$.

Let us now consider the weighted graph $\mathcal{G}_2 = \langle \mathcal{V}_2, \mathcal{E}_2, f_2 \rangle$ associated with the limit case, given in Figure 27:



$$\mathbf{W} = \begin{array}{c} \\ \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \end{array} \begin{array}{c} \begin{array}{ccc} \textcircled{1} & \textcircled{2} & \textcircled{3} \end{array} \\ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 2 \\ 1 & 2 & 0 \end{bmatrix} \end{array}$$

Figure 27 – Weighted graph (left) — and associated weights matrix (right) — corresponding to the limit of $\mathcal{G}_1$ in Figure 26 as ε tends to 0.

Now, for $v_\bullet = \textcircled{1}$, we obtain the following matrix of geodesic distances to $v_\bullet$:

$$\mathbf{P}_{v_\bullet} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{74}$$

Additionally, the normalized Laplacian matrix of this graph is given as follows:

$$\boldsymbol{\mathcal{L}} = \begin{bmatrix} 1 & 0 & -\frac{1}{\sqrt{3}} \\ 0 & 1 & -\frac{2}{\sqrt{6}} \\ -\frac{1}{\sqrt{3}} & -\frac{2}{\sqrt{6}} & 1 \end{bmatrix} . \tag{75}$$

Considering the same signal $\mathbf{x}$ as before, and using Definition 58 and Definition 59, we obtain the following spreads:

- $\Delta_{\mathcal{G},v_\bullet}(\mathbf{x}) = \mathbf{x}^\top \mathbf{P}_{v_\bullet}^2 \mathbf{x} = \frac{10}{3}$;
- $\Delta_{\mathcal{G},\lambda}(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\mathcal{L}} \mathbf{x} = \frac{1}{3}\left(3 - \frac{2}{\sqrt{3}} - \frac{4}{\sqrt{6}}\right)$.

While the spectral spread obtained for $\mathcal{G}_2$ corresponds to the one computed for $\mathcal{G}_1$ as ε tends to 0, we notice that it is not the case for the graph spread.

As a matter of fact, existence of an edge is represented in the weights matrix of a graph with a non-null entry at the corresponding row and column. Contrary, absence of an edge corresponds to the associated entry being zero. It follows that reducing the weight of an edge to 0 can cause a discontinuity, as the shortest path from a vertex to $v_\bullet$ may then change.

To highlight the cause problem, let us consider the equation of smoothness in Definition 49. For $\mathrm{S}(\mathbf{x})$ to be low, vertices that are linked with an edge associated with a strong weight must share similar signal entries. It follows that the weights matrix encodes information about *similarity* between vertices, rather than *distance* between them. Using the geodesic distance in the computation of Definition 58 then makes a misuse of this information if used on a weighted graph.

To correct this discontinuity problem, we introduced in [Pas+15b] an alternate definition for the graph spread. Let us consider a weighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, f \rangle$, of weights matrix $\mathbf{W}$. We want a measure that respects the following properties:

1. $\Delta_{\mathcal{G},v_\bullet}(\mathbf{x})$ should be small if $\mathbf{x}$ is localized around $v_\bullet$, and should increase as the distance between $v_\bullet$ and the vertices carrying $\mathbf{x}$ increases;
2. The only situation leading to $\Delta_{\mathcal{G},v_\bullet}(\mathbf{x}) = 0$ should be when the signal has only one non-null entry, located on $v_\bullet$;
3. The graph spread should be *similar* for graphs with *similar* weights, *i.e.*, $\mathcal{G} \mapsto \Delta_{\mathcal{G},v_\bullet}(\mathbf{x})$ should be a continuous function.

Proposition 2

In order to be compliant with the above desirable properties, the acceptable functions d are as follows:

1. $\forall v_1, v_2 \in \mathcal{V} : d(v_1, v_2) \geq 0$;
2. $\forall v_1, v_2 \in \mathcal{V} : d(v_1, v_2) = 0 \Leftrightarrow v_1 = v_2$;
3. d is continuous, and if we increase $f(\{v_1, v_2\})$ for a single edge $\{v_1, v_2\} \in \mathcal{E}$, then, $\forall v_3, v_4 \in \mathcal{V} : f(\{v_3, v_4\})$ does not increase.

It is interesting to remark that the squared geodesic distance used in Definition 58 does not respect the third property in Proposition 2.

Numerous functions can be used to replace it. In particular, we propose to replace the computation of squared geodesic distances on $\mathbf{W}$, by computation of squared geodesic distances on the *inverse weights matrix*, defined as follows:

Definition 62: Inverse weights matrix

The ***inverse weights matrix*** $\overline{\mathbf{W}}$ of a weighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, f \rangle$ of weights matrix $\mathbf{W}$ is defined as follows:

$$\forall v_1, v_2 \in \mathcal{V} : \overline{\mathbf{W}}[v_1, v_2] = \begin{cases} \infty & \textbf{if } \mathbf{W}[v_1, v_2] = 0 \\ 0 & \textbf{if } v_1 = v_2 \\ \frac{1}{\mathbf{W}[v_1, v_2]} & \textbf{otherwise} \end{cases} . \tag{76}$$

Using $\overline{\mathbf{W}}$, the ***inverse geodesic distance***, noted $d_{\overline{\text{geo}}}$, is the geodesic distance on a graph of weights matrix $\overline{\mathbf{W}}$:

Proposition 3

The squared inverse geodesic distance $d^2_{\overline{\text{geo}}}$ is compliant with the properties in Proposition 2.

Proof: Proposition 3

Let us consider the three properties in Proposition 3 in the same order:

1. Since we consider the squared inverse geodesic distance, this property is trivially true;
2. Following Remark 4 that we consider in this manuscript graphs with positive weights, this second property is enforced by construction of $\overline{\mathbf{W}}$, since no path between two distinct vertices can have length 0;
3. Let $\overline{\mathcal{G}} = \langle \mathcal{V}, \overline{\mathcal{E}}, \overline{f} \rangle$ be the weighted graph of weights matrix $\overline{\mathbf{W}}$. Let $p_1 \in \mathcal{P}(\overline{\mathcal{G}})$ be a path between two vertices of $\mathcal{V}$, and let $\{v_1, v_2\} \in \overline{\mathcal{E}}$ be an edge along p_1. If we increase $f(\{v_1, v_2\})$, then by construction of $\overline{\mathbf{W}}$, we obtain that $\overline{f}(\{v_1, v_2\})$ decreases, and that all other weights remain the same. Note that this is also the case if increasing $f(\{v_1, v_2\})$, leads to the creation of an edge in the graph. As a consequence, if we increase a weight in $\mathbf{W}$, then $\forall v_3, v_4 \in \mathcal{V} : d^2_{\overline{\text{geo}}}(v_3, v_4)$ does not increase.

Additional functions such that the *diffusion distance* [SHR15], the *resistance distance* [KR93] or the *commute distance* [AM12] can also be used, as well as replacing the inverse in Definition 62 with any decreasing function. However, we will not enter into the details of these alternate solutions. Experiments using the diffusion distance however can be found in [Pas+15b].

2.4 Summary of the chapter

In this chapter, we have introduced numerous definitions, that are necessary for a complete understanding of our work. Still, additional related work will be provided in the next chapters, to highlight contributions on more particular subdomains of graph signal processing, namely graph inference, and translations on graphs.

The key points of this chapter are:

- The definitions of graphs, their matrix representations, and the notion of paths on such structures. We remind that in the rest of this manuscript, we only consider connected, positively weighted graphs with no edge duplicates;
- The matrices that can be computed from graphs, and their properties. In particular, graph shift operators such as the Laplacian matrix will be of interest all along this document;
- The link between the graph domain, and the spectral domain associated with the eigenvectors of one of its graph shift operators. This link is the cornerstone from which graph signal processing has developed;
- The links between the graph and the signals that are defined on it. In particular, notions such as diffusion, smoothness and stationarity of signals will be of broad interest in the next chapter.

Chapter 3

From signals to graphs

Contents

As we have shown in the previous chapter, many tools from graph signal processing exploit knowledge of the underlying graph structure. In particular, the graph Fourier transform is a key tool to provide a spectral representation for signals defined on graphs, which requires obtainment of the eigenvectors of a graph shift operator (generally the Laplacian matrix or its normalized version).

In the case when no graph is available, graph signal processing tools cannot be used anymore. As a matter of fact, this situation arises in many cases. While modeling time with a ring graph, or images with a grid graph is intuitive due to the underlying metric space, choosing a correct topology to model the domain on which brain signals evolve is not trivial. Numerous other situations can be found in which vertices are known, but not the edges linking them. Examples of such situations include seismic sensors randomly placed on the ground, or weather stations making temperature measurements at different locations of a country, and even more abstract domains such as social networks or word embeddings.

To circumvent this problem, researchers have proposed approaches to infer a graph topology from observations of signals on its vertices. Since the problem is ill-posed, these approaches make assumptions, such as smoothness of the signals on the graph, or sparsity of the graph [LT10; Don+16; Kal16].

The contents of this chapter is mainly based on two contributions [Pas+15a; Pas+17a]. We focus here on graph inference from signals that are stationary, as introduced in Section 2.3.2. To simplify exposition, we choose to focus on diffusion of white noise on the graph using a graph filter, which is a particular case of stationary processes.

This chapter is organized as follows:

1. First, we present the problem we consider in details, and introduce related work on graph inference from signals;
2. Second, we study the particular case where the graph filter used for diffusion is a monomial (of known power) of a graph shift operator. We show that recovery of the missing eigenvalues is possible with some assumptions;
3. Third, we consider the general case where the graph filter is completely unknown, and can be any power of a graph shift operator. We propose a characterization of the set of solutions, and illustrate selection of a solution based on a sparsity prior;
4. Finally, we propose a method to correct the solutions provided by existing graph inference strategies, so that they can be used to infer graph from stationary signals. We illustrate how this method can be used to select a graph that is most adapted to stationary signals, among a set of candidate graphs.

3.1 Problem formulation and related work

As introduced in the previous lines, inferring a graph from signals is a possible approach to obtain a graph, and thus enable use of the graph signal processing tools on such signals. Let us consider the case of observed signals of N dimensions — *i.e.*, vectors $\mathbf{x} \in \mathbb{R}^N$ — corresponding to values observed on N elements of interest. From these signals, we know that the corresponding graph has N vertices. Then, since edges between these vertices are unknown, we have to choose an edge configuration among the $2^{\frac{N(N-1)}{2}}$ possible configurations, when restricting the search to simple, unweighted, undirected graphs.

Since the number of potential solutions is huge, the problem is ill-posed. Therefore, it is necessary to add some priors on the inferred graph or on the signals. Here is a non-exhaustive list of such priors:

- Priors on the inferred graph: sparsity, simplicity, correspondence of the graph with a partially known ground truth...;
- Priors on the signals: smoothness on the graph, stationarity, assumption that the signals were Dirac vectors before diffusion on the graph...

3.1.1 Problem formulation

In this work, we choose to study the case of stationary signals, as defined in Section 2.3.2. Since stationarity of signals is invariant by application of a graph shift operator, it is also invariant by application of a polynomial of this operator. Therefore, diffusion of white noise through a graph filter is a particular case of stationary processes. To simplify exposition, we focus on these settings. Still, note that all the results introduced in this chapter apply to the more generic class of stationary processes.

In this chapter, we are interested in answering the following question:

Problem 2: Graph inference from stationary signals

Given a set of stationary signals observed after diffusion on a graph, how can one characterize the graph shift operator used for the diffusion, hence the graph?

In more details, we choose to focus on graph shift operators such that edges are associated with a positive weight only (see Remark 4). A possible graph shift operator corresponding to these settings is the diffusion matrix associated with the normalized Laplacian, introduced in Definition 55. Again, note that this choice aims at simplifying exposition, and that the results presented in this chapter apply for any graph shift operator, possibly with some adaptations as explained later in this chapter.

More formally, we study the following settings:

- Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a graph of N vertices, and let $\boldsymbol{\mathcal{T}}$ be the diffusion matrix associated with the normalized Laplacian of $\mathcal{G}$;
- We are given a sequence of M signals $(\mathbf{x}_1, \dots, \mathbf{x}_M)$ in $\mathbb{R}^N$. Without loss of generality, we consider signals that are zero-mean and of unit norm;
- Let $(\mathbf{H}_1, \dots, \mathbf{H}_M)$ be a sequence of M graph filters, each a polynomial of $\boldsymbol{\mathcal{T}}$;
- Let $(\mathbf{y}_1, \dots, \mathbf{y}_M)$ be M realizations of a zero-mean white noise process;
- We assume that the observed signals $(\mathbf{x}_1, \dots, \mathbf{x}_M)$ were observed after diffusion of signals $(\mathbf{y}_1, \dots, \mathbf{y}_M)$ using these graph filters, *i.e.*, we suppose the following:

$$\forall i \in [\![1, M]\!] : \mathbf{x}_i = \mathbf{H}_i \mathbf{y}_i \; ; \tag{77}$$

- Then, from the knowledge of $(\mathbf{x}_1, \dots, \mathbf{x}_M)$, and assuming that these signals are issued from the diffusion of *i.i.d.* signals on a graph, can we find the graph shift operator $\boldsymbol{\mathcal{T}}$ used to diffuse the signals?

3.1.2 The covariance matrix and its estimates

The covariance matrix

Before introducing the methods from the literature to infer a graph from signals, we first need to introduce a particular matrix, called the *covariance matrix*, as numerous graph inference techniques make use of it:

Definition 63: Covariance matrix

Let $\mathbf{X}$ be a vector of N random variables. The ***covariance matrix*** of these variables, noted $\mathbf{\Sigma}$, measures the pairwise covariances between these random variables, as follows:

$$\mathbf{\Sigma} = \mathbb{E}\left[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^\top\right]. \tag{78}$$

The covariance matrix is symmetric. When considering zero-mean signals — which is the case of diffused zero-mean white noise, since the mean is invariant by application of the graph filter — the equality in Definition 63 simplifies to:

$$\mathbf{\Sigma} = \mathbb{E}\left[\mathbf{X}\mathbf{X}^\top\right]. \tag{79}$$

It is interesting to note that the covariance matrix of *i.i.d.* white noise is the identity matrix. Using the settings introduced in Section 3.1.1, we make this development:

$$\begin{aligned}
\mathbf{\Sigma} &= \mathbb{E}\left[\mathbf{X}\mathbf{X}^\top\right] \\
&= (\mathbb{E}[\mathbf{H}]\,\mathbb{E}[\mathbf{Y}])(\mathbb{E}[\mathbf{H}]\,\mathbb{E}[\mathbf{Y}])^\top && \text{// Using the fact that } \forall i \in [\![1, M]\!] : \mathbf{x}_i = \mathbf{H}_i\mathbf{y}_i \\
&= \mathbb{E}[\mathbf{H}]\,\mathbb{E}[\mathbf{Y}]\,\mathbb{E}\left[\mathbf{Y}^\top\right]\,\mathbb{E}\left[\mathbf{H}^\top\right] \\
&= \mathbb{E}[\mathbf{H}]\,\mathbb{E}\left[\mathbf{Y}\mathbf{Y}^\top\right]\,\mathbb{E}\left[\mathbf{H}^\top\right] \\
&= \mathbb{E}[\mathbf{H}]\,\mathbf{I}_N\,\mathbb{E}\left[\mathbf{H}^\top\right] && \text{// Using the fact that } \mathbb{E}\left[\mathbf{Y}\mathbf{Y}^\top\right] = \mathbf{I}_N \\
&= \mathbb{E}\left[\mathbf{H}^2\right] && \text{// Using the symmetry of } \mathbf{H}
\end{aligned} \tag{80}$$

This development tells us that there is a strong link between the graph filters used for diffusion, and the covariance matrix. More precisely, we have the following result:

Proposition 4

The eigenvectors of the covariance matrix are exactly the eigenvectors of the graph shift operator used for diffusion.

Proof: Proposition 2

All graph filters are polynomials of the same variable, which is the graph shift operator used for diffusion. Therefore, any graph filter has the same eigenvectors as this graph shift operator (see Section 2.1.3).

Since $\mathbf{\Sigma} = \mathbb{E}\left[\mathbf{H}^2\right]$, we conclude that the covariance matrix has the same eigenvectors as the graph shift operator used for diffusion.

This result is very strong, as it reduces the candidate graphs to represent stationary signals from $\mathcal{O}(N^2)$ variables (the whole matrix) to N variables (the missing eigenvalues). In the rest of this chapter, we will make extensive use of this result. This means that obtainment of the covariance matrix is a cornerstone of our approach.

The sample covariance matrix

Since the covariance matrix is not obtainable in practical cases, a common approach involves estimating $\mathbf{\Sigma}$ using the *sample covariance matrix* $\widetilde{\mathbf{\Sigma}}$:

Definition 64: Sample covariance matrix

Let $\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ be a set of M signals in $\mathbb{R}^N$, given in matrix form such that every signal is a column of an $N \times M$ matrix $\mathbf{X}$. The ***sample covariance matrix*** of these signals, noted $\widetilde{\mathbf{\Sigma}}$, approximates the covariance matrix as follows:

$$\widetilde{\mathbf{\Sigma}} = \frac{1}{M-1}(\mathbf{X} - \mathbf{M})(\mathbf{X} - \mathbf{M})^\top \ , \tag{81}$$

where $\mathbf{M}$ is an $N \times M$ matrix with each row containing the mean signal value for the associated vertex:

$$\forall i \in [\![1, N]\!], \forall j \in [\![1, M]\!] : \mathbf{M}[i, j] = \frac{1}{M} \sum_{k=1}^{M} \mathbf{X}[i, k] \ . \tag{82}$$

Again, considering signals obtained by diffusion of zero-mean white noise, we simplify this expression as follows:

$$\widetilde{\mathbf{\Sigma}} = \frac{1}{M-1}\mathbf{X}\mathbf{X}^\top \ . \tag{83}$$

Finally, remember from Section 2.1.3 that $\frac{1}{M-1}$ distributes on the eigenvalues of $\mathbf{X}\mathbf{X}^\top$. Therefore, we can drop the constant when interested in the eigenvectors of the sample covariance only.

Again, we could develop the expression of $\widetilde{\mathbf{\Sigma}}$ as in (80) to show that as M grows to infinity, the sample covariance tends to a function of the graph filters used for diffusion [Pas+17a]. However, we are more interested in the link between the eigenvectors of the sample covariance matrix, and those of the graph shift operator. In more details, we study the convergence of the eigenvectors of the sample covariance matrix $\widetilde{\boldsymbol{\mathcal{X}}_{\mathbf{\Sigma}}} = (\widetilde{\boldsymbol{\chi}_1}, \dots, \widetilde{\boldsymbol{\chi}_N})$ to the eigenvectors of the covariance matrix $\boldsymbol{\mathcal{X}}_{\mathbf{\Sigma}} = (\boldsymbol{\chi}_1, \dots, \boldsymbol{\chi}_N)$, as M grows to infinity.

Asymptotic results on this convergence are provided by Anderson [And63], which extends earlier related results by Girshick [Gir39] and Lawley [Law56]. Let $\mathbf{e}_i = \boldsymbol{\mathcal{X}}_{\mathbf{\Sigma}}{}^\top \widetilde{\boldsymbol{\chi}_i}$ be the vector of cosine similarities between $\widetilde{\boldsymbol{\chi}_i}$ and all eigenvectors of the covariance matrix. Anderson [And63] states that, as the number of observations tends to infinity, entries in $\mathbf{e}_i$ have a Gaussian distribution with a known variance. In particular, when all eigenvalues are distinct, the inner product between $\boldsymbol{\chi}_i$ (for all i) and $\widetilde{\boldsymbol{\chi}_j}$ (for all j) is asymptotically Gaussian with zero mean and variance:

$$\frac{\lambda_i \widetilde{\lambda_j}}{(\lambda_i - \widetilde{\lambda_j})^2} \ , \lambda_i \neq \lambda_j \ , \tag{84}$$

where λ_i is the eigenvalue associated with $\boldsymbol{\chi}_i$, and $\widetilde{\lambda}_j$ is the eigenvalue associated with $\widetilde{\boldsymbol{\chi}_j}$. Therefore, the variance depends on the squared difference between λ_i and $\widetilde{\lambda}_j$. Additionally, Anderson [And63] shows that the maximum likelihood estimate $\widetilde{\lambda_i^*}$ of λ_i (for all i) is:

$$\widetilde{\lambda_i^*} = \frac{M-1}{M} \frac{1}{\#(\lambda_i)} \sum_{j=i}^{i+\#(\lambda_i)} \widetilde{\lambda}_j \ , \tag{85}$$

where $\#(\lambda_i)$ is the multiplicity of eigenvalue λ_i. In the simple case when all eigenvalues are distinct, (85) simplifies to:

$$\widetilde{\lambda_i^*} = \frac{M-1}{M} \widetilde{\lambda}_i \ . \tag{86}$$

Additionally, Anderson [And63] provides a similar result for the more general case when eigenvalues may be repeated. The eigenvalues of the sample covariance matrix thus converge to the eigenvalues of the covariance matrix as M increases. As M tends to infinity, (84) tells us that $\mathbf{e}_i$ thus tends to the i^{th} canonical vector, indicating collinearity between $\widetilde{\boldsymbol{\chi}_i}$ and $\boldsymbol{\chi}_i$.

Other covariance estimators

The sample covariance matrix is not the only estimator of the covariance matrix.

Among other estimators, some methods for retrieving a sparse covariance matrix based on properties of its spectral norm are described in [CL11] and [CZ12]. However, these works do not provide any information on the convergence rate of the eigenvectors of their solutions to the eigenvectors of the covariance matrix.

Similarly, [WP09] and [XW+12] retrieve covariance matrices that converge in operator norm or in distribution. An intensive study of covariance estimation methods could be interesting to find techniques that improve the convergence of eigenvectors.

This paper focuses on the use of the sample covariance matrix.

3.1.3 Related work

Numerous solutions to infer a graph from signals have been proposed in the literature. In this section, we review the methods that can possibly have a connection with our stationarity assumption.

Additional approaches exist but consider different signal models such as time series [MM15; SBG17], band-limited signals [SBDL16], combinations of localized functions [Tha+16; PMTF17], or more recently opinions of agents in a network [SSJ17].

The graphical lasso

A widely-used approach to provide a graph is the *graphical lasso* [FHT08], which recovers a sparse ***precision matrix*** — *i.e.*, inverse covariance matrix — under the assumption that the data are observations from a multivariate Gaussian distribution. The core of this method consists in solving the following problem:

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta} \geq 0}{\operatorname{argmin}} \left(\operatorname{Tr}(\widetilde{\boldsymbol{\Sigma}}\boldsymbol{\Theta}) - \log\det(\boldsymbol{\Theta}) + \alpha \|\boldsymbol{\Theta}\|_{1,1} \right) \ , \tag{87}$$

where α is a regularization parameter controlling sparsity of the solution.

Numerous variations of this technique have been developed [Rot+08; WFS11; MH12; TWS15], and several applications have been using graphical lasso-based methods for inferring a sparse graph. Examples can be found for instance in the fields of neuroimaging [Hua+09; Yan+15] or traffic modeling [SHG12].

What makes this method interesting, in addition to its fast convergence to a sparse solution, is a previous result from Dempster [Dem72]. In the *covariance selection model*, Dempster proposes that the inverse covariance matrix should have numerous null off-diagonal entries. An additional result from Wermuth [Wer76] states that the non-null entries in the precision matrix correspond to existing edges in a graph that is representative of the studied data.

Therefore, in the experiments we perform later in this chapter, we evaluate whether considering the result of the graphical lasso as a graph makes it admissible or not to model a diffusion process. However, when considering (87), we can see that the method does not impose any similarity between the eigenvectors of the covariance matrix and those of the inferred solution. For this reason, we do not expect this method to provide a solution that is admissible in our settings.

Close to the graphical lasso, [PO16] and [EPO16] propose an algorithm to infer a precision matrix by adding generalized Laplacian constraints. While this allows for good recovery of the precision matrix, it proceeds in an iterative way by following a block descent algorithm that updates one row/column per iteration. As for the graphical lasso, it does not force the eigenvectors of the retrieved matrix to match those of the covariance matrix, and therefore does not match our stationarity assumption.

Interestingly, these methods could also be mentioned in the next section, dedicated to smoothness-based methods. In particular, [PO16] has pointed out that minimizing the quantity $\mathrm{Tr}(\widetilde{\mathbf{\Sigma}}\mathbf{\Theta})$ promotes smoothness of the solution when $\mathbf{\Theta}$ is a graph Laplacian. Additionally, [Rab17] promotes sparsity of the inferred graph by applying a soft threshold to the precision matrix, and shows that the solution matches a smoothness assumption on signals.

Smoothness-based methods

Another approach to infer a graph is to assume that the signal entries should be similar when the vertices on which they are defined are linked with a strong weight in $\mathbf{W}$, thus enforcing signals on this graph to be low-frequency (smooth). From Section 2.3.2, we know that for a signal $\mathbf{x}$, the smaller $\mathrm{S}(\mathbf{x})$, the more regular the entries of $\mathbf{x}$ on the graph.

A first work taking this approach has been proposed by Lake and Tenenbaum [LT10], in which they solve a convex optimization problem to recover a sparse graph from data to learn the structure best representing some concepts. More recently, Dong *et al.* [Don+16] have proposed a similar method that outperforms it.

In order to find a graph Laplacian that minimizes S in Definition 49 for a set of signals, the authors propose an iterative algorithm that converges to a local solution, based on

the resolution of the following problem:

$$\mathbf{L}^* = \underset{\mathbf{L},\mathbf{Y}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{Y}\|_F^2 + \alpha \operatorname{Tr}(\mathbf{Y}^\top \mathbf{L} \mathbf{Y}) + \beta \|\mathbf{L}\|_F^2$$

$$s.t. \begin{cases} \operatorname{Tr}(\mathbf{L}) = N \\ \forall i, j \in [\![1, N]\!], i \neq j : \mathbf{L}[i,j] = \mathbf{L}[j,i] \leq 0 \\ \forall i \in [\![1, N]\!] : \sum_{j=1}^{N} \mathbf{L}[i,j] = 0 \end{cases}, \tag{88}$$

where $\mathbf{Y}$ is a matrix in $\mathbb{R}^{N \times M}$ that can be considered as a noiseless version of signals $\mathbf{X}$, and α and β are regularization parameters controlling respectively the distance between $\mathbf{X}$ and $\mathbf{Y}$, and the sparsity of the solution.

Kalofolias [Kal16] proposes a unifying framework to improve the previous solutions of Lake and Tenenbaum, and Dong *et al.*, by proposing a better prior and reformulating the problem to optimize over entries of the (weighted) adjacency matrix rather than the Laplacian. An efficient implementation of his work is provided in the Graph Signal Processing Toolbox [Per+14]. His approach consists in rewriting the problem as an $L_{1,1}$ minimization, that leads to naturally sparse solutions. Moreover, the author has shown that the method from Dong *et al.* could be encoded in his framework.

Graph inference with smoothness priors continues to receive a lot of interest. Recently, Chepuri *et al.* [Che+17] have proposed to infer a sparse graph on which signals are smooth, using an edge selection strategy.

Finally, enforcing the smoothness property for signals defined on a graph has also been considered by Shivaswamy and Jebara [SJ10], where a method is proposed to jointly learn the kernel of an SVM classifier and optimize the spectrum of the Laplacian to improve this classification. Contrary to our approach, Shivaswamy and Jebara [SJ10] study a semi-supervised case, in which the spectrum of the Laplacian is learned based on a set of labeled examples.

Diffusion-based methods

As indicated in the introduction of this chapter, it contents is mainly based on two contributions. In [Pas+15a], we study the case of graph inference from signals diffused on a graph, under a simplified diffusion model. Then, [Pas+17a] generalizes this first work, by removing some of its assumptions, and rephrasing it in the more general context of stationary signals. As these contributions will be extensively described in the next sections, they are not summarized here.

Independently from the two pieces of work introduced above, the group of Segarra *et al.* obtained similar results as ours, by taking the same direction [Seg+16; Seg+17a; Seg+17b]. The authors propose a two-step approach, where they first retrieve the eigenvectors of a graph shift operator, and then infer the missing eigenvalues based on some criteria. They also study the case of stationary graph processes, for which the covariance matrix shares the same eigenbasis as the graph shift operator used for diffusion, and use this information to infer a graph based on additional criteria.

However, while the characterization of the set of solutions is identical to ours, our works differ in the matrix selection strategy. The authors of [Seg+17a] solve a slightly different problem, where they minimize the $L_{1,1}$ norm of the inferred matrix under more constraints than ours, which describe a valid Laplacian matrix. In particular, they

enforce the diagonal elements of the solution to be null, thus considering graphs that do not admit self-loops. In more details, they solve the following optimization problem:

$$\mathbf{S}^* = \underset{\mathbf{S},\lambda_1,\dots,\lambda_N}{\operatorname{argmin}} \|\mathbf{S}\|_{0,1}$$
$$s.t. \begin{cases} \mathbf{S} = \sum_{i=1}^{N} \lambda_i \boldsymbol{\chi}_i \boldsymbol{\chi}^\top{}_i \\ \mathbf{S} \in \mathcal{S} \end{cases}, \tag{89}$$

where $\mathcal{S}$ is the set of admissible solutions, delimited by some constraints, and $(\boldsymbol{\chi}_1, \dots, \boldsymbol{\chi}_N)$ are the eigenvectors of the covariance matrix.

Contrary to their approach, we aim at inferring a matrix that can be simple or sparse, rather than selecting a sparse matrix from the restricted set of simple matrices. Among other differences, we propose in [Pas+17a] a method to approximate the solution of any graph inference strategy to make it match our stationary assumption on signals. Our work also explores how the set of solutions can be used to evaluate which graph, among a set of given graphs, is the most adapted to given signals.

Other related work

Shahrampour and Preciado [SP13; SP15] study the context of network inference from stimulation of its vertices with noise. However, their method implies a series of *node knockout* operations that need to individually intervene on the vertices.

Also, we note that there exist methods that aim to recover a graph from the knowledge of its spectrum [IM02]. However, we do not assume that such information is available.

Finally, a recent work by Shafipour *et al.* [Sha+17] has started to explore the problem of graph inference from non-stationary graph signals, which is a direct continuation of the work presented in this chapter and of the work by Segarra *et al.*.

3.2 The simplified case of the monomial graph filter

3.2.1 Problem simplification

In this chapter, we present the main results from [Pas+15a]. In this first work, we consider a simplified case of graph filters, namely monomials of the graph shift operator. Additionally, we consider here a single filter for all signals.

In more details, we adapt the problem formulation in Section 3.1.1 as follows:

- Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a graph of N vertices, and let $\boldsymbol{\mathcal{T}}$ be the diffusion matrix associated with the normalized Laplacian of $\mathcal{G}$;
- We are given a sequence of M signals $(\mathbf{x}_1, \dots, \mathbf{x}_M)$ in $\mathbb{R}^N$. Without loss of generality, we consider signals that are zero-mean and of unit norm;
- Let $\mathbf{H} = \boldsymbol{\mathcal{T}}^K$ be a monomial graph filter of variable $\boldsymbol{\mathcal{T}}$, with $K \in \mathbb{R}$ known;
- Let $(\mathbf{y}_1, \dots, \mathbf{y}_M)$ be M realizations of a zero-mean white noise process;
- We assume that the observed signals $(\mathbf{x}_1, \dots, \mathbf{x}_M)$ were observed after diffusion of signals $(\mathbf{y}_1, \dots, \mathbf{y}_M)$ using this graph filter, *i.e.*, we suppose the following:

$$\forall i \in [\![1, M]\!] : \mathbf{x}_i = \mathbf{H}\mathbf{y}_i \; ; \tag{90}$$

- Then, from the knowledge of $(\mathbf{x}_1, \dots, \mathbf{x}_M)$ and K, and assuming that these signals are issued from the diffusion of *i.i.d.* signals on a graph, can we find the graph shift operator $\boldsymbol{\mathcal{T}}$ used to diffuse the signals?

Using these simplified settings, the development in (80) gives us that:

$$\begin{aligned} \boldsymbol{\Sigma} &= \mathbb{E}\left[\mathbf{H}^2\right] \\ &= \boldsymbol{\mathcal{T}}^{2K} \end{aligned} \quad . \tag{91}$$

Using Proposition 4, we know that the eigenvectors of the covariance matrix are the eigenvectors of the graph shift operator we want to find. Additionally, since power distributes on the eigenvalues of a matrix, we obtain that:

$$\boldsymbol{\Sigma} = \boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{T}}} \operatorname{diag}(\boldsymbol{\lambda}_{\boldsymbol{\mathcal{T}}})^{2K} \, \boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{T}}}{}^{\top} \, , \tag{92}$$

where $\boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{T}}}$ and $\boldsymbol{\lambda}_{\boldsymbol{\mathcal{T}}}$ are the eigenvectors and eigenvalues of $\boldsymbol{\mathcal{T}}$, respectively.

3.2.2 Identifying the missing eigenvalues

From the eigenvectors of the covariance matrix

Using the simplification in (92), the problem of retrieving $\boldsymbol{\mathcal{T}}$ from the signals falls down to identifying its eigenvalues.

A possible idea to do so consists in taking the $2K$-square root of the eigenvalues of the covariance matrix. However, since $2K$ is an even power, we would be missing the sign of the eigenvalues of $\boldsymbol{\mathcal{T}}$ by doing so.

In [Pas+15a], we propose to take this direction, and to find the missing signs. Let $\boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}$ be a vector of N entries such that:

$$\forall i \in [\![1, N]\!] : \boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}[i] = \left| \sqrt[2K]{\boldsymbol{\lambda}_{\boldsymbol{\Sigma}}[i]} \right| \, , \tag{93}$$

where $\boldsymbol{\lambda}_{\boldsymbol{\Sigma}}$ are the eigenvalues of the covariance matrix. Additionally, let $\mathbf{s}$ be a vector of signs, *i.e.*, such that its entries are in $\{-1, 1\}$.

We can thus characterize the graph shift operator used to diffuse the signals as follows:

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}} \operatorname{diag}\left(\mathbf{s} \odot \boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}\right) \, \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}{}^{\top} \, . \tag{94}$$

Also, individual entries of $\boldsymbol{\mathcal{T}}$ can be written as linear combinations of N variables, corresponding to the missing signs:

$$\forall i, j \in [\![1, N]\!] : \boldsymbol{\mathcal{T}}[i, j] = \sum_{k=1}^{N} \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}[i, k] \, \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}[k, j] \, \boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}[k] \, \mathbf{s}[k] \, . \tag{95}$$

Even in this simplified framework, there are 2^N solutions corresponding to the possible signs of the eigenvalues. Therefore, we need to make assumptions on the graph shift operator we want to find.

Let us assume that the given signals $\mathbf{X}$ are obtained after diffusion on a positively weighted graph (see Remark 4). Additionally, let us assume that the graph is simple. Under these settings, we know that the weights of the graph shift operator used to

diffuse the signals are positive. Also, the simplicity assumption gives us that entries on the diagonal of the graph shift operator are null.

Using (95), these properties can be translated into a set $\mathcal{S}$ of constraints as follows:

- Positivity of the off-diagonal entries yields the following $\frac{N(N+1)}{2}$ constraints:

$$\forall i,j \in [\![1,N]\!], i \geq j : \left(\sum_{k=1}^{N} \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}[i,k]\ \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}[k,j]\ \boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}[k]\ \mathbf{s}[k] > 0 \right) \in \mathcal{S}\ ; \tag{96}$$

- Simplicity of the matrix yields the following N constraints:

$$\forall i \in [\![1,N]\!] : \left(\sum_{k=1}^{N} \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}[i,k]\ \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}[k,j]\ \boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}[k]\ \mathbf{s}[k] = 0 \right) \in \mathcal{S}\ . \tag{97}$$

Note that off-diagonal constraints are only necessary for the triangular upper part of the matrix, as symmetry is enforced by (94).

Then, we propose to solve $\mathcal{S}$ using a linear equations solver. Examples of such tools include `MATLAB` [MAT12] framework `CVX` [GBY08], used alongside with `SDPT3` [TTT99] or `Gurobi` [GO15] solvers. However, such solvers generally consider real variables, which is not our case as signs take their values in $\{-1,1\}$.

To cope with this problem, we relax our settings by performing a change of variable as if $\boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}$ were not known. In details, let $\boldsymbol{\lambda} \in \mathbb{R}^N$ be a vector of variables. We update (96) and (97) by replacing $\boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}[k]\ \mathbf{s}[k]$ with $\boldsymbol{\lambda}[k]$.

By doing this modification, the trivial solution $\boldsymbol{\lambda} = \mathbf{0}_N$ satisfies the new set of constraints. Therefore, noticing that the eigenvalue associated with the constant-sign eigenvector of $\boldsymbol{\mathcal{T}}$ is equal to 1 (see Section 2.3.2), we impose a scale to the solution with the following constraint:

$$(\boldsymbol{\lambda}[1] = 1) \in \mathcal{S}\ , \tag{98}$$

if $\boldsymbol{\chi}_1$ is the constant-sign eigenvector of $\boldsymbol{\mathcal{T}}$.

Our problem can then be written as follows:

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}}{\operatorname{solve}}\, \mathcal{S}\ , \tag{99}$$

where $\mathcal{S}$ follows from (96), (97) and (98).

As a matter of fact, due to the important number of constraints, and especially the equality constraints for the diagonal entries, the solution of $\mathcal{S}$ appears to be unique in most situations. Therefore, in most cases, solving $\mathcal{S}$ allows for perfect retrieval of $\boldsymbol{\lambda}_{\boldsymbol{\mathcal{T}}}$, the eigenvalues of the graph shift operator. Still, note that the unique solution of $\mathcal{S}$ is $\boldsymbol{\lambda}_{\boldsymbol{\mathcal{T}}}$ only in the case when the eigenvectors used in the constraints are those of the covariance matrix.

The constraints defined above encode properties of the graph shift operator we want to retrieve, namely a positively weighted simple matrix. When considering different assumptions on the signals, such as diffusion using a graph Laplacian for instance, these constraints need to be adapted accordingly.

From the eigenvectors of the sample covariance matrix

In practical cases, the eigenvectors of the sample covariance are used instead of those of the covariance matrix, which requires a slight adaptation of the set of constraints. As the eigenvectors of the sample covariance matrix are not exactly those of the covariance matrix, the equality constraints (97) along the diagonal are nearly impossible to match. Therefore, when considering the sample covariance matrix, we consider a set of constraints $\mathcal{S}$ in (99) that contains (96) and (98) only, in which $\widetilde{\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}}$ replace $\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}$.

Still, to keep the simplicity assumption, we use the property that the trace of a matrix is equal to the sum of its eigenvalues (see Section 2.1.3). Since the positivity constraints in (96) enforce the diagonal entries to be non-negative, this sum cannot be negative. Our problem thus becomes an optimization problem, stated as follows:

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda} \in \mathbb{R}^N}{\operatorname{argmin}} \sum_{i=1}^{N} \boldsymbol{\lambda}[i] \quad s.t. \quad \boldsymbol{\lambda} \text{ verifies } \mathcal{S} \ , \tag{100}$$

Equation (100) is a linear program for which it is known that polynomial-time algorithms exist. The main bottleneck of this method is the definition of the $\frac{N(N+1)}{2}$ linear constraints in (96), that are computed in $\mathcal{O}(N^3)$ time and space.

Under these settings, the solution found by the solver is not exactly $\boldsymbol{\lambda}_{\boldsymbol{\mathcal{T}}}$, but a deformation of it due to imprecisions in the eigenvectors of the sample covariance matrix. However, if these imprecisions are small — *i.e.*, if the number of signals from which the sample covariance matrix is computed is high enough — then the eigenvectors of the sample covariance matrix are close to those of the covariance matrix (see Section 3.1.2 for details). As a consequence, the retrieved eigenvalues are also close to $\boldsymbol{\lambda}_{\boldsymbol{\mathcal{T}}}$, and are probably of the same sign.

Therefore, one possible method to improve the result of the solver is to keep only the signs of $\boldsymbol{\lambda}^*$, and to inject them in $\widetilde{\boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}}$, the $2K$-square roots of the eigenvalues of the sample covariance matrix. Reconstruction of the graph shift operator is thus performed as follows:

$$\boldsymbol{\mathcal{T}}^* = \widetilde{\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}} \operatorname{diag}\left(\operatorname{sign}(\boldsymbol{\lambda}^*) \odot \widetilde{\boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}}\right) \widetilde{\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}}^{\top} , \tag{101}$$

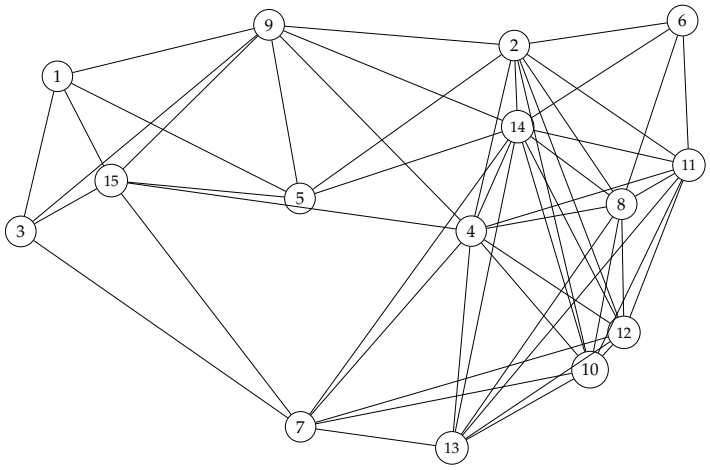
where $\operatorname{sign}(\cdot)$ is a function that returns the sign of the given scalar:

$$\operatorname{sign} : \begin{cases} \mathbb{R} & \to & \{-1, 1\} \\ x & \mapsto & \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases} \end{cases} . \tag{102}$$

Note that when the equality constraints (97) along the diagonal entries are removed due to the use of the eigenvectors of the sample covariance matrix, the space solutions to inequalities $\mathcal{S}$ is very unlikely to be a singleton. As a matter of fact, intersection of multiple inequality constraints is most probably a subspace of $\mathbb{R}^N$ rather than a single point in $\mathbb{R}^N$. This particular subspace will be studied in more details when considering the more general case of graph inference from stationary signals in Section 3.3.

3.2.3 A complete example

To illustrate the method introduced in this section, let us consider a ground truth graph that we want to infer from signals diffused on it. Figure 30 depicts the random geometric graph we work with:



$$\mathbf{W} = \begin{array}{c} \\ \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{4} \\ \textcircled{5} \\ \textcircled{6} \\ \textcircled{7} \\ \textcircled{8} \\ \textcircled{9} \\ \textcircled{10} \\ \textcircled{11} \\ \textcircled{12} \\ \textcircled{13} \\ \textcircled{14} \\ \textcircled{15} \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} \textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} & \textcircled{5} & \textcircled{6} & \textcircled{7} & \textcircled{8} & \textcircled{9} & \textcircled{10} & \textcircled{11} & \textcircled{12} & \textcircled{13} & \textcircled{14} & \textcircled{15} \end{array} \\ \begin{bmatrix} 0 & 0 & 0.58 & 0 & 0.689 & 0 & 0 & 0 & 0.726 & 0 & 0 & 0 & 0 & 0 & 0.678 \\ 0 & 0 & 0 & 0.805 & 0.636 & 0.711 & 0 & 0.803 & 0.696 & 0.515 & 0.765 & 0.569 & 0 & 0.961 & 0 \\ 0.58 & 0 & 0 & 0 & 0 & 0 & 0.498 & 0 & 0.536 & 0 & 0 & 0 & 0 & 0 & 0.938 \\ 0 & 0.805 & 0 & 0 & 0 & 0 & 0.668 & 0.87 & 0.607 & 0.819 & 0.734 & 0.818 & 0.753 & 0.925 & 0.453 \\ 0.689 & 0.636 & 0 & 0 & 0 & 0 & 0 & 0 & 0.866 & 0 & 0 & 0 & 0 & 0.512 & 0.459 \\ 0 & 0.711 & 0 & 0 & 0 & 0 & 0 & 0.513 & 0 & 0 & 0.606 & 0 & 0 & 0.583 & 0 \\ 0 & 0 & 0.498 & 0.668 & 0 & 0 & 0 & 0 & 0 & 0.594 & 0 & 0.507 & 0.87 & 0.44 & 0.561 \\ 0 & 0.803 & 0 & 0.87 & 0 & 0.513 & 0 & 0 & 0 & 0.844 & 0.964 & 0.906 & 0.59 & 0.904 & 0 \\ 0.726 & 0.696 & 0.536 & 0.607 & 0.866 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.649 & 0.745 \\ 0 & 0.515 & 0 & 0.819 & 0 & 0 & 0.594 & 0.844 & 0 & 0 & 0.734 & 0.985 & 0.86 & 0.681 & 0 \\ 0 & 0.765 & 0 & 0.734 & 0 & 0.606 & 0 & 0.964 & 0 & 0.734 & 0 & 0.825 & 0.443 & 0.832 & 0 \\ 0 & 0.569 & 0 & 0.818 & 0 & 0 & 0.507 & 0.906 & 0 & 0.985 & 0.825 & 0 & 0.773 & 0.726 & 0 \\ 0 & 0 & 0 & 0.753 & 0 & 0 & 0.87 & 0.59 & 0 & 0.86 & 0.443 & 0.773 & 0 & 0.525 & 0 \\ 0 & 0.961 & 0 & 0.925 & 0.512 & 0.583 & 0.44 & 0.904 & 0.649 & 0.681 & 0.832 & 0.726 & 0.525 & 0 & 0 \\ 0.678 & 0 & 0.938 & 0.453 & 0.459 & 0 & 0.561 & 0 & 0.745 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 28 – Random geometric graph of $N = 15$ vertices (top) serving as a continuing example in this section. Weights on the edges (bottom) are set using a decreasing function of the distance, as in (22), for $\theta = 0.4$, rounded at 10^{-3}.

From the adjacency matrix of this graph, we compute $\boldsymbol{\mathcal{T}}$, the diffusion matrix associated with the normalized graph Laplacian. The eigenvectors and eigenvalues of this graph shift operator are given as follows:

$$\boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{T}}} = \begin{bmatrix} 0.189 & 0.439 & -0.371 & 0.061 & 0.064 & 0.411 & -0.484 & -0.023 & -0.027 & 0.048 & -0.061 & 0.143 & -0.037 & -0.221 & 0.379 \\ 0.294 & -0.08 & -0.243 & 0.39 & -0.247 & 0.331 & 0.166 & 0.177 & -0.187 & -0.036 & -0.27 & -0.537 & -0.016 & 0.158 & -0.208 \\ 0.185 & 0.39 & 0.546 & -0.229 & -0.081 & 0.307 & 0.126 & 0.282 & 0.117 & -0.14 & 0.129 & -0.126 & 0.442 & 0.054 & 0.055 \\ 0.316 & -0.102 & 0.241 & -0.129 & -0.021 & 0.153 & -0.353 & -0.192 & -0.072 & 0.165 & -0.142 & 0.275 & -0.159 & 0.663 & -0.189 \\ 0.206 & 0.297 & 0.493 & 0.299 & 0.051 & -0.046 & 0.198 & -0.396 & -0.146 & 0.079 & -0.109 & -0.04 & -0.442 & -0.317 & 0.006 \\ 0.18 & -0.133 & 0.184 & 0.366 & 0.062 & -0.337 & -0.475 & 0.031 & -0.106 & 0.035 & -0.162 & 0.035 & 0.522 & -0.275 & -0.225 \\ 0.236 & 0.027 & -0.195 & -0.478 & 0.546 & 0.114 & 0.079 & -0.214 & -0.113 & 0.028 & -0.042 & -0.192 & 0.087 & -0.217 & -0.455 \\ 0.293 & -0.23 & -0.052 & 0.102 & 0.243 & 0.017 & 0.281 & -0.141 & -0.147 & -0.63 & -0.175 & 0.303 & 0.163 & 0.089 & 0.338 \\ 0.254 & 0.352 & -0.116 & 0.19 & 0.39 & -0.393 & 0.078 & 0.553 & 0.114 & -0.004 & 0.068 & 0.098 & -0.243 & 0.217 & -0.1 \\ 0.285 & -0.203 & 0.062 & -0.206 & -0.032 & -0.179 & -0.162 & -0.033 & 0.679 & -0.073 & -0.316 & -0.336 & -0.163 & -0.093 & 0.239 \\ 0.281 & -0.222 & -0.055 & 0.148 & 0.162 & 0.086 & 0.354 & -0.044 & 0.13 & 0.693 & 0.125 & 0.119 & 0.266 & 0.008 & 0.295 \\ 0.286 & -0.208 & 0.066 & -0.164 & -0.015 & -0.204 & -0.224 & 0.048 & -0.425 & -0.037 & 0.547 & -0.375 & -0.138 & 0.013 & 0.327 \\ 0.254 & -0.17 & 0.021 & -0.355 & -0.365 & -0.024 & 0.105 & 0.425 & -0.258 & 0.113 & -0.272 & 0.358 & -0.179 & -0.372 & -0.052 \\ 0.322 & -0.117 & -0.114 & 0.177 & -0.242 & 0.165 & 0.003 & -0.096 & 0.371 & -0.207 & 0.567 & 0.249 & -0.088 & -0.184 & -0.371 \\ 0.227 & 0.422 & -0.303 & -0.172 & -0.44 & -0.46 & 0.169 & -0.357 & -0.046 & 0.026 & -0.034 & -0.003 & 0.246 & 0.153 & 0 \end{bmatrix}$$

$$\boldsymbol{\lambda}_{\boldsymbol{\mathcal{T}}}^{\top} = \begin{bmatrix} 1 & 0.723 & -0.405 & 0.359 & -0.315 & -0.291 & -0.261 & -0.215 & -0.169 & -0.158 & -0.154 & -0.129 & 0.094 & -0.06 & -0.018 \end{bmatrix}$$

Figure 29 – Eigenvectors (top) and associated eigenvalues (bottom) of the matrix $\boldsymbol{\mathcal{T}}$ obtained from the graph in Figure 30, rounded at 10^{-3}. Eigenvectors are sorted by decreasing modulus of the corresponding eigenvalues.

Using the knowledge that $\boldsymbol{\mathcal{X}}$ are also the eigenvectors of the covariance matrix of signals diffused on the graph using the graph shift operator $\boldsymbol{\mathcal{T}}$, we create the set of con-

straints $\mathcal{S}$ as explained in Section 3.2.2. Solving this system of constraints gives us the following vector of eigenvalues:

$$\boldsymbol{\lambda}^{*\top} = \begin{bmatrix} 1 & 0.723 & -0.405 & 0.359 & -0.315 & -0.291 & -0.261 & -0.215 & -0.169 & -0.158 & -0.154 & -0.129 & 0.094 & -0.06 & -0.018 \end{bmatrix} , \tag{103}$$

which is exactly the vector of eigenvalues of $\boldsymbol{\mathcal{T}}$ in Figure 30.

Now, let us consider the practical case of the sample covariance matrix. We create an $N \times M$ ($M = 500$) matrix $\mathbf{Y}$ of *i.i.d.* signals such that every signal entry follows a $\mathcal{N}(0, 1)$ distribution,. Then, we diffuse each signal $K = 4$ times to obtain a matrix of diffused signals, *i.e.*:

$$\mathbf{X} = \boldsymbol{\mathcal{T}}^4 \mathbf{Y} . \tag{104}$$

Figure 30 depicts the eigenvectors and eigenvalues of the sample covariance matrix obtained from a realization of $\mathbf{X}$:

$$\widetilde{\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}} = \begin{bmatrix} 0.187 & 0.441 & 0.358 & -0.062 & -0.093 & 0.398 & -0.5 & -0.019 & 0.042 & -0.064 & 0.05 & 0.139 & -0.037 & -0.221 & 0.379 \\ 0.295 & -0.076 & 0.243 & -0.383 & 0.261 & 0.34 & 0.143 & 0.174 & 0.153 & 0.004 & 0.275 & -0.547 & -0.026 & 0.157 & -0.208 \\ 0.183 & 0.389 & -0.541 & 0.243 & 0.101 & 0.304 & 0.118 & 0.281 & -0.146 & 0.156 & -0.097 & -0.124 & 0.438 & 0.048 & 0.055 \\ 0.317 & -0.102 & -0.248 & 0.127 & -0.001 & 0.132 & -0.362 & -0.184 & 0.107 & -0.197 & 0.1 & 0.264 & -0.141 & 0.666 & -0.189 \\ 0.204 & 0.299 & -0.496 & -0.288 & -0.038 & -0.038 & 0.208 & -0.393 & 0.163 & -0.081 & 0.096 & -0.04 & -0.446 & -0.311 & 0.006 \\ 0.181 & -0.13 & -0.192 & -0.381 & -0.102 & -0.346 & -0.444 & 0.048 & 0.104 & -0.066 & 0.148 & 0.015 & 0.52 & -0.282 & -0.225 \\ 0.235 & 0.026 & 0.195 & 0.484 & -0.537 & 0.116 & 0.096 & -0.213 & 0.114 & -0.01 & 0.038 & -0.198 & 0.08 & -0.219 & -0.455 \\ 0.294 & -0.227 & 0.054 & -0.096 & -0.218 & 0.034 & 0.297 & -0.144 & 0.055 & 0.554 & 0.358 & 0.318 & 0.177 & 0.087 & 0.338 \\ 0.252 & 0.355 & 0.122 & -0.194 & -0.39 & -0.372 & 0.132 & 0.549 & -0.124 & 0.002 & -0.069 & 0.112 & -0.239 & 0.221 & -0.1 \\ 0.286 & -0.203 & -0.06 & 0.196 & 0.016 & -0.188 & -0.166 & -0.048 & -0.701 & -0.096 & 0.266 & -0.321 & -0.172 & -0.09 & 0.239 \\ 0.282 & -0.22 & 0.058 & -0.14 & -0.141 & 0.107 & 0.36 & -0.057 & -0.017 & -0.645 & -0.322 & 0.087 & 0.264 & 0.005 & 0.295 \\ 0.288 & -0.208 & -0.067 & 0.156 & -0.002 & -0.218 & -0.213 & 0.072 & 0.422 & 0.267 & -0.486 & -0.363 & -0.149 & 0.014 & 0.327 \\ 0.256 & -0.17 & -0.01 & 0.353 & 0.374 & -0.034 & 0.083 & 0.427 & 0.258 & -0.173 & 0.257 & 0.346 & -0.174 & -0.369 & -0.052 \\ 0.323 & -0.114 & 0.112 & -0.177 & 0.244 & 0.168 & -0.021 & -0.107 & -0.366 & 0.296 & -0.509 & 0.282 & -0.089 & -0.183 & -0.371 \\ 0.225 & 0.423 & 0.309 & 0.163 & 0.44 & -0.466 & 0.14 & -0.359 & 0.054 & -0.029 & 0.024 & -0.014 & 0.249 & 0.15 & 0 \end{bmatrix}$$

$$\widetilde{\boldsymbol{\lambda}_{\boldsymbol{\Sigma}}}^{\top} = \begin{bmatrix} 1 & 0.616 & -0.336 & 0.281 & -0.261 & -0.278 & -0.21 & -0.183 & -0.155 & -0.151 & -0.146 & -0.112 & 0.076 & -0.08 & -0.045 \end{bmatrix}$$

$$\widetilde{\boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}}^{\top} = \begin{bmatrix} 1 & 0.713 & 0.4 & 0.349 & 0.312 & 0.288 & 0.256 & 0.209 & 0.165 & 0.156 & 0.152 & 0.128 & 0.092 & 0.059 & 0.018 \end{bmatrix}$$

Figure 30 – Eigenvectors (top) and associated eigenvalues (middle) of the sample covariance matrix $\widetilde{\boldsymbol{\Sigma}}$ obtained from the signals $\mathbf{X}$ (104), rounded at 10^{-3}. Eigenvalues of $\widetilde{\boldsymbol{\lambda}_{\boldsymbol{\Sigma}}}$ are normalized so that the highest eigenvalue has amplitude 1. The $2K$-square root of these eigenvalues are also given (bottom).

Then, we consider the set of constraints $\mathcal{S}$ that contains (96) and (98) only, and solve (100) using the sample covariance matrix:

$$\boldsymbol{\lambda}^{*\top} = \begin{bmatrix} 1 & 0.616 & -0.336 & 0.281 & -0.261 & -0.278 & -0.21 & -0.183 & -0.155 & -0.151 & -0.146 & -0.112 & 0.076 & -0.08 & -0.045 \end{bmatrix} . \tag{105}$$

Finally, to provide a better approximate of the eigenvalues of the graph shift operator, we keep only the signs of $\boldsymbol{\lambda}^*$, and inject them into $\widetilde{\boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}}$:

$$\left(\operatorname{sign}(\boldsymbol{\lambda}^*) \odot \widetilde{\boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}}\right)^{\top} = \begin{bmatrix} 1 & 0.713 & -0.4 & 0.349 & -0.312 & -0.288 & -0.256 & -0.209 & -0.165 & -0.156 & -0.152 & -0.128 & 0.092 & -0.059 & -0.018 \end{bmatrix} . \tag{106}$$

To check whether use of the $2K$-square roots improves the result, we compute the ℓ_2 norm of the difference with the ground truth eigenvalues:

- $\|\boldsymbol{\lambda}_{\boldsymbol{\mathcal{T}}} - \boldsymbol{\lambda}^*\|_2 = 0.1762$;
- $\left\|\boldsymbol{\lambda}_{\boldsymbol{\mathcal{T}}} - \operatorname{sign}(\boldsymbol{\lambda}^*) \odot \widetilde{\boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}}\right\|_2 = 0.0183$.

Finally, we retrieve $\boldsymbol{\mathcal{T}}^*$, our estimate for $\boldsymbol{\mathcal{T}}$, using (94) with the eigenvectors of the sample covariance matrix:

$$\boldsymbol{\mathcal{T}}^* = \begin{array}{c} \\ ① \\ ② \\ ③ \\ ④ \\ ⑤ \\ ⑥ \\ ⑦ \\ ⑧ \\ ⑨ \\ ⑩ \\ ⑪ \\ ⑫ \\ ⑬ \\ ⑭ \\ ⑮ \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} ① & ② & ③ & ④ & ⑤ & ⑥ & ⑦ & ⑧ & ⑨ & ⑩ & ⑪ & ⑫ & ⑬ & ⑭ & ⑮ \end{array} \\ \begin{bmatrix} 0.002 & 0.002 & 0.217 & 0 & 0.233 & 0.002 & -0.001 & 0 & 0.2 & 0 & 0 & 0.001 & 0 & 0.002 & 0.211 \\ 0.002 & -0.003 & -0.002 & 0.118 & 0.137 & 0.179 & 0.003 & 0.121 & 0.125 & 0.086 & 0.119 & 0.094 & 0.003 & 0.133 & 0.004 \\ 0.217 & -0.002 & 0.002 & 0.003 & -0.001 & -0.004 & 0.154 & 0.002 & 0.151 & 0.001 & 0.001 & -0.001 & 0.005 & -0.001 & 0.295 \\ 0 & 0.118 & 0.003 & 0 & -0.001 & 0.002 & 0.119 & 0.127 & 0.099 & 0.121 & 0.112 & 0.121 & 0.125 & 0.122 & 0.085 \\ 0.233 & 0.137 & -0.001 & -0.001 & -0.003 & 0 & 0.004 & -0.002 & 0.219 & 0.004 & -0.002 & 0.004 & 0.003 & 0.101 & 0.134 \\ 0.002 & 0.179 & -0.004 & 0.002 & 0 & 0.005 & -0.006 & 0.13 & 0.003 & 0 & 0.157 & 0 & 0.001 & 0.136 & 0.003 \\ -0.001 & 0.003 & 0.154 & 0.119 & 0.004 & -0.006 & 0.003 & 0.004 & -0.002 & 0.115 & 0.003 & 0.099 & 0.194 & 0.078 & 0.136 \\ 0 & 0.121 & 0.002 & 0.127 & -0.002 & 0.13 & 0.004 & 0.003 & 0 & 0.135 & 0.155 & 0.145 & 0.107 & 0.126 & 0.003 \\ 0.2 & 0.125 & 0.151 & 0.099 & 0.219 & 0.003 & -0.002 & 0 & 0.003 & 0.002 & 0.001 & 0 & 0 & 0.106 & 0.172 \\ 0 & 0.086 & 0.001 & 0.121 & 0.004 & 0 & 0.115 & 0.135 & 0.002 & 0 & 0.125 & 0.16 & 0.158 & 0.102 & -0.002 \\ 0 & 0.119 & 0.001 & 0.112 & -0.002 & 0.157 & 0.003 & 0.155 & 0.001 & 0.125 & -0.001 & 0.138 & 0.085 & 0.122 & 0.003 \\ 0.001 & 0.094 & -0.001 & 0.121 & 0.004 & 0 & 0.099 & 0.145 & 0 & 0.16 & 0.138 & 0.001 & 0.141 & 0.106 & -0.002 \\ 0 & 0.003 & 0.005 & 0.125 & 0.003 & 0.001 & 0.194 & 0.107 & 0 & 0.158 & 0.085 & 0.141 & 0 & 0.087 & -0.003 \\ 0.002 & 0.133 & -0.001 & 0.122 & 0.101 & 0.136 & 0.078 & 0.126 & 0.106 & 0.102 & 0.122 & 0.106 & 0.087 & 0.001 & 0.003 \\ 0.211 & 0.004 & 0.295 & 0.085 & 0.134 & 0.003 & 0.136 & 0.003 & 0.172 & -0.002 & 0.003 & -0.002 & -0.003 & 0.003 & -0.002 \end{bmatrix} \end{array} . \tag{107}$$

As we can see, this matrix is not exactly equal to the graph shift operator used to diffuse the signals. However, entries that were equal to 0 in $\boldsymbol{\mathcal{T}}$ are very low in $\boldsymbol{\mathcal{T}}^*$, and setting to 0 all entries $\boldsymbol{\mathcal{T}}^*[i,j]$ such that $|\boldsymbol{\mathcal{T}}^*[i,j]| > 10^{-2}$, and the others to 1, allows for a perfect reconstruction of the adjacency matrix of the ground truth graph.

3.2.4 Removing some constraints

As indicated in Section 3.2.2, the solution of (99) is in most cases unique. This means that multiple constraints are probably not informative, and do not restrict the set of solutions. To confirm this hypothesis, in [Pas+15a], we tried to remove random constraints (chosen uniformly) from (96).

In the following experiment, we use the eigenvectors $\widetilde{\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}}$ of the sample covariance matrix of $M = 500$ signals, obtained after $K = 4$ diffusions on Erdős-Rényi graphs of $N = 15$ vertices, with weights drawn uniformly in $[0, 1]$. We restrict our analysis to graphs such that all eigenvalues of $\boldsymbol{\mathcal{T}}$ have a distinct modulus (see Remark 6). As a matter of fact, in the case where modulus of eigenvalues may be repeated, finding their missing signs in cannot be done uniquely.

Figure 31 studies the ℓ_2 norm of the difference between the eigenvalues of the ground truth graph shift operator and their estimates, as a function of the number of removed constraints. In more details, each constraint is given a probability to appear in $\mathcal{S}$.

The obtained results show that, for every edge probability, the error increases as the average number of off-diagonal constraints in $\mathcal{S}$ reduces. However, this error does not grow much, which indicates that suppression of constraints is possible to accelerate the solver used, without a strong impact on the results.

The results also show that the graph sparsity has some impact on the results. As a matter of fact, graphs with more edges tend to have a larger error, due to the fixed number of diffusions K. Since the average shortest path length between two vertices is smaller for denser graphs, convergence of diffused signals to a stable state occurs faster than for less connected graphs. Therefore, the signals from which the sample covariance matrix is computed encodes less information on the graph topology than if diffused on a sparser graph.

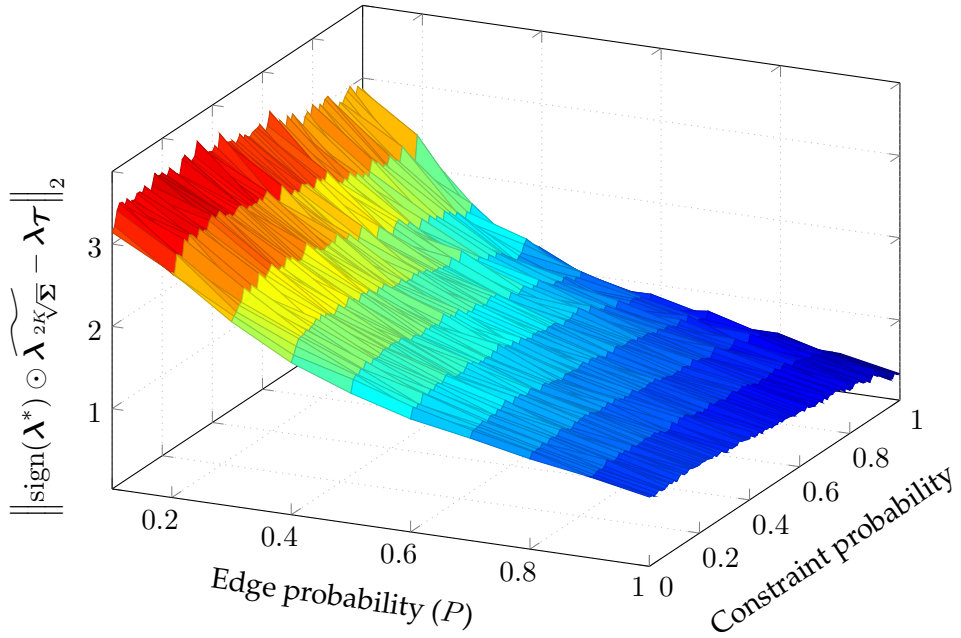


Figure 31 – Impact of the number of kept positivity constraints on the retrieved eigenvalues. Constraints are created out of the eigenvectors of the sample covariance matrix of $M = 500$ signals, diffused $K = 4$ times on ground truth graphs using $\boldsymbol{\mathcal{T}}$. The depicted results are obtained from 100 Monte-Carlo simulations, each one considering an Erdős-Rényi graph of $N = 15$ vertices, with weights drawn uniformly in $[0, 1]$. The ℓ_2 norm of the difference between the inferred eigenvalues $\text{sign}(\boldsymbol{\lambda}^*) \odot \widetilde{\boldsymbol{\lambda}_{\sqrt[2K]{\boldsymbol{\Sigma}}}}$ and those of the ground truth matrix $\boldsymbol{\mathcal{T}}$ is given as a function of P (in the Erdős-Rényi model) and of the individual probability for a constraint to appear in $\mathcal{S}$.

This is confirmed by the smaller median distance between eigenvalues for denser graphs, as depicted in Figure 32. Therefore, convergence of the eigenvectors of the sample covariance matrix to those of the covariance matrix is imprecise (see Section 3.1.2):

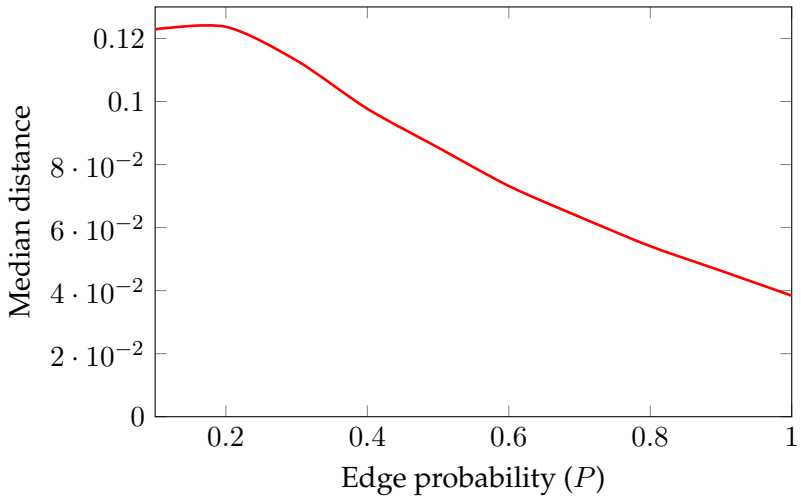


Figure 32 – Median distance between consecutive eigenvalues of the ground truth graphs in Figure 30, as a function of the edge probability of the Erdős-Rényi model. The median distance is computed as follows: $\text{median}\left(\{\|\lambda_i - \lambda_{i+1}\|_2 \mid \forall i \in [\![1, N-1]\!] : \lambda_i < \lambda_{i+1}\}\right)$.

3.3 Graph inference from stationary signals

We have seen in Section 3.2 that under simplified settings, it is possible to retrieve a ground truth graph on which signals have been diffused, provided that the eigenvectors of the covariance matrix are known or correctly approximated. Additionally, knowledge of K — the number of diffusions per signal — helps improving the quality of the inferred solution, since the ground truth graph shift operator used for diffusion is a root of the covariance matrix.

Also, we have made the assumption that observed signals were obtained after diffusion on a simple graph with non-negative weights. Under these assumptions, we were able to define a set of constraints $\mathcal{S}$ encoding these properties. When using the eigenvectors of the covariance matrix to define these constraints, the solution was in most cases unique, mostly due to the strong equality constraints along the diagonal of the inferred matrix. When removing these equality constraints, and using the eigenvectors of the sample covariance matrix instead of those of the covariance matrix, the space of solutions was generally not a singleton anymore.

In this section, we consider the more general case introduced in Section 3.1.1, in which the observed signals can be diffused using different graph filters of the same variable, which details are not known. Note that under these more general settings, the result in Proposition 4 still holds.

Let us consider again heat diffusion of signals using $\boldsymbol{\mathcal{T}}$ on positively-weighted graphs. However, contrary to the previous section, we do not assume the graph shift operator to be a simple matrix for the moment. Under these settings, the set of constraints $\boldsymbol{\mathcal{T}}$ can be built using (96) and (98). In this section, we propose to study in more details the set of solutions to these constraints.

3.3.1 Characterization of the set of admissible solutions

The admissible solutions lie in a convex set

Due to the absence of equality constraints, the admissible solutions lie in a convex space delimited by affine inequalities. Additionally, noticing that eigenvalues of $\boldsymbol{\mathcal{T}}$ are located in $[-1, 1]$, we can add the following constraints:

$$\forall i \in [\![1, N]\!] : (-1 - \varepsilon \leq \boldsymbol{\lambda}[i] \leq 1 + \varepsilon) \in \mathcal{S} \ , \tag{108}$$

where ε allows a controlled margin to cope with the imprecisions of the eigenvectors of the sample covariance matrix. Adding these constraints has the effect to cause the space of solutions to be closed. Still, note that these constraints are not necessary in all situations, since the positivity constraints in (96) can already yield a closed set.

In general, it is interesting to note that this convex set is not a singleton. As a matter of fact, the covariance matrix belongs to the set of admissible matrices, along with all its powers, which are different if the covariance matrix is not the identity matrix. When additional constraints delimit the polytope, for example when enforcing the matrices to be simple, the solution may be unique (see [Seg+17a] for more information).

To illustrate this, let us consider the following weights matrix:

$$\mathbf{W} = \begin{bmatrix} 0.417 & 0.302 & 0.186 \\ 0.302 & 0.147 & 0.346 \\ 0.186 & 0.346 & 0.397 \end{bmatrix} \ . \tag{109}$$

We compute the diffusion matrix $\boldsymbol{\mathcal{T}}$ associated with the normalized Laplacian of $\mathbf{W}$, and its eigenvectors $\boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{T}}}$. This simulates a perfect retrieval of the eigenvectors of the covariance matrix of signals diffused by any graph filter of variable $\boldsymbol{\mathcal{T}}$ on the graph. Using these eigenvectors, we plot in Figure 33 the pairs (λ_2, λ_3) (using a step of 10^{-2}) such that $\boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{T}}} \operatorname{diag}\left(\begin{bmatrix} 1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}\right) \boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{T}}}^{\top}$ contains non-negative entries only:

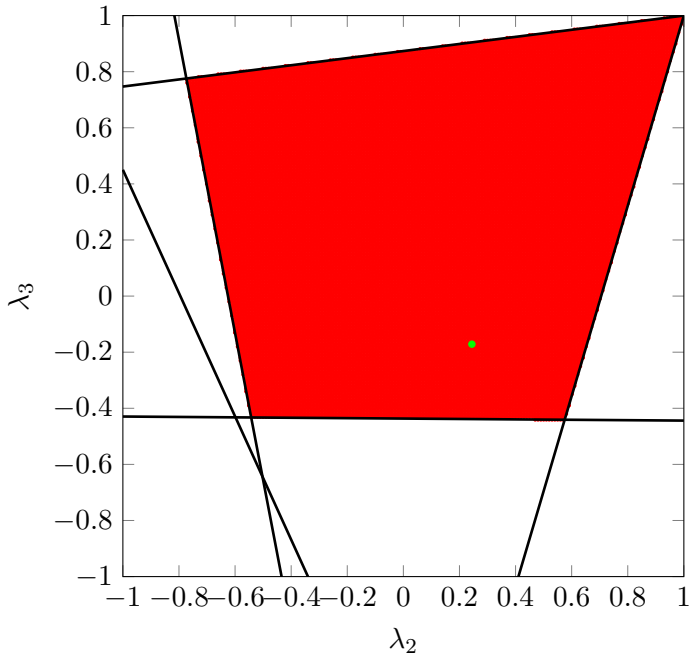


Figure 33 – Pairs $(\lambda_2, \lambda_3) \in [-1, 1] \times [-1, 1]$ (using a step of 10^{-2}) that, when used as eigenvalues along with $\boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{T}}}$, allow inference of a matrix with non-negative entries only (in red). The eigenvalues of the matrix $\boldsymbol{\mathcal{T}}$ are located using a green dot, and the positivity constraints in (96) are depicted with black lines.

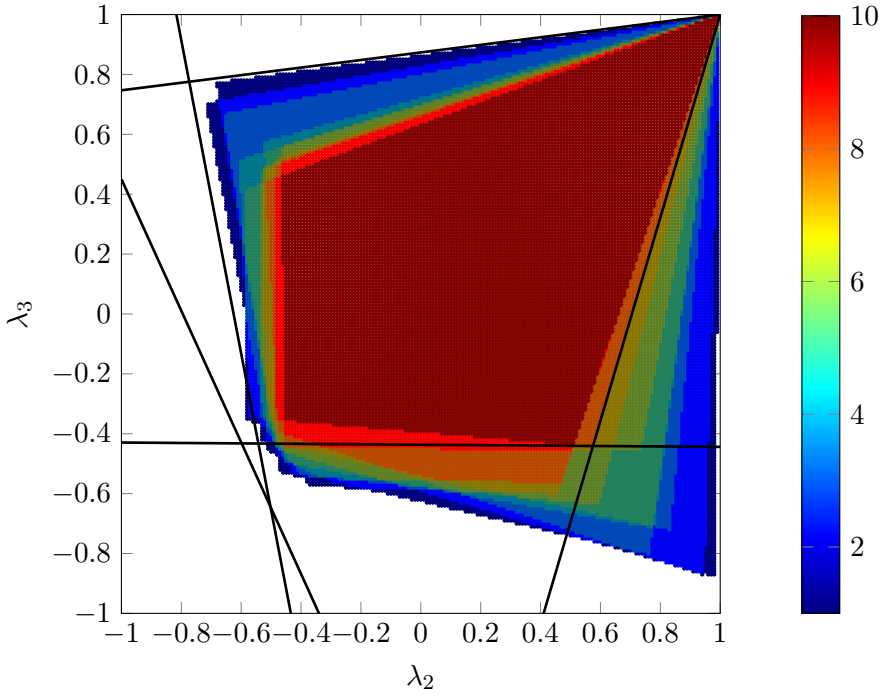
Our problem of recovering the correct set of eigenvalues to reconstruct the graph shift operator thus becomes a problem of selecting a vector of dimension $N - 1$ (since one eigenvalue is set to 1 due to the imposed scale) in a convex polytope. Since the number of solutions is possibly infinite, it is an ill-posed problem. To cope with this issue, one then needs to incorporate additional information or a selection criterion to enforce desired properties on the reconstructed matrix. A first example, enforcing simplicity of the solution, was already studied in Section 3.2.2. Section 3.3.1 introduces another method to select a point from the polytope, based on a sparsity criterion.

Characterization of the convex set using the sample covariance matrix

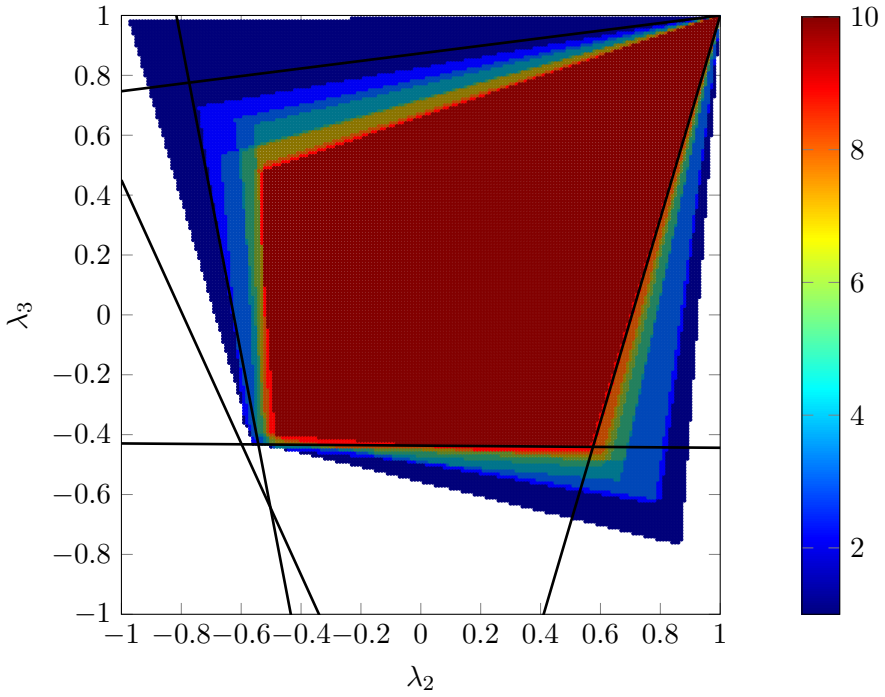
Figure 33 depicts a polytope delimited by affine constraints, computed from the eigenvectors of the covariance matrix. In practical case, the eigenvectors of the sample covariance matrix are used instead, which leads to deformations of the polytope.

In Figure 34, we consider the matrix in (109), and we diffuse $M \in \{10, 10^2, 10^3, 10^4, 10^5\}$ signals using graph filters of variable $\boldsymbol{\mathcal{T}}$, the diffusion matrix associated with the normalized Laplacian of (109). Here, we consider random graph filters with maximum

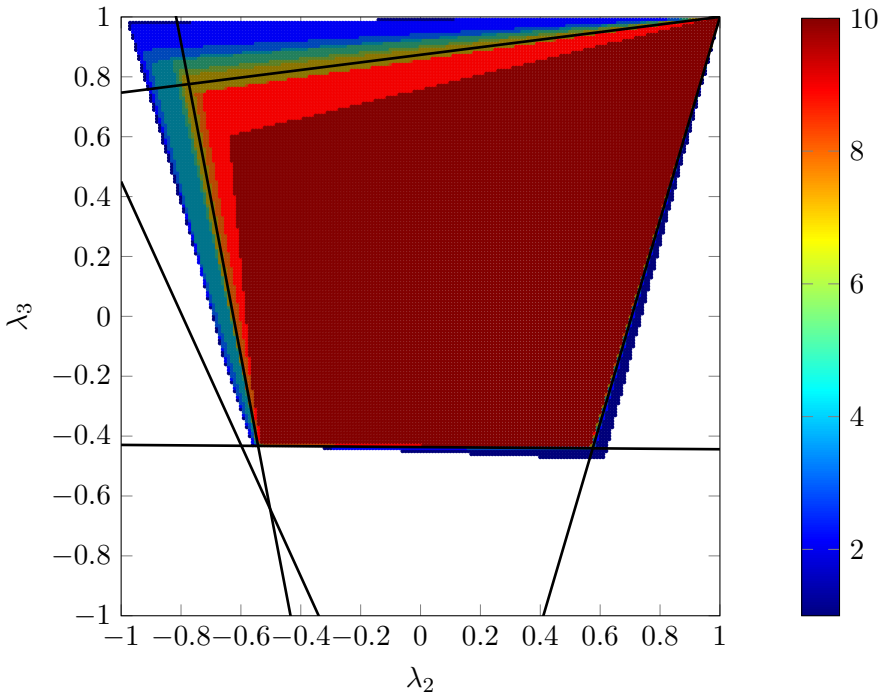
degree 5. This value is chosen not too high to prevent signals to be stable before observation. Then, from these diffused signals, we compute $\widetilde{\boldsymbol{\mathcal{X}}_{\mathcal{T}}}$, from which we compute the set $\mathcal{S}$ of constraints including (96), (98) and (108). As for Figure 33, we iterate over the pairs $(\lambda_2, \lambda_3) \in [-1, 1] \times [-1, 1]$ (using a step of 10^{-2}) to identify those that allow inference of a positively-weighted matrix:



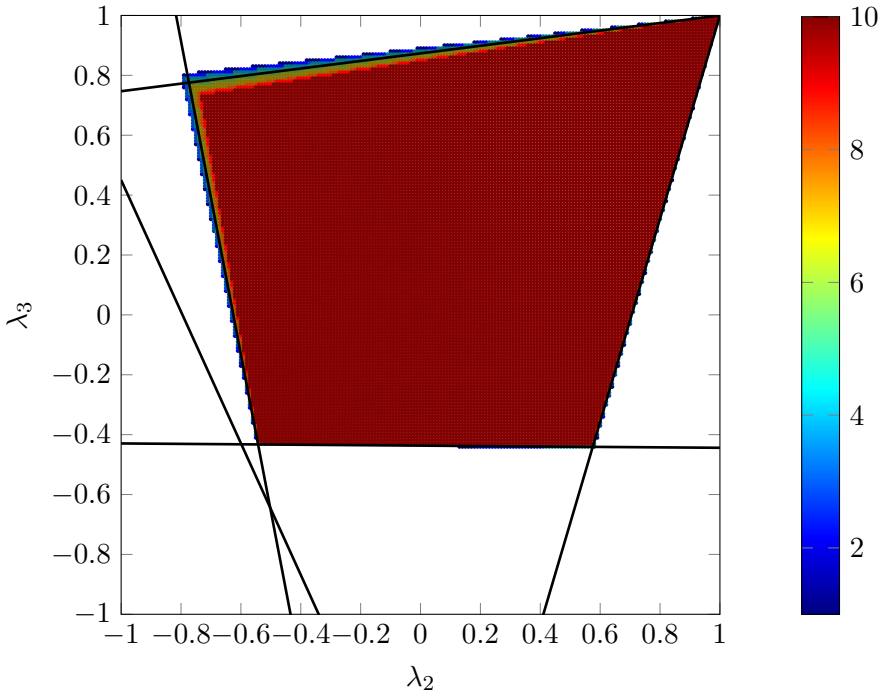
(a) $M = 10$.



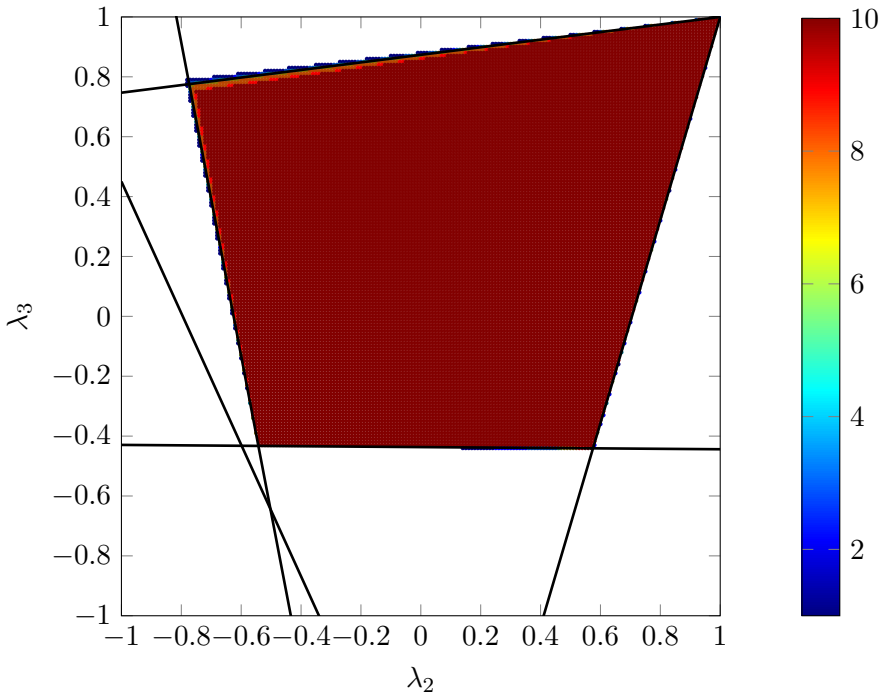
(b) $M = 10^2$.


1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
λ_3
−1 −0.8 −0.6 −0.4 −0.2 0 0.2 0.4 0.6 0.8 1
λ_2
10
8
6
4
2

(c) $M = 10^3$.


1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
λ_3
−1 −0.8 −0.6 −0.4 −0.2 0 0.2 0.4 0.6 0.8 1
λ_2
10
8
6
4
2

(d) $M = 10^4$.



(e) $M = 10^5$.

Figure 34 – Histograms of the number of times a pair (λ_2, λ_3) (using a step of 10^{-2}) allows the inference of a graph shift operator respecting the constraints in $\mathcal{S}$, when used jointly with the eigenvectors of a sample covariance matrix, for 10 such matrices. The positivity constraints in (96) defined from the eigenvectors of $\boldsymbol{\mathcal{T}}$ are depicted using black lines.

As it can be seen from these plots, experimental results confirm the study in Section 3.1.2 about convergence of the eigenvectors of the sample covariance matrix to those of the covariance matrix, as M increases. This indicates that, provided that the eigenvectors of the covariance matrix are approximated with sufficient precision, any particular selection strategy from the ground truth polytope can also apply to the polytope defined from the eigenvectors of the sample covariance matrix.

Selecting a point from the convex set of solutions using a sparsity assumption

Once the set of solutions is delimited using equations corresponding to the assumptions on the underlying graph shift operator, one needs to select a point from this set to use as eigenvalues. As stated in Section 3.3.1, when considering diffusion on a positively weighted graph, the set of solutions contains an infinite number of elements. It is therefore necessary to use additional priors on the matrix to recover.

A first strategy enforcing simplicity of the solution was already studied in Section 3.2.2 (equation (100)) when considering the simplified case of a monomial graph filter. As a matter of fact, numerous graphs are simple, which makes it a very common assumption. In the case of a diffusion process for example, a simple matrix models a process that maximizes signal propagation over a graph.

Another property of broad interest is sparsity of the solution. In many applications one may believe the graph underlying the observations is sparse. Similar to the case when

trying to recover a simple graph, finding a sparse admissible solution can be formulated as a linear program.

Since the inequalities in (96) enforce entries to be greater or equal to zero, any vector of eigenvalues located on one such constraint will cause the inferred matrix to feature at least one null entry. Therefore, in order to find a sparse solution, we seek the set of admissible eigenvalues for which the maximum number of constraints in (96) are null. This reduces to minimizing the $L_{0,1}$ norm of the inferred matrix, which is known to be an NP-hard problem [AK98].

A common approach to circumvent this problem is to approximate the minimizer of the $L_{0,1}$ norm by minimizing the $L_{1,1}$ norm instead [CDS01; GN03; Rin09]. In our case, we use the $L_{1,1}$ matrix norm, which is the sum of all entries, since they are all positive. In this section, we adopt this approach and consider again a linear programming problem as follows:

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}\in\mathbb{R}^N}{\operatorname{argmin}} \ \mathbf{1}_N^\top \boldsymbol{\mathcal{X}} \operatorname{diag}(\boldsymbol{\lambda}) \boldsymbol{\mathcal{X}}^\top \mathbf{1}_N \quad s.t. \quad \boldsymbol{\lambda} \text{ verifies } \mathcal{S} \ , \tag{110}$$

where $\mathcal{S}$ is the set of constraints including (96), (98) and — if needed — (108).

3.3.2 Experiments on a dataset of temperatures in Brittany

Throughout this section, we study an open dataset [1] of temperature observations from 37 weather stations located in Brittany, France [Gir15a]. Figure 35 presents the various locations of these stations:

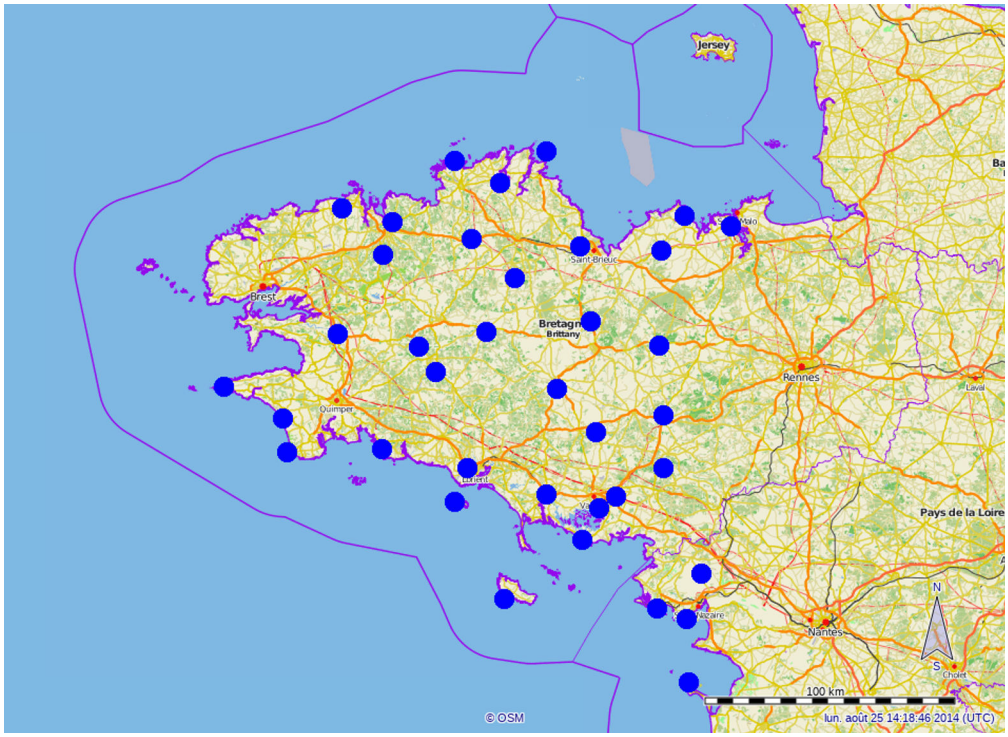


Figure 35 – Locations of the 37 weather stations in Brittany from the dataset in [Gir15a]. These stations can be considered as vertices of a graph, on which signal entries are temperature observations.

1. In http://data.gouv.fr.

Our inference methods in (100) and (110), as well as other existing methods, are evaluated on this dataset in terms of $L_{1,1}$ sparsity, trace of the solution, and smoothness. Since all methods do not impose the same scale on the inferred matrices, these quantities are computed for the inferred diffusion matrices after normalization such that their first eigenvalue equals one, as in constraint (98).

When applying our methods *Simple* (100) and *Sparse* (110), as well as those of Kalofolias [Kal16], Segarra *et al.* [Seg+17a] and the graphical lasso [FHT08] on the dataset of temperatures, we have obtained the results in Table 1:

	$L_{1,1}$	Tr	$\mathrm{S}(\mathbf{X})$
Simple (100)	36.9974	0.0013	0.0551
Sparse (110)	36.9971	0.9093	0.0585
Kalofolias [Kal16]	36.9979	**0**	0.0751
Segarra *et al.* [Seg+17a]	36.9993	1.97×10^{-5}	**0.0245**
Graphical lasso [FHT08]	**35.3539**	13.4977	32.8421

Table 1 – Sparsity, trace and smoothness obtained for the dataset of temperatures. The last column indicates the total smoothness for all signals, *i.e.*, $\mathrm{S}(\mathbf{X}) = \sum_{i=1}^{M} \mathrm{S}(\mathbf{X}[:, i])$.

Interestingly, the methods we introduced in this chapter appear not to perform as well as other graph inference methods. We will see in the Section 3.4.3 that this is not exactly the case.

Still, while the methods we introduced do not provide the best results in Table 1, it is still interesting to remark that the *Simple* method infers a graph with a low trace, as expected. Similarly, the *Sparse* method allow inference of a graph with the second lowest $L_{1,1}$ norm. Also, both methods lead to inference of matrices with a low smoothness value, which is something expected when assuming diffusion of signals, due to the remarks in Section 2.3.2.

3.4 Adaptation of other strategies to stationary signals

The two linear programs introduced in (100) and (110) allow selection of a vector of eigenvalues from a convex set of admissible solutions, delimited by inequalities derived from the eigenvectors of the (sample or not) covariance matrix. They distinguish from each other by the priors they make on the inferred graph, as the first enforces simplicity of the solution, while the second favors sparsity.

In this section, we take a different point of view to select a vector of eigenvalues from the set of admissible solutions. Many graph inference techniques exist (see Section 4.1), all enforcing different properties of the graph that is retrieved. However, most of them do not impose the eigenvectors of the inferred solution to match those of the covariance matrix. The idea here is to adapt these solutions to stationary signals.

3.4.1 Introduction of the method

Let us consider an inference method m providing an adjacency or a Laplacian matrix $\mathbf{M}_m$ from a set of stationary signals $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$. Let $\boldsymbol{\mathcal{T}}_m$ be a matrix derived from

$\mathbf{M}_m$ that represents the diffusion process on the graph (for example, the one derived from the normalized Laplacian). Let $\boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{T}}_m}$ and $\boldsymbol{\lambda}_{\boldsymbol{\mathcal{T}}_m}$ be the eigenvectors and eigenvalues of $\boldsymbol{\mathcal{T}}_m$, and let $\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}$ be the eigenvectors of the covariance matrix.

The idea here is to consider $\boldsymbol{\mathcal{T}}_m$ as if it were expressed in the eigenbasis of $\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}$, to check whether it belongs or not to the polytope of admissible solutions. In other words, we want to find a matrix $\boldsymbol{\Lambda}_m$ such that:

$$\begin{aligned}
\boldsymbol{\mathcal{T}}_m &= \boldsymbol{\mathcal{X}}_m \operatorname{diag}(\boldsymbol{\lambda}_m) \, {\boldsymbol{\mathcal{X}}_m}^\top \\
&= \left(\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}} {\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}}^\top\right) \boldsymbol{\mathcal{X}}_m \operatorname{diag}(\boldsymbol{\lambda}_m) \, {\boldsymbol{\mathcal{X}}_m}^\top \left(\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}} {\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}}^\top\right) \quad \text{// Since } \left(\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}} {\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}}^\top\right) = \mathbf{I}_N \\
&= \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}} \left({\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}}^\top \boldsymbol{\mathcal{X}}_m \operatorname{diag}(\boldsymbol{\lambda}_m) \, {\boldsymbol{\mathcal{X}}_m}^\top \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}\right) {\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}}^\top \\
&= \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}} \, \boldsymbol{\Lambda}_m \, {\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}}^\top
\end{aligned} . \tag{111}$$

Again, using the fact that $\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}$ forms an orthonormal basis, we have:

$$\boldsymbol{\Lambda}_m = {\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}}^\top \boldsymbol{\mathcal{T}}_m \, \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}} \, . \tag{112}$$

Unless eigenvectors of $\boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{T}}_m}$ and $\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}$ are pairwise orthogonal, $\boldsymbol{\Lambda}_m$ is not necessarily a diagonal matrix. Therefore, $\boldsymbol{\Lambda}_m$ lies in a space of dimension N^2, while the polytope is defined by N variables.

Let us call $\operatorname{diag}^{-1}(\boldsymbol{\Lambda}_m)$ the vector of elements on the diagonal of $\boldsymbol{\Lambda}_m$. Since the polytope of admissible diffusion matrices is defined in $\mathbb{R}^N$, $\operatorname{diag}^{-1}(\boldsymbol{\Lambda}_m)$ is the point of this set that forms the best estimate for $\boldsymbol{\Lambda}_m$, defined in $\mathbb{R}^{N^2}$, after dimensionality reduction. In other words, $\operatorname{diag}^{-1}(\boldsymbol{\Lambda}_m)$ is the orthogonal projection of $\boldsymbol{\Lambda}_m$ in the dimensions of the polytope. If $\operatorname{diag}^{-1}(\boldsymbol{\Lambda}_m)$ does not belong to the polytope of admissible diffusion matrices characterized by $\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}$, then the method m provides a solution that does not satisfy the conditions to be a diffusion process. To find the point in the polytope that is the closest to $\operatorname{diag}^{-1}(\boldsymbol{\Lambda}_m)$ in the sense of the Euclidean norm, we solve the following problem:

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda} \in \mathbb{R}^N}{\operatorname{argmin}} \left\| \boldsymbol{\lambda} - \operatorname{diag}^{-1}(\boldsymbol{\Lambda}_m) \right\|_2 \quad \textit{s.t.} \quad \boldsymbol{\lambda} \text{ verifies } \mathcal{S} \, , \tag{113}$$

where $\mathcal{S}$ is the set of constraints describing properties of the matrix to infer, *i.e.*, (96), (98) and — if needed — (108) in the case of heat diffusion.

The solution to (113) gives us a set of eigenvalues $\boldsymbol{\lambda}^*$ that represent the best approximation of $\boldsymbol{\Lambda}_m$ when restricting the search to diffusion matrices. Therefore, the matrix:

$$\boldsymbol{\mathcal{T}}_m^* = \boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}} \operatorname{diag}(\boldsymbol{\lambda}^*) \, {\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}}^\top \tag{114}$$

is the adaptation of the solution of method m to stationary signals.

We measure the distance between $\boldsymbol{\Lambda}_m$, the projection of $\boldsymbol{\mathcal{T}}_m$ in the space defined by $\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}}$, and $\boldsymbol{\lambda}^*$, the closest point in the polytope, as follows:

$$\left\| \boldsymbol{\Lambda}_m - \operatorname{diag}(\boldsymbol{\lambda}^*) \right\|_F \, . \tag{115}$$

Figure 36 provides a graphical illustration of the various steps to compute this distance:

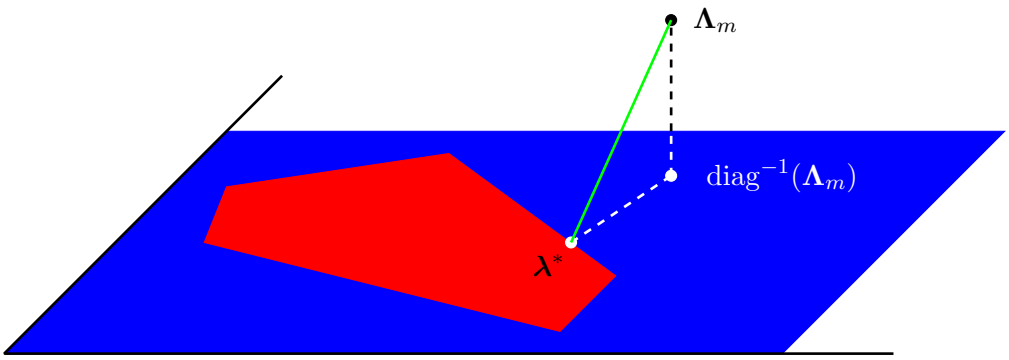


Figure 36 – Correction of the result of an inference method m to match the stationarity hypothesis on the observed signals. The eigenvalues of the result of m are expressed in the space defined by $\boldsymbol{\mathcal{X}}_{\mathbf{\Sigma}}$ as a matrix $\mathbf{\Lambda}_m$. Then, $\mathbf{\Lambda}_m$ is approximated by $\mathrm{diag}^{-1}(\mathbf{\Lambda}_m)$, its orthogonal projection in the space of the polytope. The closest point in the polytope (in the sense of the Euclidean norm), $\boldsymbol{\lambda}^*$, is then found by solving (113). Finally, the distance between $\mathbf{\Lambda}_m$ and its estimate in the polytope is given by the measure in (115). This corresponds to the norm of the vector in green.

3.4.2 Application to the method from Kalofolias

This section explores the application of the regularization strategy in Section 3.4.1 to the method of Kalofolias [Kal16], and shows that it provides matrices that do not belong to the polytope of solutions. To correct this, the closest point in the polytope is therefore considered, and evaluation on a dataset shows that the result has interesting similarities with the original matrix.

The method from Kalofolias has two major qualities: it recovers a graph in a very short amount of time, and encourages smoothness of the solution, which can be a desirable property. To evaluate whether the retrieved solution happens to match a diffusion process, let us consider the following experiment:

1. Let $\mathcal{G}$ be a random geometric graph of $N = 10$ vertices ($R = 0.6$), and let $\boldsymbol{\mathcal{T}}$ be the diffusion matrix associated with its normalized Laplacian. Using this matrix, we diffuse $N = 10^6$ *i.i.d.* signals using various graph filters of variable $\boldsymbol{\mathcal{T}}$ to obtain a matrix $\mathbf{X}$. Using Principal Component Analysis on $\mathbf{X}$ [Pea01], we obtain $\widetilde{\boldsymbol{\mathcal{X}}_{\mathbf{\Sigma}}}$, an estimate for the eigenvectors of $\boldsymbol{\mathcal{T}}$. This set of eigenvectors yields a polytope of admissible solutions;
2. Then, we use the method from Kalofolias to infer a graph $\mathcal{G}_K$ from $\mathbf{X}$, and compute the associated matrix $\boldsymbol{\mathcal{T}}_K$. Since the log method from Kalofolias depends on parameters α and β, we keep the minimal distance obtained for values of α and β ranging from 0.01 to 2, with a step of 10^{-2}. Equation (115) gives us the distance between the polytope and the inferred solution;
3. Additionally, we generate a random geometric graph $\mathcal{G}_{rg}$ (independent from the ground truth one), using the same settings as for $\mathcal{G}$ ($N = 10$, $R = 0.6$). This gives us a baseline of how close a random graph with the same edges distribution can be to the ground truth one, and gives information on whether the results of Kalofolias are closer to the ground truth than a random matrix. Again, (115) measures the distance between the polytope and the associated matrix $\boldsymbol{\mathcal{T}}_{rg}$.

We perform these three steps for 10^5 occurrences of random geometric graphs. Let $\mathbf{d}_K$ be the vector of distances (115) to the polytope obtained for each ground truth graph using the method of Kalofolias, and $\mathbf{d}_{rg}$ the vector of distances to the polytope for the baseline random graphs. In Figure 37, we plot a histogram of the number of times each distance was observed:

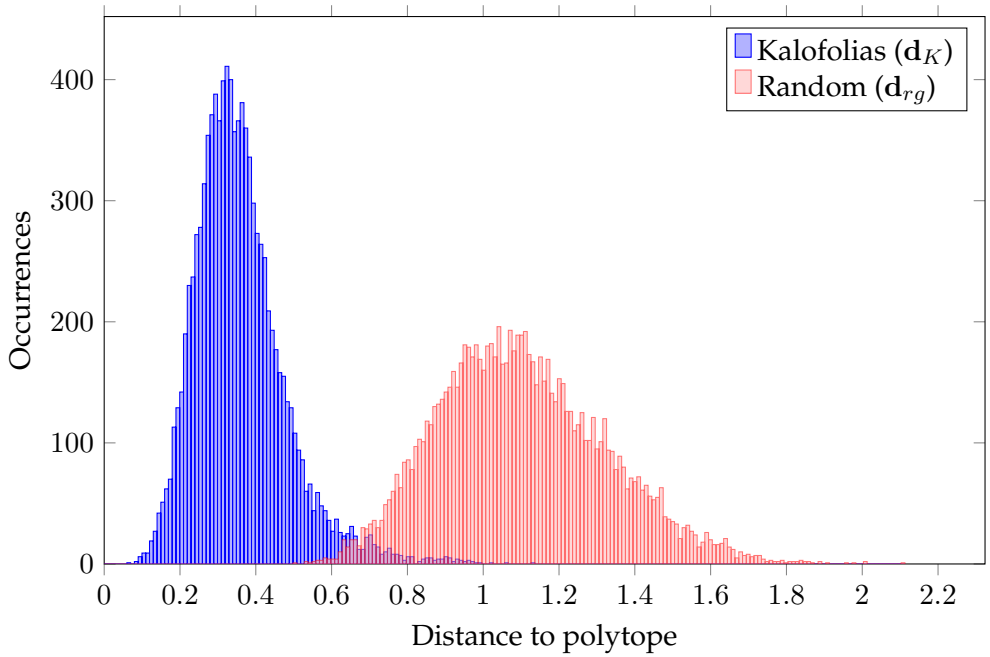


Figure 37 – Number of times a distance (115) to the ground truth polytope was observed using either the method from Kalofolias [Kal16] ($\mathbf{d}_K$), or a random geometric graph ($\mathbf{d}_{rg}$). Distances are grouped in bins of size 10^{-2}. Tests were performed for 10^5 occurrences of graphs per method, with $M = 10^6$ signals.

From these results, a first observation is that neither the methods from Kalofolias nor the random method ever returned a graph that was located in the polytope of solutions. Two direct interpretations of this result can be made: first, it implies that the set of admissible matrices per ground truth graph is small relatively to the set of random graphs. Second, it implies that the method from Kalofolias does not succeed in recovering a graph that matches diffusion priors on the signals.

Mann-Whitney U test [MW47] on $\mathbf{d}_K$ and $\mathbf{d}_{rg}$ shows that the distributions differ significantly ($U = 9.9813 \times 10^7$, $P < 10^{-5}$ two-tailed). This implies that the results obtained with the method from Kalofolias are most of the time closer to an admissible matrix than random solutions. This observation can be explained by the remarks in Section 2.3.2. Diffusion of signals on a graph tends to smoothen them, as the low frequencies are attenuated slower than higher ones. Since the method of Kalofolias retrieves a graph on which signals are smooth, the observation that it provides solutions that are closer to the polytope than random solutions is quite natural.

The question is then whether the closest point to the retrieved solution in the polytope has interesting properties. Let us evaluate this solution on the dataset of temperature introduced in Section 3.3.2. Figure 38 depicts the 10% most significant connections in the adjacency matrix of the graph $\mathcal{G}_K$ retrieved by Kalofolias, as well as those of the matrix associated with the closest point in the polytope in (114), where $\boldsymbol{\lambda}^*$ is the solution to (113):

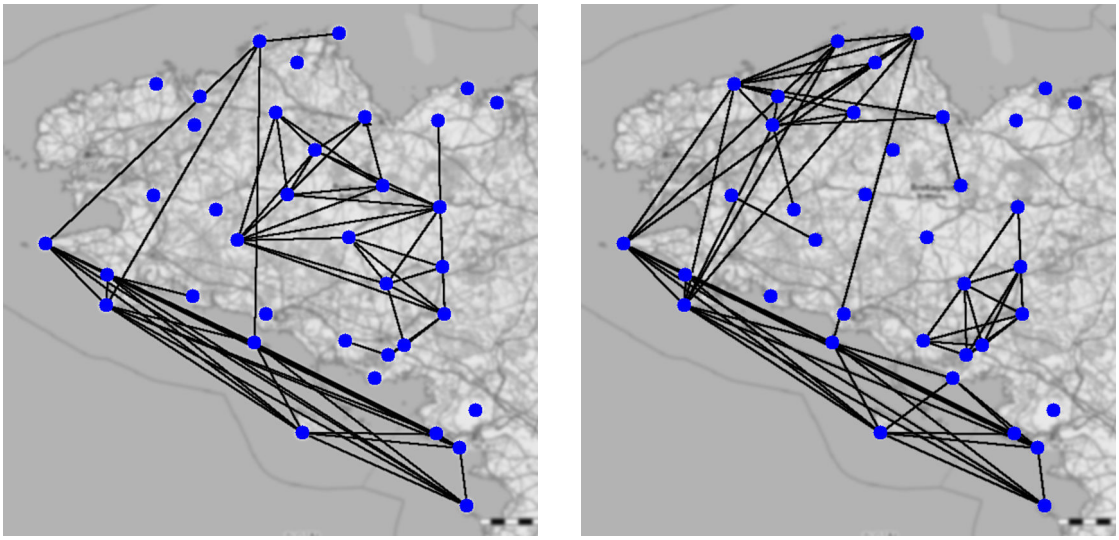


Figure 38 – Most significant connections in the adjacency matrix of the graph $\mathcal{G}_K$ retrieved by the method from Kalofolias (left), and most significant connections from the matrix associated with the closest point in the polytope, $\mathcal{T}^*$ (right).

The method from Kalofolias retrieves a matrix that has stronger connections between stations with similar locations. There is a strong connectivity among stations located on the south coast of Brittany. Stations located more in the land also tend to be linked to close inland stations. The regularized matrix appears to keep these properties: the strong links on the coasts still appear, and the result also still gives importance on the coastal versus inland aspect of the stations. Still, differences can be seen, as the regularized matrix appears to give more importance to the relations between stations on the north coasts. Such relations also exist in the original matrix, but are not depicted due to the threshold.

When computing the total smoothness of the signals with both matrices, we obtain that the solution from Kalofolias has a higher total smoothness $\left(\sum_{i=1}^{M} \mathrm{S}(\mathbf{X}[:,i]) = 0.0751\right)$ than the closest point in the polytope $\left(\sum_{i=1}^{M} \mathrm{S}(\mathbf{X}[:,i]) = 0.0548\right)$. This implies that the signals are smoother on the approximate matrix than on the one recovered by the method of Kalofolias. This may seem counter-intuitive, since the solutions of the method by Kalofolias are not restricted to the polytope. However, the method from Kalofolias imposes inference of a matrix with an empty diagonal, which is not the case of the approximate one.

These measurements, in addition to those below, suggest that inferring a graph using the method from Kalofolias, and considering the closest point in the polytope, is an interesting method to infer a valid graph on which signals are smooth.

3.4.3 Additional experiments on the dataset of temperatures

Let us consider again the dataset of temperatures introduced in Section 3.3.2. In these first experiments, it appeared that other graph inference methods performed better than the methods (100) and (110) we introduced in this chapter. In these additional experiments, we enrich Table 1 with the closest solutions from the polytope using the method in Section 3.4.1, as we detailed in Section 3.4.2 for the method of Kalofolias [Kal16]. The obtained results are given in Table 2:

	polytope	$L_{1,1}$	Tr	$\mathrm{S}(\mathbf{X})$
Simple (100)	✓	36.9974	$\mathbf{0.0013}$	0.0551
Sparse (110)	✓	$\mathbf{36.9971}$	0.9093	0.0585
Kalofolias [Kal16]	0.0313	36.9979	0	0.0751
Kalofolias closest	✓	36.9974	0.0298	0.0548
Segarra *et al.* [Seg+17a]	0.0062	36.9993	1.97×10^{-5}	0.0245
Segarra *et al.* closest	✓	36.9974	0.0046	0.0551
Graphical lasso [FHT08]	1.3730	35.3539	13.4977	32.8421
Graphical lasso closest	✓	36.9984	13.3584	$\mathbf{0.0335}$

Table 2 – Sparsity, trace and smoothness obtained for the dataset of temperatures. Elements in bold denote the method performing best among those that return a solution located in the polytope. If a method provides a solution that does not belong to the polytope, the distance (115) to the closest point is indicated in the first column. The last column indicates the total smoothness for all signals, *i.e.*, $\mathrm{S}(\mathbf{X}) = \sum_{i=1}^{M} \mathrm{S}(\mathbf{X}[:, i])$.

First, we notice that the method from Segarra *et al.* [Seg+17a] returns a matrix that is at a distance of 0.0062 from the polytope. As the polytope description is the same for their method and for ours, we would expect this distance to be 0. This small difference comes from their implementation. In order to keep their equality constraints enforcing the elements in the polytope to have an empty diagonal, while coping with the noise in the eigenvectors, they allow small deviations from the polytope. In more details, they do not return a matrix $\mathbf{S}^*$ (see (89)) that shares the eigenvectors of the covariance matrix, but a matrix $\mathbf{S}^*_{\text{approx}}$ such that $\left\|\mathbf{S}^* - \mathbf{S}^*_{\text{approx}}\right\|_F \leq \varepsilon$. Here, experiments were performed for $\varepsilon = 10^{-3}$. For this reason, the matrix they return can be located slightly outside of the polytope of solutions. When considering the closest point to this result in the polytope, it appears to be very close to the solution returned by the *Simple* method.

Contrary to the experiments in Section 3.3.2, we consider here the best solutions among those that return a point located in the polytope. Under these settings, the *Sparse* method recovers the matrix with the lowest $L_{1,1}$ norm. It is also interesting to remark that the projection of the solution of the graphical lasso on the polytope is smoother than the projection of the solution obtained by the method from Kalofolias. This echoes the remark in Section 3.1.3 that minimization of the quantity $\mathrm{Tr}(\widetilde{\mathbf{\Sigma}}\mathbf{\Theta})$ in (87) tends to promote smoothness of the signals on the graph when $\mathbf{\Theta}$ is a Laplacian matrix, which appears to be encouraged by the regularization algorithm. Note that the method from Kalofolias infers a graph which projection on the polytope gets the second best smoothness score, while having a small trace. On the other hand, the solution inferred by the graphical lasso appears to have most of its energy on the diagonal entries. This is confirmed by the traces of the matrices, both for the original solution and its approximate in the polytope. Therefore, these two solutions provide interesting ways to find a graph on which stationary signals are smooth, with different simplicity assumptions.

3.4.4 Application of regularization to graph hypothesis testing

To evaluate the practical interest of the regularization strategy, let us consider the situation where some signals are observed, and various graph shift operators are provided

by inference methods to explain these signals. The objective is to determine which of the proposed solutions matches the signals best under a stationarity assumption.

In the following experiment, we proceed as follows: let $(\boldsymbol{\mathcal{T}}_1 \dots \boldsymbol{\mathcal{T}}_{20})$ be a set of 20 diffusion matrices corresponding to graphs of $N = 10$ vertices, equally divided into random geometric graphs (with R drawn uniformly in $[0.2, 0.6]$) and Erdős-Rényi graphs (with P drawn uniformly in $[0.2, 0.6]$). For each of these matrices $\boldsymbol{\mathcal{T}}_i$, let us diffuse M random signals using various graph filters of variable $\boldsymbol{\mathcal{T}}_i$ to obtain observations $(\mathbf{X}_1 \dots \mathbf{X}_{20})$, where $\mathbf{X}_i$ is the set of signals obtained after diffusion by $\boldsymbol{\mathcal{T}}_i$. From these sets, we can compute the eigenvectors of the sample covariance matrices $\left(\widetilde{\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}_1}} \dots \widetilde{\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}_{20}}}\right)$.

For each matrix of eigenvectors $\widetilde{\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}_i}}, i \in [\![1, 20]\!]$, and for each diffusion matrix $\boldsymbol{\mathcal{T}}_j, j \in [\![1, 20]\!]$, we compute the distance between the polytope yielded by $\widetilde{\boldsymbol{\mathcal{X}}_{\boldsymbol{\Sigma}_i}}$ and the projection of $\boldsymbol{\mathcal{T}}_j$ in the space of the polytope (see Section 3.4.1) using (115). The graph that minimizes the distance is then selected as the most appropriate. Figure 39 depicts the ratio of times $\boldsymbol{\mathcal{T}}_i$ is selected as the most appropriate diffusion matrix when considering signals $\mathbf{X}_i$, for various values of M:

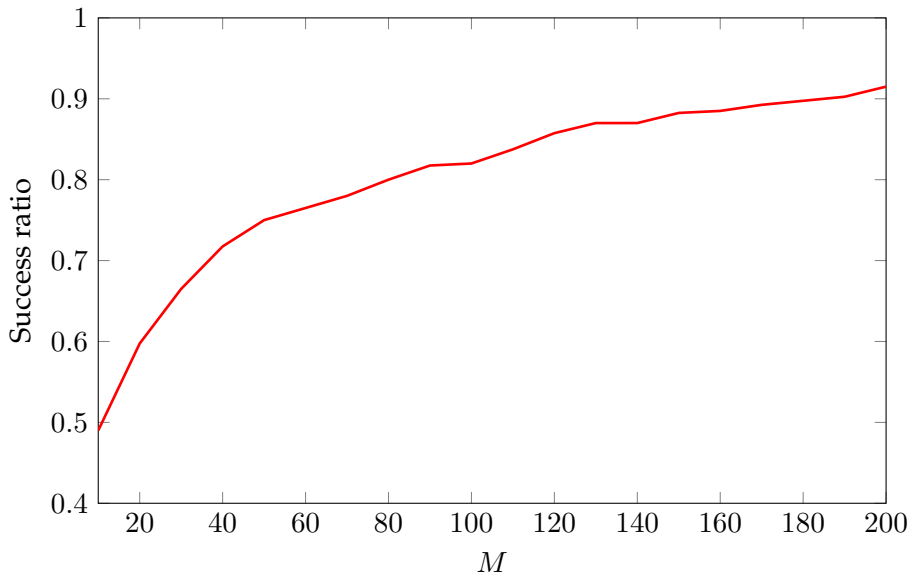


Figure 39 – Ratio of times when the diffusion matrix $\boldsymbol{\mathcal{T}}_i$ was chosen by the algorithm as the most adapted to signals $\mathbf{X}_i$ among a set of 20 possible diffusion matrices, for $i \in [\![1, 20]\!]$. Mean results for 100 iterations of the experiment.

The results show that the regularization strategy selects the matrix used to diffuse the signals in most cases, even when M is low. Additional experiments were performed for larger graphs, and similar results were observed. Also, increasing the number of signals eventually leads to a selection of the correct diffusion matrix in all cases. This experiment illustrates that the regularization strategy introduced in Section 3.4.1 can be successfully used to select the graph that is the most adapted to given signals among a set of candidates.

An interesting direction for future work includes evaluation of the performance of the method when considering noisy versions of $\boldsymbol{\mathcal{T}}_i$.

3.5 Summary of the chapter

In this chapter, we have presented a methodology to infer a graph from stationary signals, that we summarize as follows:

1. Compute or approximate the eigenvectors of the covariance matrix of the signals. This can be done by diagonalizing the (sample) covariance matrix, or using Principal Component Analysis [Pea01] on the signals;
2. Identify a set of essential properties of the matrix to infer and, with the help of the eigenvectors of the (sample) covariance matrix, create a set $\mathcal{S}$ of constraints encoding these properties. These constraints delimit a convex set of admissible eigenvalues, to use with the eigenvectors of the (sample) covariance matrix to infer a matrix modeling a diffusion process for the signals on the underlying graph. In the case of heat diffusion on graphs using the matrix $\mathcal{T}$ associated with the normalized Laplacian matrix, such properties are positivity of the entries (constraints (96)) and some additional information on the eigenvalues of such matrices (constraints (98) and (108));
3. Using additional priors on the matrix to infer, select a vector of eigenvalues such that the constraints $\mathcal{S}$ are satisfied. In this chapter, we have presented two priors, namely simplicity of the solution in (100) and sparsity of the matrix in (110).

Additionally to this methodology, we have introduced in this chapter a technique to correct the solutions of other graph inference methods, in order to make them applicable to stationary signals. We have shown that this particular method could be useful to select the most appropriate matrix among a set of candidates, to model a diffusion process for given signals on an unknown graph.

Also, experiments on a dataset of temperatures in Brittany have experimentally shown that this correction appears to keep the particularities of these methods, allowing the definition of strategies with new priors to select eigenvalues from the polytope. Additional experiments on synthetic data can be found in [Pas+17a], and present with more details the performance of the methods we introduced in this chapter.

There are numerous possible extensions to this work:

- Obtainment of a good estimate for the eigenvectors of the covariance matrix is a cornerstone of our approach. In this chapter, we have considered the sample covariance estimator only. A complete review of covariance estimation techniques would therefore be very interesting to identify those that favor convergence of the eigenvectors;
- We could also explore new strategies to select a point in the polytope, for example by enforcing the reconstruction of a binary matrix;
- Definition of the polytope requires $\frac{N(N+1)}{2}$ constraints in (96). While we have experimentally observed in Section 3.2.4 that some of these constraints were unnecessary, a more formal identification of these constraints would lead to acceleration of the solvers used;
- In Section 3.4.4, we have shown that the regularization strategy could select the ground truth matrix among a set of candidates to represent a diffusion process for given signals. An interesting extension of this work would be to consider identification of noisy versions of the ground truth matrix.

Chapter 4

From graphs to translations

Contents

Translation of signals is a natural operation when the signals are evolving on an Euclidean space. As examples, translating a signal of τ seconds in time falls down to replacing every signal entry $s(t)$ with the entry at instant $t \pm \tau$, where the sign indicates direction in time (see Definition 40). Similarly, translating a signal representing an image can be done by sending every pixel up, down, left or right using the information that the image is two-dimensional.

These natural translations can intuitively be ported to graphs, since periodical time can conveniently be represented using a ring graph, and since a periodical pixel grid can be represented using a torus. The possible directions for Euclidean translations of signals on such graphs are given by the *graph orientations* in Figure 40:

Definition 65: Orientation of a graph

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be an undirected graph. An ***orientation*** of $\mathcal{G}$ is a digraph $\overrightarrow{\mathcal{G}} = \langle \mathcal{V}, \overrightarrow{\mathcal{E}} \rangle$ such that $\forall (v_1, v_2) \in \overrightarrow{\mathcal{E}} : \{v_1, v_2\} \in \mathcal{E}$.

We choose to represent an orientation $\overrightarrow{\mathcal{G}}$ of $\mathcal{G}$ by a graph in which the edges of $\mathcal{E}$ are depicted with dotted lines, and diedges of $\overrightarrow{\mathcal{E}}$ with red arrows.

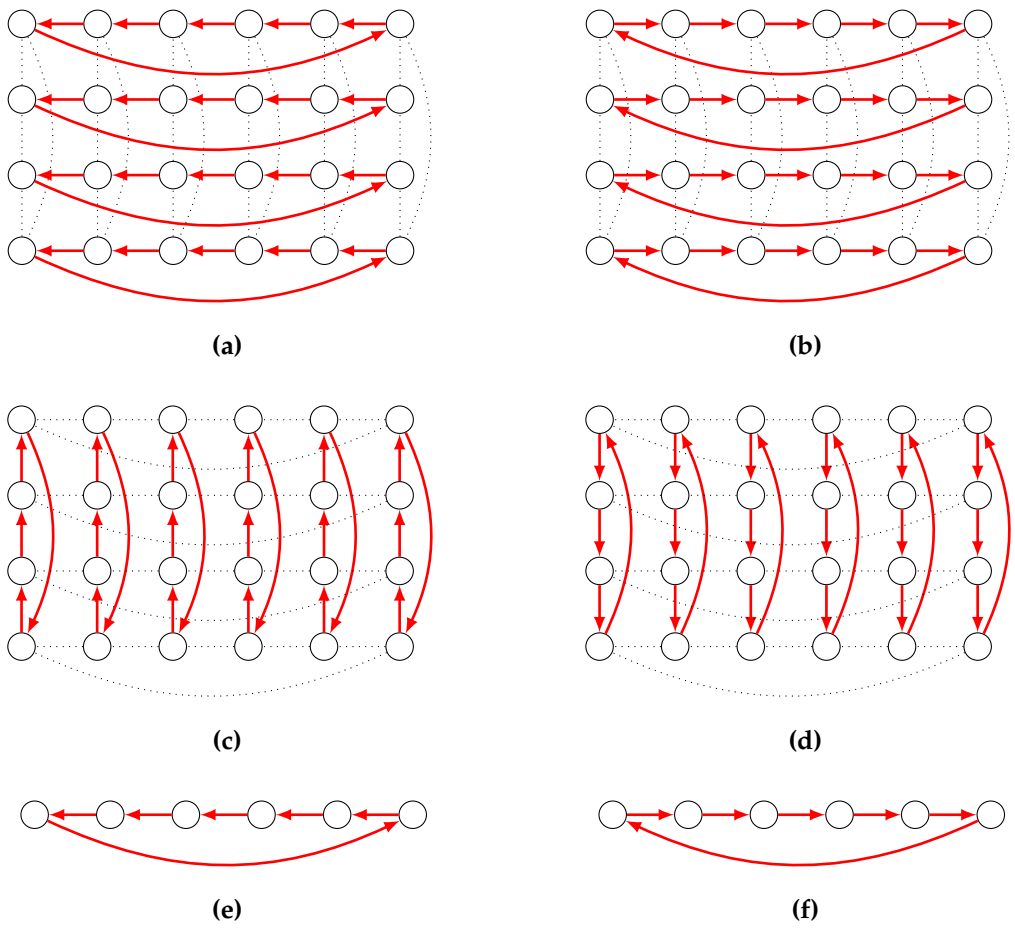
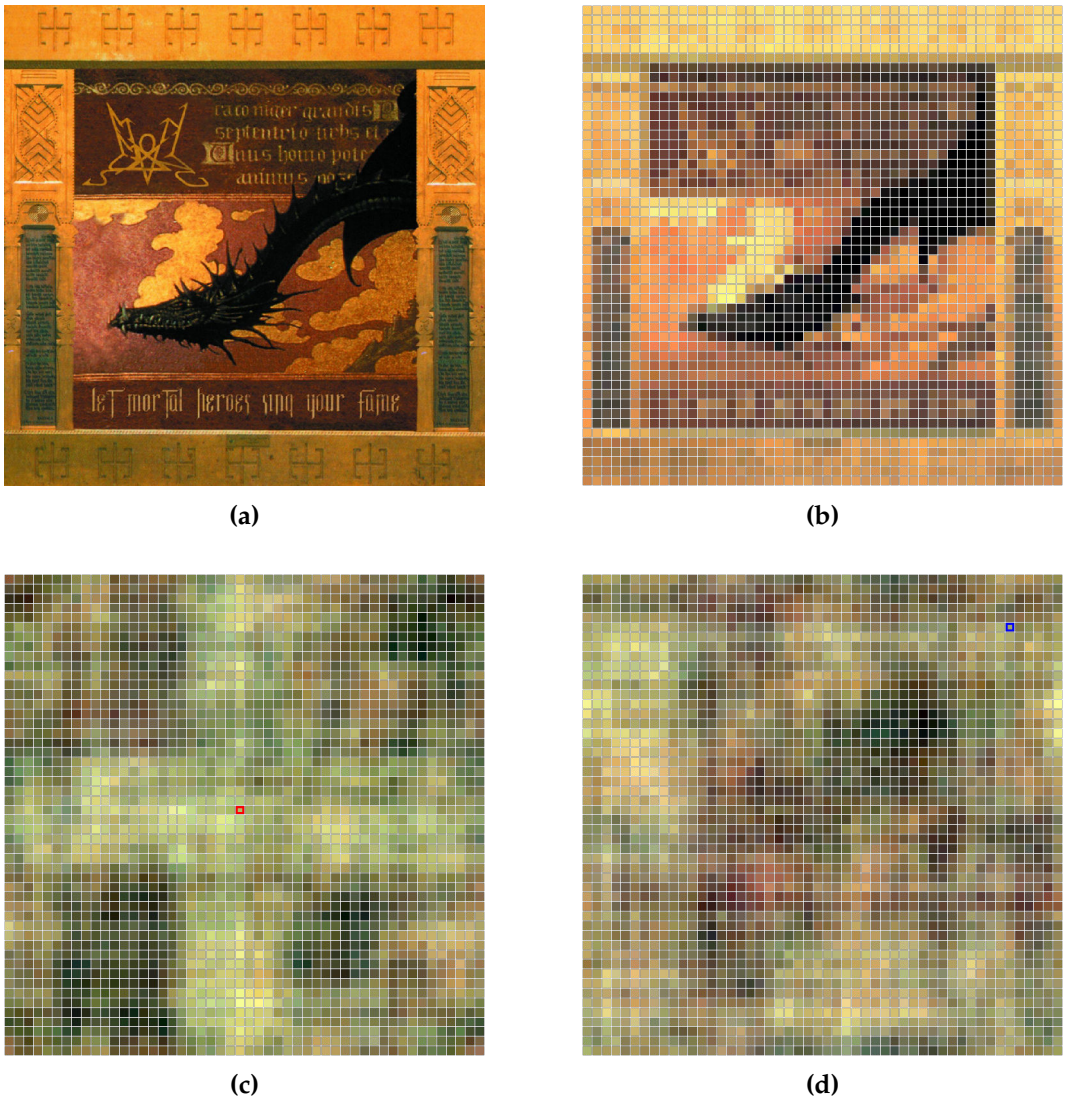


Figure 40 – Natural translations on a two-dimensional torus graph of dimensions $\mathbf{d} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$ **(a-d)**, and on a ring graph of $N = 6$ vertices **(e-f)**. These translations use the underlying metric space, as for every vertex of coordinates $\mathbf{v}$, the vertex of coordinates $\mathbf{v} +_{\mathbf{d}[i]} \boldsymbol{\delta}_i$ exists, where $\boldsymbol{\delta}_i$ is the Dirac vector of i^{th} non-null entry (see Section 2.1.4).

However, graphs in general do not have an associated Euclidean space. Also, if we consider a grid graph and move its vertices to random locations, this does not change its adjacency matrix, which makes it strictly equivalent to the regular grid graph.

For this reason, translations on graphs have been defined not to take this underlying metric space into account. As introduced in Section 2.3.1, Shuman *et al.* in [Shu+13] uses the property that translation of time signals is equivalent to convolution with a Dirac signal located at a particular time instant. Analogously, translations on graphs consist in a convolution with a Dirac signal located on a target vertex, thus localizing the translated signal around this particular vertex.

While this definition of convolution allows to move localized signals to various places of a graph, thus providing a fine tool for filtering operations, it does not apply well to signals that have a certain local coherence. As an example, Figure 41 depicts an image on a two-dimensional torus, which is then translated to a particular vertex:



(a) **(b)** **(c)** **(d)**

Figure 41 – Original album cover for *Let Mortal Heroes Sing Your Fame* by Summoning **(a)**, and its representation as a signal on a two-dimensional torus graph of dimensions $\mathbf{d} = \begin{bmatrix} 50 \\ 50 \end{bmatrix}$ **(b)** (edges are not depicted for readability). Images **(c)** and **(d)** represent localizations of the original signal to the vertices circled in red and blue, respectively.

This experiment illustrates that translations as defined in Definition 47 do not match the classical definition of translations on metric spaces. As a matter of fact, most signals have interesting neighborhood properties, that are not kept when convolving it with a Dirac signal. Additionally, performing such operation allows modification of the signal entries when translating it, which does not occur in classical translation.

In this chapter, we are interested in finding a way to infer translations on a graph that match classical Euclidean translations, without using information on the underlying metric space. Such translations should offer multiple properties:

1. The translation of a localized signal should be localized;
2. In regular cases, translating a signal to a vertex should have similar effect to moving the observer's point of view to this same vertex.

In previous work several definitions have been proposed, but none of them satisfy both 1. and 2. Here, we introduce results from two of our works [Gre+16; Pas+17b] that define translations based on neighborhood preservation properties. In more details, the chapter is organized as follows:

1. First, we review existing work on translations on graphs;
2. Then, we introduce numerous definitions, and propose a characterization of translations on graphs that matches the properties above;
3. In a third part, we show some properties on these translations, and demonstrate that they are equivalent to classical Euclidean translations on the ring and on the torus. Among other results, we show that identification of desired translations on an arbitrary graph is an NP-complete problem, and we propose a relaxation of the problem to approximate them;
4. Finally, we discuss the identification of translations on small variations of regular graphs, as well as on randomly generated graph.

4.1 Existing translation operators on graphs

4.1.1 The graph shift approach

Studying the case of the ring graph, Püschel and Moura [PM06], followed by Sandrihaila and Moura [SM13], propose to use a graph shift operator as a translation operator, and choose the adjacency matrix of the graph on which signals are defined. In particular, when considering the directed ring graph — *i.e.*, the orientation of all edges of the ring graph in the same direction as in Figure 40 **(e-f)** — as a graph shift, multiplication of a signal by this shift has the effect to *advance* it in time.

In the general case, considering an adjacency matrix as a translation operator has the effect to *diffuse* a signal as it is translated. Note that this is also the case where the adjacency matrix is normalized by its eigenvalue with the highest magnitude [SM14], in which case the signal energy only decreases as it is translated (see Section 2.3.2).

For this reason, translation of signals with this approach cannot match our objective of conserving the signal entries during translation. However, our approach is similar to this one in the sense that we identify translations by finding a subset of non-null entries of the adjacency matrix, which can be interpreted as an orientation of a subset of edges in the graph. In particular, the directed ring graph is a valid translation on the ring graph according to our definitions.

4.1.2 The convolutive approach

In the context of applying wavelets to graph signals, Hammond *et al.* [HVG11] propose to define translation as a localization function to move a wavelet at a particular location of the graph. This is done by applying the wavelet to an *impulse*, *i.e.*, a signal that has all its energy concentrated at a single vertex.

As already introduced in Definition 47, the same approach is taken by Shuman *et al.* [SRV12; Shu+13], who propose a definition of translation of a signal *to a vertex* v, by convolution of this signal with an impulse located on v. This is done with analogy to the classical result in Fourier analysis that states that convolution in the time domain (in our case the graph) is equivalent to multiplication in the frequency domain (in our case the spectral domain of the graph).

With this approach, the signal is moved to a particular location rather than by a certain quantity. The convolution operation does not take the neighborhood in consideration, and allows modification of the signal when translating it, as shown in Figure 41.

4.1.3 The isometric approach

Girault *et al.* [GGF15; Gir15a] propose a translation operator for graphs that is isometric with respect to the ℓ_2 norm, *i.e.*, that does not change the signal energy as it is translated. Their approach consists in changing the phase of the signal in the spectral domain to move it in the graph domain. Additionally to keeping the signal norm unchanged, this operator has the property to preserve the signal localization, *i.e.*, to have its energy located around a target vertex [Gir+16].

This approach can also be considered as convolutive, since the translation is performed by convolving the signal with complex exponentials. Therefore, it suffers from the same drawback as the method introduced before.

Gavili and Zhang [GZ15] take a similar direction, and propose a phase change for translation. Contrary to the approach of Girault *et al.*, their solution does not consider the graph spectrum. Again, this method does not have preserve neighborhoods.

4.1.4 Neighborhood-preserving translations

The translation operators we introduce in this chapter have first been defined in [Gre+16]. In this first work, we have explored translations on grid and torus graphs, and have shown that Euclidean translations of images are equivalent to neighborhood preserving properties on these graphs.

In [Pas+17b], we have reformulated and extended the results in [Gre+16] to make them more general and comprehensive. Additionally, we have provided properties of our translations, and have shown that identifying them is an NP-complete problem. Then, we have proposed a relaxation of this problem to identify *pseudo-translations*, and have illustrated our results on grid graphs as well as on graphs following a random model.

This chapter gives the details of these two articles.

4.2 Transformations, translations and isometries on graph

In this section, we introduce some functions on graphs. In particular, we define transformations on graphs with various properties, and introduce translations as transformations with particular neighborhood-preserving properties.

4.2.1 Transformations on graphs

Let us consider a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. Additionally, we introduce an element $\perp$ such that $\perp \notin \mathcal{V}$. We define a *transformation* on a graph as follows:

Definition 66: Transformation on a graph

A ***transformation*** on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is a function $\phi : \mathcal{V} \to \mathcal{V} \cup \{\perp\}$ such that:

$$\forall v_1, v_2 \in \mathcal{V} : (\phi(v_1) = \phi(v_2) \neq \perp) \Rightarrow (v_1 = v_2) \ . \tag{116}$$

We denote the set of transformations on $\mathcal{G}$ by $\Phi_{\mathcal{G}}$.

Informally, a transformation $\phi \in \Phi_{\mathcal{G}}$ on a graph is a function that is injective for every vertex whose image is not $\perp$.

The number of vertices that have their image equal to $\perp$ by a transformation give the loss of this transformation:

Definition 67: Loss of a transformation

We call ***loss of a transformation*** $\phi \in \Phi_{\mathcal{G}}$ on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ the quantity:

$$loss(\phi) = |\{v \in \mathcal{V} \mid \phi(v) = \perp\}| \ . \tag{117}$$

In the case where $loss(\phi) = 0$, we say that ϕ is ***lossless***, and we note it ϕ^*. We denote the set of lossless transformations on $\mathcal{G}$ by $\Phi^*_{\mathcal{G}}$.

In the case of lossless transformations, every vertex has an image in $\mathcal{V}$. Therefore, they are bijective from $\mathcal{V}$ to $\mathcal{V}$.

It is also interesting to notice that every graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ admits a transformation of loss N, defined as follows:

$$\phi_\perp : \left\{ \begin{array}{ccc} \mathcal{V} & \to & \mathcal{V} \cup \{\perp\} \\ v & \mapsto & \perp \end{array} \right. \ . \tag{118}$$

Note that transformations do not take into consideration the edges of the graph. To add the constraint that vertices should be mapped to vertices in their neighborhood, we introduce edge-constrained transformations:

Definition 68: Edge-constrained (EC) transformation

A transformation $\phi \in \Phi_{\mathcal{G}}$ on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is said to be ***edge-constrained*** if it is a function $\phi : \mathcal{V} \to \mathcal{V} \cup \{\perp\}$ such that:

$$(\{v, \phi(v)\} \in \mathcal{E}) \vee (\phi(v) = \perp) \ . \tag{119}$$

We denote the set of EC transformations on $\mathcal{G}$ by $\mathrm{EC}(\Phi_{\mathcal{G}})$, and the set of lossless EC transformations on $\mathcal{G}$ by $\mathrm{EC}\left(\Phi^*_{\mathcal{G}}\right)$.

Since EC transformations restrict the set of image vertices to the set of neighboring vertices, an EC transformation can be represented by an orientation of the graph on which it is defined:

Proposition 5

An EC transformation $\phi \in \mathrm{EC}(\Phi_{\mathcal{G}})$ on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ injectively defines an orientation of $\mathcal{G}$, with the possible exclusion of vertices of which image is $\perp$.

Proof: Proposition 5

Let $\mathbf{A}$ be the adjacency matrix of $\mathcal{G}$, and let $\mathbf{A}_\phi$ be the following $N \times N$ matrix:

$$\forall v_1, v_2 \in \mathcal{V} : \mathbf{A}_\phi[v_1, v_2] = \begin{cases} 1 & \text{if } v_2 = \phi(v_1) \\ 0 & \text{otherwise} \end{cases} . \tag{120}$$

Remind that $\mathbf{A}$ and $\mathbf{A}_\phi$ take their values in $\{0, 1\}$. First, we show that $\forall v_1, v_2 \in \mathcal{V} : \mathbf{A}_\phi[v_1, v_2] \leq \mathbf{A}[v_1, v_2]$. Let us consider a vertex $v_1 \in \mathcal{V}$. There are three possible cases:

1. $\phi(v_1) = \perp$. In that case, $\forall v_2 \in \mathcal{V} : \mathbf{A}_\phi[v_1, v_2] = 0$;
2. $\exists v_2 \in \mathcal{N}(v_1) : (v_2 = \phi(v_1)) \wedge (v_1 = \phi(v_2))$. In that case, $\mathbf{A}_\phi[v_1, v_2] = \mathbf{A}_\phi[v_2, v_1] = \mathbf{A}[v_1, v_2] = 1$, and $\forall v_3 \in \mathcal{V}, v_3 \neq v_2 : \mathbf{A}_\phi[v_1, v_3] = \mathbf{A}_\phi[v_3, v_1] = 0$ (due to the injectivity of ϕ);
3. $\exists v_2 \in \mathcal{N}(v_1) : (v_2 = \phi(v_1)) \wedge (v_1 \neq \phi(v_2))$. In that case, $\mathbf{A}_\phi[v_1, v_2] = \mathbf{A}[v_1, v_2] = 1$, and $\mathbf{A}_\phi[v_2, v_1] < \mathbf{A}[v_2, v_1]$, and $\forall v_3 \in \mathcal{V}, v_3 \neq v_2 : \mathbf{A}_\phi[v_1, v_3] = \mathbf{A}_\phi[v_3, v_1] = 0$ (due to the injectivity of ϕ).

In all cases, entries of $\mathbf{A}_\phi$ are lower or equal than those of $\mathbf{A}$. Additionally, due to case 3), there may exist $v_1, v_2 \in \mathcal{V} : \mathbf{A}_\phi[v_1, v_2] < \mathbf{A}[v_1, v_2]$. Therefore, $\mathbf{A}_\phi$ is not necessarily symmetric, and corresponds to a digraph in which every diedge contains elements that form an edge in $\mathcal{E}$.

Additionally, if $\mathbf{A}_{\phi_1} = \mathbf{A}_{\phi_2}$, then $\forall v \in \mathcal{V} : \phi_1(v) = \phi_2(v)$, *i.e.*, $\phi_1 = \phi_2$. So the mapping $\phi \mapsto \mathbf{A}_\phi$ is injective.

The proof of Proposition 5 shows that it is possible to represent an EC transformation $\phi \in \mathrm{EC}(\Phi_{\mathcal{G}})$ on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ by a digraph $\overrightarrow{\mathcal{G}}^{\phi} = \langle \mathcal{V}, \overrightarrow{\mathcal{E}}^{\phi} \rangle$, with:

$$\forall v \in \mathcal{V} : (\phi(v) \neq \perp) \Leftrightarrow \left((v, \phi(v)) \in \overrightarrow{\mathcal{E}}^{\phi} \right) . \tag{121}$$

This allows a visual representation of EC transformations on a graph, where edges of $\mathcal{E}$ are depicted with dotted lines, on top of which edges of $\overrightarrow{\mathcal{E}}^{\phi}$ are drawn with plain arrows. Additionally, we mark the vertices that have their image being $\perp$ by coloring them in black. Figure 42 depicts an EC transformation on an example graph:

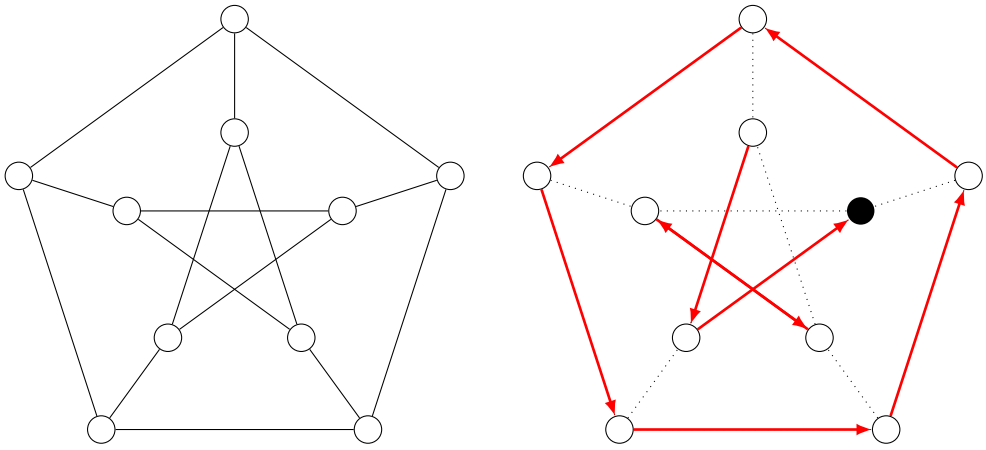


Figure 42 – Example of a graph (left) and an associated EC transformation with loss 1 (right). This graph is the Petersen graph [Pet98].

Using this correspondence with a digraph, we can reformulate the loss of an EC transformation as follows:

Proposition 6

Let $\phi \in \mathrm{EC}(\Phi_{\mathcal{G}})$ be an EC transformation on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, with digraph $\overrightarrow{\mathcal{G}}^{\phi} = \langle \mathcal{V}, \overrightarrow{\mathcal{E}}^{\phi} \rangle$. Let $\mathbf{A}_{\phi}$ be the adjacency matrix associated with $\overrightarrow{\mathcal{G}}^{\phi}$. We have:

$$loss(\phi) = N - \left| \left\{ \{v_1, v_2\} \in \binom{\mathcal{V}}{2} \;\middle|\; \mathbf{A}_{\phi}[v_1, v_2] = 1 \right\} \right|, \tag{122}$$

where $\binom{\mathcal{V}}{2}$ denotes the set of unordered pairs of distinct elements of $\mathcal{V}$.

Proof: Proposition 6

Let $v_1 \in \mathcal{V}$. If $\phi(v_1) \in \mathcal{V}$, then due to injectivity, there is a unique $v_2 \in \mathcal{V}$ such that $\mathbf{A}_{\phi}[v_1, v_2] = 1$. If $\phi(v_1) = \perp$, then $\forall v_2 \in \mathcal{V} : \mathbf{A}_{\phi}[v_1, v_2] = 0$. Therefore, $loss(\phi)$ is N minus the number of vertices that have an image in $\mathcal{V}$.

Not all graphs admit lossless EC transformations. Indeed, we can derive a few sufficient properties as well as necessary ones for an EC transformation to be lossless:

Proposition 7

Consider a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. In order to have $\mathrm{EC}(\Phi^*_{\mathcal{G}}) \neq \emptyset$, we have the following properties:

1. (Necessary): $\forall v \in \mathcal{V} : |\mathcal{N}(v)| > 0$;
2. (Necessary): No vertex is the unique neighbor for two other vertices;
3. (Sufficient): There exists an Hamiltonian cycle in $\mathcal{G}$;
4. (Sufficient): There exists a perfect matching between all vertices in $\mathcal{V}$, *i.e.*, there is a subset $\mathcal{E}^-$ of $\mathcal{E}$ such that every vertex appears exactly once in the edges of $\mathcal{E}^-$.

Proof: Proposition 7

Let $\phi \in \text{EC}(\Phi_{\mathcal{G}})$ be an EC transformation. Let us consider the properties in the same order as above:

1. Let $v \in \mathcal{V}$. If $|\mathcal{N}(v)| = 0$, then the case $\{v, \phi(v)\} \in \mathcal{E}$ of Definition 68 is never matched, therefore $\phi(v) = \perp$;
2. Let $v_1, v_2, v_3 \in \mathcal{V}$, with $\mathcal{N}(v_1) = \{v_3\}$ and $\mathcal{N}(v_2) = \{v_3\}$. To avoid the case where a vertex has its image equal to $\perp$, we must have $\phi(v_1) = v_3$ and $\phi(v_2) = v_3$. However, this contradicts injectivity of transformations;
3. Let $v_1 \to v_2 \to \cdots \to v_N \to v_1$ be an Hamiltonian cycle in the graph. The transformation that associates with every vertex its successor in the cycle is EC, and lossless;
4. If a perfect matching exists, we can determine $\mathcal{E}^- \subset \mathcal{E}$ with $\left|\mathcal{E}^-\right| = \frac{N}{2}$ such that $\forall v_1 \in \mathcal{V} : \exists v_2 \in \mathcal{V} : \{v_1, v_2\} \in \mathcal{E}^-$. In this case, the transformation that associates with v_1 its neighbor v_2 is EC and lossless.

Among all transformations, we are in particular interested in translations. Since their definition is not straightforward, we introduce the following properties:

Definition 69: Weakly neighborhood-preserving (WNP) transformation

We say that a transformation $\phi \in \Phi_{\mathcal{G}}$ on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is ***weakly neighborhood-preserving*** if:

$$\forall \{v_1, v_2\} \in \mathcal{E} : (\{\phi(v_1), \phi(v_2)\} \in \mathcal{E}) \vee (\phi(v_1) = \perp) \vee (\phi(v_2) = \perp) \ . \tag{123}$$

We note $\text{WNP}(\Phi_{\mathcal{G}})$ the set of WNP transformations on $\mathcal{G}$, and $\text{WNP}\left(\Phi^*_{\mathcal{G}}\right)$ the set of lossless WNP transformations on $\mathcal{G}$.

Informally, WNP transformations conserve existing neighborhoods. However, note that two vertices that are not neighbors may be associated with neighboring vertices through a WNP transformation. Transformations that do not create additional neighborhoods are characterized as follows:

Definition 70: Strongly neighborhood-preserving (SNP) transformation

We say that a transformation $\phi \in \Phi_{\mathcal{G}}$ on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is ***strongly neighborhood-preserving*** if:

$$\forall v_1, v_2 \in \mathcal{V} : (\{v_1, v_2\} \in \mathcal{E} \Leftrightarrow \{\phi(v_1), \phi(v_2)\} \in \mathcal{E}) \vee (\phi(v_1) = \perp) \vee (\phi(v_2) = \perp) \ . \tag{124}$$

We denote the set of SNP transformations on $\mathcal{G}$ by $\text{SNP}(\Phi_{\mathcal{G}})$, and the set of lossless SNP transformations on $\mathcal{G}$ by $\text{SNP}\left(\Phi^*_{\mathcal{G}}\right)$.

4.2.2 Translations on graphs

Using the definitions of transformations and their properties in Section 4.2.1, we can now define translations on graphs as follows:

Definition 71: Translation on a graph

A ***translation*** ψ on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is an EC and SNP transformation. We call the set of translations on $\mathcal{G}$ $\Psi_{\mathcal{G}}$, and the set of lossless translations on $\mathcal{G}$ $\Psi^*_{\mathcal{G}}$.

Figure 43 depicts two examples of translations on a graph. Again, note that the function $\phi_\perp$ introduced in (118) is a translation for any graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$:

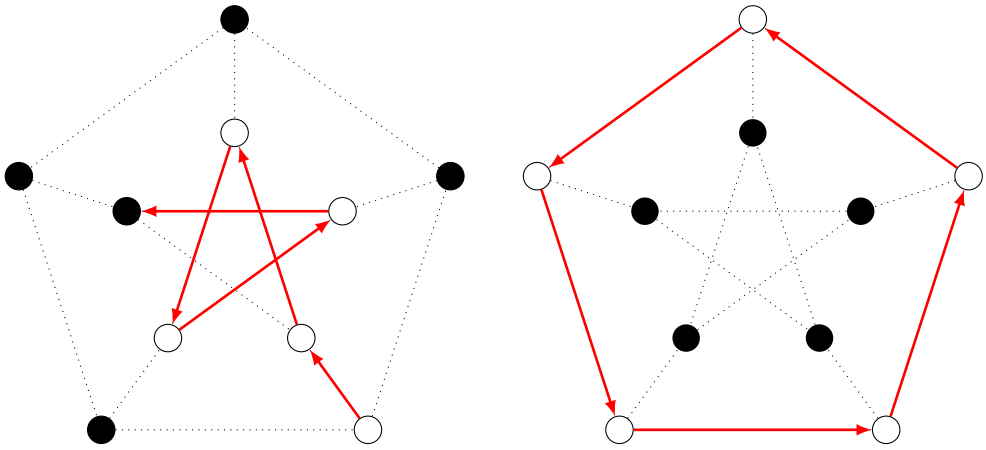


Figure 43 – Examples of transformations that are translations on the Petersen graph introduced in Figure 42.

Additionally, we observe the following property:

Proposition 8

Let $\psi \in \Psi_{\mathcal{G}}$ be a translation on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, with associated digraph $\overrightarrow{\mathcal{G}}^{\psi} = \langle \mathcal{V}, \overrightarrow{\mathcal{E}}^{\psi} \rangle$. Edges of $\overrightarrow{\mathcal{E}}^{\psi}$ can be partitioned into directed cycles, and directed paths that have one vertex for which the image is $\perp$.

Proof: Proposition 8

By injectivity of transformations, any vertex $v_1 \in \mathcal{V}$ has an image by ψ which is either a vertex $v_2 \in \mathcal{V}$ with no other inverse image, or $\perp$. Therefore, every vertex belongs either to a path $v_1 \to v_2 \to \cdots \to \perp$, or a cycle $v_1 \to v_2 \to \cdots \to v_1$. Additionally, the associated digraph restricts the existence of these paths and cycles to paths and cycles that exist in $\mathcal{E}$.

It is also interesting to notice that every translation admits an inverse translation with the same loss:

Proposition 9

Let $\psi \in \Psi_{\mathcal{G}}$ be a translation on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, with associated digraph $\overrightarrow{\mathcal{G}}^{\psi} = \langle \mathcal{V}, \overrightarrow{\mathcal{E}}^{\psi} \rangle$ of adjacency matrix $\mathbf{A}_\psi$. Let us call ψ^{-1} the inverse translation associated with the digraph $\overrightarrow{\mathcal{G}}^{\psi^{-1}} = \langle \mathcal{V}, \overrightarrow{\mathcal{E}}^{\psi^{-1}} \rangle$, with $(v_1, v_2) \in \overrightarrow{\mathcal{E}}^{\psi^{-1}} \Leftrightarrow (v_2, v_1) \in \overrightarrow{\mathcal{E}}^{\psi}$. We have the relation $loss(\psi) = loss(\psi^{-1})$.

Proof: Proposition 9

First, let us notice that $\overrightarrow{\mathcal{E}}^{\psi^{-1}}$ is the exact same set of edges as in $\overrightarrow{\mathcal{E}}^{\psi}$, but with reverse direction. Therefore, since ψ is EC, it is also the case for ψ^{-1}. Additionally, since ψ is SNP, it preserves the existing neighborhoods and does not create additional ones. Therefore, it is also the case for the converse, and ψ^{-1} is thus SNP. From Definition 71, ψ^{-1} is therefore a translation. Finally, from Proposition 6, and noticing that the adjacency matrix associated with $\overrightarrow{\mathcal{G}}^{\psi^{-1}}$ is the transpose of $\mathbf{A}_{\psi}$, we conclude.

Translations can be given an arbitrary well-founded relation $\prec$:

$$\forall \psi_1, \psi_2 \in \Psi_{\mathcal{G}} : (\psi_1 \prec \psi_2) \Leftrightarrow (loss(\psi_1) > loss(\psi_2)) \wedge (\exists v \in \mathcal{V} : \psi_1(v) = \psi_2(v)) \ . \quad (125)$$

Proposition 10

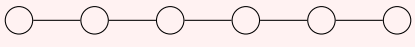
Let $\psi_1, \psi_2, \psi_3 \in \Psi_{\mathcal{G}}$. The relation $\prec$ has the following properties:

1. It is ***irreflexive***, *i.e.*, $\psi_1 \prec \psi_1$ is not true;
2. It is ***antisymmetric***, *i.e.*, it is not possible to have both $\psi_1 \prec \psi_2$ and $\psi_2 \prec \psi_1$;
3. It is ***intransitive***, *i.e.*, it is not true that $((\psi_1 \prec \psi_2) \wedge (\psi_2 \prec \psi_3)) \Rightarrow (\psi_1 \prec \psi_3)$.

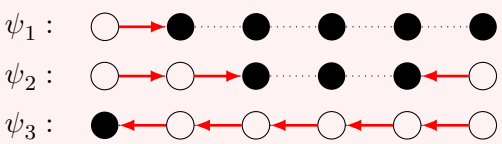
Proof: Proposition 10

Let us consider the three properties separately:

1. By definition, ψ_1 is comparable to itself, since there exists at least one edge in common. Comparison is then made using $<$, which is an irreflexive order on $\mathbb{Z}$;
2. In the case where $\nexists v \in \mathcal{V} : \psi_1(v) = \psi_2(v)$, then ψ_1 and ψ_2 are not comparable (noted $\psi_1 \sim \psi_2$). In the case where such an edge exists, $<$ is an antisymmetric order on $\mathbb{Z}$;
3. Let us consider the following graph:

   Let ψ_1, ψ_2, ψ_3 be the following translations:

   In this example, we observe that $\psi_1 \prec \psi_2$ and $\psi_2 \prec \psi_3$. However, ψ_1 and ψ_3 have no edge in common, thus $\psi_1 \sim \psi_3$. Still, note that ψ_3^{-1}, the inverse translation of ψ_3, is comparable with ψ_1 and ψ_2. As a consequence, $\prec$ is not an ***antitransitive*** relation.

Using this relation, we define minimal translations:

Definition 72: Minimal translation

A translation $\psi_1 \in \Psi_{\mathcal{G}}$ is ***minimal*** if there is no $\psi_2 \in \Psi_{\mathcal{G}}$ such that $\psi_1 \prec \psi_2$, *i.e.*, if it minimizes the loss.

Indeed, lossless translations $\psi^* \in \Psi^*_{\mathcal{G}}$ are necessarily minimal. Additionally, we define pseudo-minimal translations as follows:

Definition 73: Pseudo-minimal translation

Pseudo-minimal translations are defined inductively. A translation $\psi_1 \in \Psi_{\mathcal{G}}$ is pseudo-minimal if one of the following holds:

1. ψ_1 is minimal;
2. Any translation $\psi_2 \in \Psi_{\mathcal{G}}$ such that $\psi_1 \prec \psi_2$ is not pseudo-minimal.

The following result gives that minimal translations can be found on any graph:

Proposition 11

Any graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ admits at least one minimal translation $\psi \in \Psi_{\mathcal{G}}$. Additionally, ψ^{-1} is minimal.

Proof: Proposition 11

In order to show that a minimal translation exists, we study the following cases:

1. In the case where $|\mathcal{E}| = 0$, the only possible translation is $\phi_\perp$ (118), which is thus minimal;
2. In the more general case, let us consider an edge $\{v_1, v_2\} \in \mathcal{E}$. The function ψ_1 such that $\psi_1(v_1) = v_2$, and $\forall v_3 \neq v_1 : \psi_1(v_3) = \perp$ is obviously a translation. Now, consider a maximal sequence $(\psi_i)_i$ of translations of which first element is ψ_1 and such that $\forall i : \psi_i \prec \psi_{i+1}$. This sequence is necessarily finite, since $loss(\psi_i)$ decreases and $<$ is a well-founded order on $\mathbb{Z}$. By definition, the last element ψ_j of this sequence is a maximal translation.

   Now, let us show that ψ_j^{-1} — the inverse translation as defined in Proposition 9 — is also minimal. From Proposition 9, we have $loss(\psi_j) = loss(\psi_j^{-1})$. Now, let us imagine that there exists ψ_k such that $\psi_j^{-1} \prec \psi_k$. Using the same reasoning as above, and noticing that ψ_j^{-1} and ψ_k share at least one edge, we obtain that $\psi_j \prec \psi_k^{-1}$. Since ψ_j is minimal, we reach a contradiction. As a consequence, both ψ_j and ψ_j^{-1} are minimal translations. It is interesting to notice that a special case occurs when ψ_j is a perfect matching between all vertices in $\mathcal{V}$. In this situation, we have $\psi_j = \psi_j^{-1}$.

To sum up the various sets we introduced in this section, Figure 44 presents the corresponding Venn diagram:

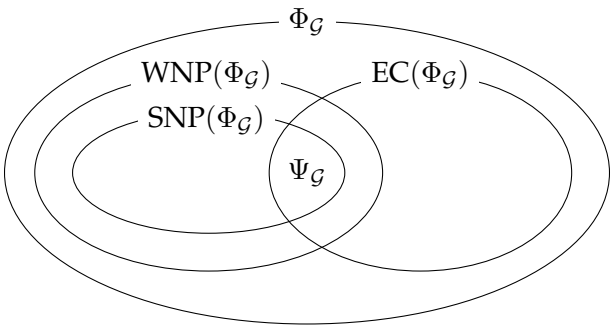


Figure 44 – Venn diagram summarizing the various types of transformations introduced in this section.

4.2.3 Isometries on graphs

Translations on Euclidean spaces are isometries. When it comes to graphs, we also want the translations to be distance-preserving functions. However, since in the general case there is no Euclidean space associated with the graph, we consider here the geodesic distance d_{geo}:

Definition 74: Isometry on a graph

A transformation $\phi \in \Phi_{\mathcal{G}}$ on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is an ***isometry*** if:

$$\forall v_1, v_2 \in \mathcal{V} : \big(d_{\text{geo}}(v_1, v_2) = d_{\text{geo}}(\phi(v_1), \phi(v_2))\big) \vee (\phi(v_1) = \perp) \vee (\phi(v_2) = \perp) \ . \tag{126}$$

We denote the set of isometries on $\mathcal{G}$ by ISO($\Phi_{\mathcal{G}}$), and the set of lossless isometries on $\mathcal{G}$ by ISO$\big(\Phi^*_{\mathcal{G}}\big)$.

Examples of isometries are the translations presented in Figure 43.

In the previous section, some of the sets that were introduced are isometries. In particular, we have the following property:

Proposition 12

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a graph. SNP$\big(\Phi^*_{\mathcal{G}}\big) \subset$ ISO$\big(\Phi^*_{\mathcal{G}}\big)$.

Proof: Proposition 12

Let $v_1, v_2 \in \mathcal{V}$. For a lossless SNP transformation $\phi^* \in$ SNP$\big(\Phi^*_{\mathcal{G}}\big)$, we distinguish the following cases:

1. There is no path between vertices v_1 and v_2, *i.e.*, $d_{\text{geo}}(v_1, v_2)$ is infinite. This corresponds to the case where they belong to different connected components. By contradiction, let $d_{\text{geo}}(\phi^*(v_1), \phi^*(v_2))$ be finite. This implies that there exists a path $\phi^*(v_1) \to \phi^*(v_{i_1}) \to \phi^*(v_{i_2}) \to \cdots \to \phi^*(v_{i_k}) \to \phi^*(v_2)$ in the graph. Since ϕ^* is SNP, we have that $v_1 \to v_{i_1} \to v_{i_2} \to \cdots \to v_{i_k} \to v_2$ is also a path in the graph, therefore we reach a contradiction. As a conse-

quence, $d_{\text{geo}}(v_1, v_2)$ and $d_{\text{geo}}(\phi^*(v_1), \phi^*(v_2))$ are both infinite;

2. A shortest path $v_1 \to v_{i_1} \to v_{i_2} \to \cdots \to v_{i_k} \to v_2$ exists. Since ϕ^* is lossless, $\phi^*(v_1) \to \phi^*(v_{i_1}) \to \phi^*(v_{i_2}) \to \cdots \to \phi^*(v_{i_k}) \to \phi^*(v_2)$ is also a path (no intermediary vertex has its image equal to $\perp$). Additionally, ϕ^* being SNP, it does not create nor remove neighborhoods, so $\phi^*(v_1) \to \phi^*(v_{i_1}) \to \phi^*(v_{i_2}) \to \cdots \to \phi^*(v_{i_k}) \to \phi^*(v_2)$ is also a shortest path from $\phi^*(v_1)$ to $\phi^*(v_2)$. Therefore, we have $d_{\text{geo}}(v_1, v_2) = d_{\text{geo}}(\phi^*(v_1), \phi^*(v_2))$.

The previous result simply implies the following:

Corrolary 1

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a graph. $\Psi^*_{\mathcal{G}} \subset \text{ISO}(\Phi^*_{\mathcal{G}})$.

Proof: Corrolary 1

$\Psi^*_{\mathcal{G}} \subset \text{SNP}(\Phi^*_{\mathcal{G}}) \subset \text{ISO}(\Phi^*_{\mathcal{G}})$.

Note that Proposition 12 holds for lossless SNP transformations only. In the more general case of SNP transformations $\phi \in \text{SNP}(\Phi_{\mathcal{G}})$, having a vertex that has its image equal to $\perp$ may cause $d_{\text{geo}}(\phi(v_1), \phi(v_2))$ and $d_{\text{geo}}(v_1, v_2)$ to be different.

As an example, Figure 45 depicts an SNP transformation $\phi \in \text{SNP}(\Phi_{\mathcal{G}})$ for which $d_{\text{geo}}(①, ②) < d_{\text{geo}}(\phi(①), \phi(②))$. When considering the inverse transformation ϕ^{-1}, we have $d_{\text{geo}}(①, ②) > d_{\text{geo}}(\phi^{-1}(①), \phi^{-1}(②))$:

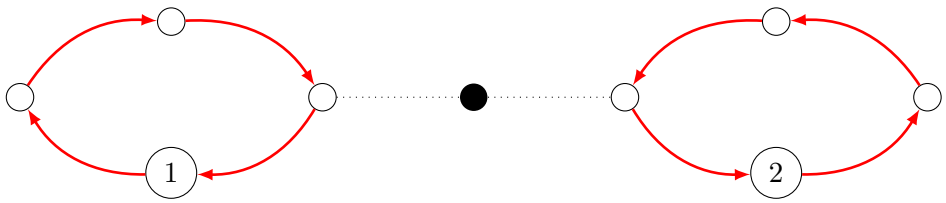


Figure 45 – Example of a SNP transformation $\phi \in \text{SNP}(\Phi_{\mathcal{G}})$ with a non-zero loss that is not an isometry. In this example, $d_{\text{geo}}(①, ②) = 4$, $d_{\text{geo}}(\phi(①), \phi(②)) = 6$ and $d_{\text{geo}}(\phi^{-1}(①), \phi^{-1}(②)) = 2$.

Still, it is interesting to note that some transformations with a non-zero loss are isometries. Examples of such are depicted in Figure 43.

4.3 Results on translations on graphs

In this section, we are interested in identifying translations on arbitrary graphs. After introducing some generic results, we study the case of the torus graph, on which defining a notion of translation is intuitive. This gives us intuition on how to find these translations without considering the underlying Euclidean space. Then, we show that we are able to find the translations on a grid graph, and extend our results to families of graphs that are more generic.

4.3.1 Results on generic graphs

As stated before, the function $\phi_\perp$ introduced in (118) is a translation for any graph. However, it is not very interesting, since it destroys all signal information when translating it. Therefore, we need to identify more complex translations that keep most of the signal entries. As a consequence, we are particularly interested in minimal translations.

Before trying to identify translations, we provide bounds on the number of translations:

Proposition 13

A graph with order N cannot admit more than:

$$\sum_{k=0}^{N} \frac{1}{(N-k)!} \sum_{j=0}^{k} (-1)^j \binom{k}{j} (N-j)! \tag{127}$$

translations. This is reached for the complete graph $\mathcal{G}_c = \langle \mathcal{V}_c, \mathcal{E}_c \rangle$.

Proof: Proposition 13

Every EC transformation on $\mathcal{G}_c$ is necessary SNP, since all vertices are pairwise linked and therefore share the same neighborhood. However, not every transformation on such graph is EC, since transformations can map vertices to themselves, and we consider simple graphs only. Therefore, for a fixed loss $N-k$ ($k \leq N$), the set of translations of loss $N-k$ is exactly the set of injective functions that have no fixed points of k elements to N. The cardinal of such a set is given by the solution of the (N,k)-*matching problem* in [HSW83] as follows:

$$\frac{1}{(N-k)!} \sum_{j=0}^{k} (-1)^j \binom{k}{j} (N-j)! \,. \tag{128}$$

By summing for every possible value of k, corresponding to the number of vertices that have an image different to $\perp$, we obtain the number of translations. Then, note that any graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ of order N has its edges $\mathcal{E} \subset \mathcal{E}_c$. As a consequence, since any EC transformation is a translation on $\mathcal{G}_c$, it follows that any translation on $\mathcal{G}$ is also a translation on $\mathcal{G}_c$. Therefore, a graph of order N cannot admit more translations than the complete graph.

This characterization of the number of translations gives us the following result on the number of minimal translations:

Proposition 14

A graph of order N cannot admit more than:

$$N! \sum_{j=0}^{N} \frac{(-1)^j}{j!} \tag{129}$$

minimal translations. This is reached for the complete graph $\mathcal{G}_c = \langle \mathcal{V}_c, \mathcal{E}_c \rangle$.

Proof: Proposition 14

Minimal translations on $\mathcal{G}_c$ are necessarily lossless, since any translation shares at least a diedge with an Hamiltonian cycle on this graph, which is lossless. By particularizing (128) for $k = N$, we obtain the number of lossless translations on $\mathcal{G}_c$, which is exactly the number of derangements of a set of N elements, *i.e.*, the number of permutations of N elements with no fixed points [Mon13].

Using the same reasoning as in the proof of Proposition 13, any minimal translation on a graph $\mathcal{G}$ is included in a minimal translation on $\mathcal{G}_c$. Therefore, a graph of order N cannot admit more minimal translations than the complete graph.

The number of translations on graphs is therefore exponential in the general case. Additionally, we prove that identifying translations is a complex problem:

Proposition 15

The problem of deciding, for an input graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ and two subsets $\mathcal{V}_1$ and $\mathcal{V}_2$ of $\mathcal{V}$, if there is a translation for which the image is exactly $\mathcal{V}_2$ and with inverse images only in $\mathcal{V}_1$ is NP-complete.

Proof: Proposition 4.3.1

We first prove that the problem is NP, and then that it is NP-hard.

All possible transformations with inverse image set $\mathcal{V}_1$ and image set $\mathcal{V}_2$ can be generated non-deterministically by induction:

1. $\phi_\perp \in \Phi_\mathcal{G}$;
2. If we have a transformation $\phi_1 \in \Phi_\mathcal{G}$, and two vertices $v_1 \in \mathcal{V}_1$, $v_2 \in \mathcal{V}_2$ such that $\phi_1(v_1) = \perp$ and $\nexists v_3 \in \mathcal{V}_1 : \phi_1(v_3) = v_2$, then define $\phi_2 \in \Phi_\mathcal{G}$ as follows:
   - $\phi_2(v_1) = v_2$;
   - $\forall v_3 \in \mathcal{V}_1, v_3 \neq v_1 : \phi_2(v_3) = \phi_1(v_3)$.

Furthermore, determining whether any such transformation is a translation or not can be done by checking EC and SNP constraints, which can be done in polynomial time. So the problem is NP.

Then, we prove the problem is NP-hard by reduction from the subgraph isomorphism problem. Consider two graphs $\mathcal{G}_1 = \langle \mathcal{V}_1, \mathcal{E}_1 \rangle$ and $\mathcal{G}_2 = \langle \mathcal{V}_2, \mathcal{E}_2 \rangle$. Without loss of generality, we consider that $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$. The more general case where $\mathcal{V}_1 \cap \mathcal{V}_2 \neq \emptyset$ can be included by duplication of the vertices in the intersection.

From these graphs, we build the graph $\mathcal{G}_3 = \langle \mathcal{V}_1 \cup \mathcal{V}_2, \mathcal{E}_3 \rangle$, where $\mathcal{E}_3 = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \{\{v_1, v_2\}, v_1 \in \mathcal{V}_1, v_2 \in \mathcal{V}_2\}$. Note that this construction is at most quadratic in the order of $\mathcal{G}$. Then, we show that answering our problem on $\mathcal{G}_3$ solves the problem of subgraph isomorphism between $\mathcal{G}_1$ and $\mathcal{G}_2$.

To this end, consider the following properties, that we prove to be equivalent:

1. There is a translation of which image set is $\mathcal{V}_2$ and inverse images are in $\mathcal{V}_1$;
2. There is a subgraph of $\mathcal{G}_1$ isomorph to $\mathcal{G}_2$.

We prove this in two steps. First, consider there exists such a translation. Then, since it is SNP, the subgraph corresponding to the inverse images of vertices in $\mathcal{V}_2$ is isomorph to $\mathcal{G}_2$.

Conversely, consider that there exists an isomorphism, then the transformation that associates each vertex in $\mathcal{V}_2$ with its corresponding vertex in $\mathcal{V}_1$ is a translation. indeed, it is EC because of the complete bipartite subgraph connecting vertices in $\mathcal{V}_2$ to vertices in $\mathcal{V}_1$, and it is SNP as a particularization of the isomorphism property.

As a consequence, the problem of deciding, for an input graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ and two subsets $\mathcal{V}_1$ and $\mathcal{V}_2$ of $\mathcal{V}$, if there is a translation for which the image is exactly $\mathcal{V}_2$ and with inverse images only in $\mathcal{V}_1$ is at least as difficult as the subgraph isomorphism problem. Since it is also NP, it is NP-complete.

These results tell us that finding the translations on a graph is a hard problem. Therefore, one may need to establish approximate methods to identify interesting translations on a given graph. In the following subsections, we focus on the particular case of highly regular graphs, namely the torus graph and the grid graph. We develop generic results on these graphs and extend them to any class of graphs.

4.3.2 Results on the torus graph

Let us first consider the case of the torus graph $\mathcal{G}_t = \langle \mathcal{V}_t, \mathcal{E}_t \rangle$ yielded by a dimensions vector $\mathbf{d} \in \mathbb{N}^{*D}$. Such graphs are highly regular, and are often used to model classical domains, such as the periodical time with a 1-dimensional torus graph, or the pixels of a periodical image with a 2-dimensional torus graph. Additionally, what makes these graphs interesting is the fact that they are constructed using an Euclidean space (see Definition 25), associating each vertex with coordinates.

In this section, we aim to find a relation between translations defined on an Euclidean space, and those defined on the graph, with no reference to the underlying metrics. To do so, let us first formalize the notion of Euclidean translation on $\mathcal{G}_t$:

Definition 75: Euclidean translation on the torus graph

An ***Euclidean translation*** ψ_{Et} on the torus graph is such that:

$$\exists \boldsymbol{\gamma} \in \mathbb{N}^D : \forall \mathbf{v} \in \mathcal{V}_t : \psi_{Et}(\mathbf{v}) = \mathbf{v} +_{\mathbf{d}} \boldsymbol{\gamma} \,, \tag{130}$$

where $+_{\mathbf{d}}$ means $\forall i \in [\![1, D]\!] : \mathbf{v}[i] +_{\mathbf{d}[i]} \boldsymbol{\gamma}[i]$.

By construction of the torus graph, we have that $\forall \mathbf{v} \in \mathcal{V}_t, \forall i \in [\![1, D]\!] : \{\mathbf{v}, \mathbf{v} +_{\mathbf{d}} \boldsymbol{\delta}_i\} \in \mathcal{E}_t$.

Remark 7

Note that this is also true for the negative Dirac vectors, defined to contain a single non-null entry i being -1. As a consequence, the following results also apply using such vectors.

Using the coordinates associated with the vertices of the torus graph, we can show that lossless translations on this graph consist in moving all vertices to the same direction:

Lemma 1: Contamination lemma on the torus graph

Let $\psi \in \Psi^*_{\mathcal{G}_t}$ be a lossless translation on the torus graph, with $\forall i \in [\![1, D]\!] : \mathbf{d}[i] \geq 5$. Let $\mathbf{v}_1 \in \mathcal{V}_t$. Let us consider the Dirac vector $\boldsymbol{\delta}_j = \psi(\mathbf{v}_1) -_{\mathbf{d}} \mathbf{v}_1$. Then, $\forall \mathbf{v}_2 \in \mathcal{V}_t : \psi(\mathbf{v}_2) = \mathbf{v}_2 +_{\mathbf{d}} \boldsymbol{\delta}_j$.

Proof: Lemma 1

We proceed in two steps:

1. First, let us show that $\psi(\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j) = \mathbf{v}_1$. By construction of the torus graph in Definition 25, we have that $\{\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j, \mathbf{v}_1\} \in \mathcal{E}_t$. Since ψ is EC, we must have $\psi(\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j) \in \mathcal{N}(\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j)$. Also, since ψ is SNP, we must have $\psi(\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j) \in \mathcal{N}(\psi(\mathbf{v}_1))$. As a consequence, $\psi(\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j) \in \mathcal{N}(\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j) \cap \mathcal{N}(\psi(\mathbf{v}_1))$.
   The neighborhood of $\psi(\mathbf{v}_1)$ is:

$$\begin{aligned}\mathcal{N}(\psi(\mathbf{v}_1)) = \mathcal{N}(\mathbf{v}_1 +_{\mathbf{d}} \boldsymbol{\delta}_j) = \{&\mathbf{v}_1 +_{\mathbf{d}} \boldsymbol{\delta}_j +_{\mathbf{d}} \boldsymbol{\delta}_1, \mathbf{v}_1 +_{\mathbf{d}} \boldsymbol{\delta}_j -_{\mathbf{d}} \boldsymbol{\delta}_1,\\ &\mathbf{v}_1 +_{\mathbf{d}} \boldsymbol{\delta}_j +_{\mathbf{d}} \boldsymbol{\delta}_2, \mathbf{v}_1 +_{\mathbf{d}} \boldsymbol{\delta}_j -_{\mathbf{d}} \boldsymbol{\delta}_2,\\ &\dots,\\ &\mathbf{v}_1 +_{\mathbf{d}} 2\boldsymbol{\delta}_j, \mathbf{v}_1,\\ &\dots,\\ &\mathbf{v}_1 +_{\mathbf{d}} \boldsymbol{\delta}_j +_{\mathbf{d}} \boldsymbol{\delta}_D, \mathbf{v}_1 +_{\mathbf{d}} \boldsymbol{\delta}_j -_{\mathbf{d}} \boldsymbol{\delta}_D\}\ .\end{aligned} \tag{131}$$

   Similarly, the neighborhood of $\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j$ is:

$$\begin{aligned}\mathcal{N}(\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j) = \{&\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j +_{\mathbf{d}} \boldsymbol{\delta}_1, \mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j -_{\mathbf{d}} \boldsymbol{\delta}_1,\\ &\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j +_{\mathbf{d}} \boldsymbol{\delta}_2, \mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j -_{\mathbf{d}} \boldsymbol{\delta}_2,\\ &\dots,\\ &\mathbf{v}_1, \mathbf{v}_1 -_{\mathbf{d}} 2\boldsymbol{\delta}_j,\\ &\dots,\\ &\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j +_{\mathbf{d}} \boldsymbol{\delta}_D, \mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j -_{\mathbf{d}} \boldsymbol{\delta}_D\}\ .\end{aligned} \tag{132}$$

   Since $\forall i \in [\![1, D]\!] : \mathbf{d}[i] \geq 3$, we have $\mathbf{v}_1 +_{\mathbf{d}} \boldsymbol{\delta}_j +_{\mathbf{d}} \boldsymbol{\delta}_k \neq \mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j +_{\mathbf{d}} \boldsymbol{\delta}_k$ ($j \neq k$). Therefore, vertices with coordinates that differ by an entry in dimension $k \neq j$ cannot belong to the intersection by construction of the torus graph. Similarly, because $\forall i \in [\![1, D]\!] : \mathbf{d}[i] \geq 5$, we obtain that $\mathbf{v}_1 +_{\mathbf{d}} 2\boldsymbol{\delta}_j$ cannot be the same vertex as $\mathbf{v}_1 -_{\mathbf{d}} 2\boldsymbol{\delta}_j$, since they differ by $\alpha\boldsymbol{\delta}_j$ ($\alpha > 1$). As a consequence, $\mathcal{N}(\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j) \cap \mathcal{N}(\psi(\mathbf{v}_1)) = \{\mathbf{v}_1\}$, and thus $\psi(\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j) = \mathbf{v}_1$;
2. Now, let us consider a vertex $\mathbf{v}_2 \in \mathcal{N}(v_1) \backslash \{v_1 -_{\mathbf{d}} \boldsymbol{\delta}_j, v_1 +_{\mathbf{d}} \boldsymbol{\delta}_j\}$. Let us show that $\psi(\mathbf{v}_2) = \mathbf{v}_2 +_{\mathbf{d}} \boldsymbol{\delta}_j$. As in the step 1., comparing the neighborhoods of $\mathbf{v}_2$ and $\psi(\mathbf{v}_1)$ gives us $\mathcal{N}(\mathbf{v}_2) \cap \mathcal{N}(\psi(\mathbf{v}_1)) = \{\mathbf{v}_1, \mathbf{v}_2 +_{\mathbf{d}} \boldsymbol{\delta}_j\}$. However, step 1. gives us that $\mathbf{v}_1$ is necessarily the image of $\mathbf{v}_1 -_{\mathbf{d}} \boldsymbol{\delta}_j$. Since ψ is injective, it follows that $\psi(\mathbf{v}_2) = \mathbf{v}_2 +_{\mathbf{d}} \boldsymbol{\delta}_j$.

By induction, we conclude that for every vertex $\mathbf{v}_2 \in \mathcal{V}_t : \psi(\mathbf{v}_2) = \mathbf{v}_2 +_{\mathbf{d}} \boldsymbol{\delta}_j$.

The constraint of having all dimensions in $\mathbf{d}$ being larger than 5 allows any lossless

translation to verify $\forall \mathbf{v} \in \mathcal{V}_t : \psi(\mathbf{v}) = \mathbf{v} +_{\mathbf{d}} \boldsymbol{\delta}_j$. For smaller graphs, lossless translations can be found for which this property is not true. As an example, Figure 46 depicts lossless translations on a torus of dimensions $\mathbf{d} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$ that are not translations by a Dirac vector. Still, note that any translation ψ such that $\forall \mathbf{v} \in \mathcal{V}_t : \psi(\mathbf{v}) = \mathbf{v} +_{\mathbf{d}} \boldsymbol{\delta}_j$ is lossless even for smaller grid graphs:

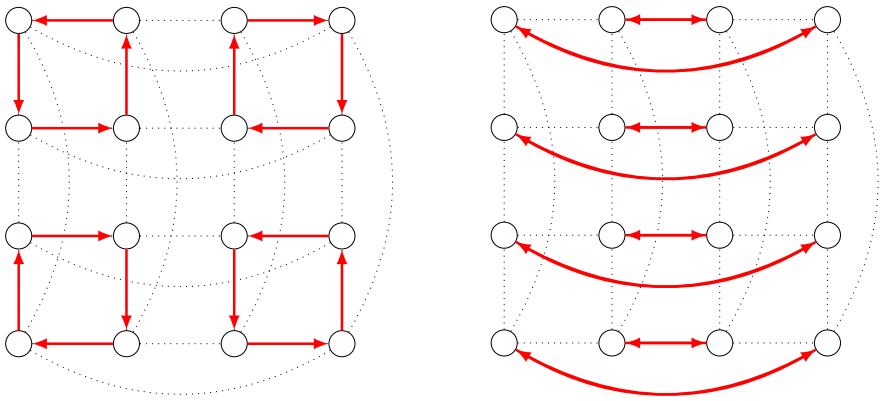


Figure 46 – Examples of lossless translations on a torus of dimensions $\mathbf{d} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$ that are not translations by a Dirac vector.

Note that a direct consequence of Lemma 1 is that there are as many lossless translations as there are neighbors for a given vertex. By composing the lossless translations on the torus graph, we obtain more complex functions, that induce the following monoid:

Definition 76: Monoid induced by $\Psi^*_{\mathcal{G}_t}$

We call ***monoid induced by*** $\Psi^*_{\mathcal{G}_t}$ the minimum monoid containing $\Psi^*_{\mathcal{G}_t}$ with the composition of functions as inner law.

Proposition 16

For torus graphs with $\forall i \in [\![1, D]\!] : \mathbf{d}[i] \geq 5$, the monoid induced by $\Psi^*_{\mathcal{G}_t}$ is exactly the set of Euclidean translations on the torus graph.

Proof: Proposition 16

A direct consequence of Lemma 1 is that lossless translations $\psi \in \Psi^*_{\mathcal{G}_t}$ on the torus graph can be obtained by choosing a Dirac vector for a dimension $i \in [\![1, D]\!]$ and applying the contamination. Therefore, $\forall i \in [\![1, D]\!] : \exists! \psi \in \Psi^*_{\mathcal{G}_t} : \forall \mathbf{v} \in \mathcal{V}_t : \psi(\mathbf{v}) = \mathbf{v} +_{\mathbf{d}} \boldsymbol{\delta}_i$. We obtain that $\boldsymbol{\gamma}$ in Definition 75 is a linear combination of vectors in $\{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \ldots, \boldsymbol{\delta}_D\}$. As a consequence, any Euclidean translation on the torus graph can be written as a composition of lossless translations on the torus graph, which are elements of the monoid induced by $\Psi^*_{\mathcal{G}_t}$.

4.3.3 Results on the grid graph

Let us now proceed with grid graphs $\mathcal{G}_g = \langle \mathcal{V}_g, \mathcal{E}_g \rangle$ yielded by a dimensions vector $\mathbf{d} \in {\mathbb{N}^*}^D$. We can adapt the definition of Euclidean translation on the torus graph in Definition 75 to grid graphs as follows:

Definition 77: Euclidean translation on the grid graph

An ***Euclidean translation*** ψ_{Eg} on the grid graph $\mathcal{G}_g = \langle \mathcal{V}_g, \mathcal{E}_g \rangle$ of dimensions $\mathbf{d} \in {\mathbb{N}^*}^D$ is such that:

$$\exists \boldsymbol{\gamma} \in \mathbb{N}^D : \forall \mathbf{v} \in \mathcal{V}_g : \psi_{Eg}(\mathbf{v}) = \begin{cases} \mathbf{v} + \boldsymbol{\gamma} & \text{if } \mathbf{v} + \boldsymbol{\gamma} \in \mathcal{V}_g \\ \perp & \text{otherwise} \end{cases} . \tag{133}$$

Restricting the translation to Dirac vertices, we have the following loss:

Proposition 17

Let us consider the translation $\psi \in \Psi_{\mathcal{G}_g}$ such that:

$$\forall \mathbf{v} \in \mathcal{V}_g : \psi(\mathbf{v}) = \begin{cases} \mathbf{v} + \boldsymbol{\delta}_i & \text{if } \mathbf{v} + \boldsymbol{\delta}_i \in \mathcal{V}_g \\ \perp & \text{otherwise} \end{cases} , \tag{134}$$

for $\boldsymbol{\delta}_i$ the Dirac vector for dimension i. We have:

$$loss(\psi) = \prod_{\substack{j \in [\![1, D]\!] \\ j \neq i}} \mathbf{d}[j] . \tag{135}$$

Proof: Proposition 17

By construction of the grid graph, two vertices are neighbors if their coordinates differ by 1 or -1 along a single dimension. In particular, it is true for dimension i. Therefore, any vertex $\mathbf{v}$ such that $\mathbf{v} + \boldsymbol{\delta}_i \notin \mathcal{V}_g$ is such that $\mathbf{v}[i] = \mathbf{d}[i]$. The product of dimensions that are different from i gives us the number of such vertices, hence the loss of ψ.

Remark 8

As for torus graphs, note that all the results in this section also apply when considering negative Dirac vectors containing a single non-null entry i being -1.

As for torus graphs, we can introduce the monoid induced by the translations on the grid graph as follows:

Definition 78: Monoid induced by $\Psi_{\mathcal{G}_g}$

We call ***monoid induced by*** $\Psi_{\mathcal{G}_g}$ the minimum monoid containing $\Psi_{\mathcal{G}_g}$ with the composition of functions as inner law.

Proposition 18

The monoid induced by $\Psi_{\mathcal{G}_g}$ includes the set of Euclidean translations on the grid graph.

Proof: Proposition 18

Translations by Dirac vectors introduced in Proposition 17 exist for every dimension i. It follows that $\boldsymbol{\gamma}$ in Definition 4.3.3 is a linear combination of vectors in $\{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \dots, \boldsymbol{\delta}_D\}$. As a consequence, any Euclidean translation on the grid graph can be written as a composition of translations on the grid graph, which are elements of the monoid induced by $\Psi_{\mathcal{G}_g}$.

However, contrary to the case of torus graphs in Proposition 16, translations introduced in Proposition 17 are only a subset of $\Psi_{\mathcal{G}_g}$. Therefore, Euclidean translations are included in the monoid induced by $\Psi_{\mathcal{G}_g}$, but the converse is not true.

As an counterexample, for a Dirac vector $\boldsymbol{\delta}_1$ and a grid graph such that $\forall i \in [\![1, D]\!] : \mathbf{d}[i] \geq 3$, the translation $\psi \in \Psi_{\mathcal{G}_g}$ such that:

$$\forall \mathbf{v} \in \mathcal{V}_g : \psi(\mathbf{v}) = \begin{cases} \mathbf{v} + \boldsymbol{\delta}_1 & \text{if } (\mathbf{v} + \boldsymbol{\delta}_1 \in \mathcal{V}_g) \wedge \left(\mathbf{v} \neq \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}\right) \\ \bot & \text{otherwise} \end{cases} \tag{136}$$

is not an Euclidean translation on the grid graph.

Now, we are interested in showing that Euclidean translations by $\boldsymbol{\delta}_i$ (or $-\boldsymbol{\delta}_i$) are pseudo-minimal on the grid graph. We restrict our study to a subclass of grid graphs such that each dimension is large compared to the following ones, *i.e.*:

$$\mathbf{d}[D] \geq 3 \wedge \forall i \in [\![1, D-1]\!] : \mathbf{d}[i] \geq 2 + 2 \prod_{j=i+1}^{D} \mathbf{d}[j] \ . \tag{137}$$

This hypothesis is necessary for the subsequent proofs. However, we conjecture the following result:

Conjecture 1

The forthcoming results apply for grid graphs such that $\forall i \in [\![1, D]\!] : \mathbf{d}[i] \geq 6$.

To ease exposition of the following results, we introduce the notion of *slice of a grid graph* as follows:

Definition 79: Grid graph slice

We call ***slice of a grid graph***, noted $\mathcal{V}_g^{i,j}$ the subset of vertices $\mathcal{V}_g$ such that have their i coordinate equal to j, *i.e.*:

$$\mathcal{V}_g^{i,j} = \{\mathbf{v} \in \mathcal{V}_g \mid \mathbf{v}[i] = j\} \ . \tag{138}$$

This notion of graph slices, along with the upper bound on the loss, allows us to show that some particular vertices necessarily have an image by a minimal translation:

Lemma 2

Let $\mathcal{G}_g$ be a grid graph respecting assumption (137). If $\psi \in \Psi_{\mathcal{G}_g}$ is a minimal translation, then:

$$\exists i : \forall \mathbf{v} \in \mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1} : \psi(\mathbf{v}) \neq \perp . \tag{139}$$

Proof: Lemma 2

By Proposition 17, we have an upper bound on the loss of minimal translations when translating vertices along a single dimension. When considering dimension 1, any minimal translation has therefore at most $\prod_{j \in [\![2, D]\!]} \mathbf{d}[j]$ vertices that have their image through ψ being $\perp$. From assumption (137), we have that $\mathbf{d}[1] \geq 2 + 2\prod_{j \in [\![2, D]\!]} \mathbf{d}[j]$. Since $\mathbf{d}[1] - 2\prod_{j \in [\![2, D]\!]} \mathbf{d}[j] + 1 > 1$, there cannot be a strict alternation of vertices with image in $\mathcal{V}_g$, and vertices with image equal to $\perp$, as it would violate the upper bound on the loss. Therefore, there exist two slices $\mathcal{V}_g^{1,i}$ and $\mathcal{V}_g^{1,i+1}$ that contain no vertex $\mathbf{v}$ such that $\psi(\mathbf{v}) = \perp$.

Additionally, we can characterize some of the vertices that have an image different from $\perp$ as follows:

Lemma 3

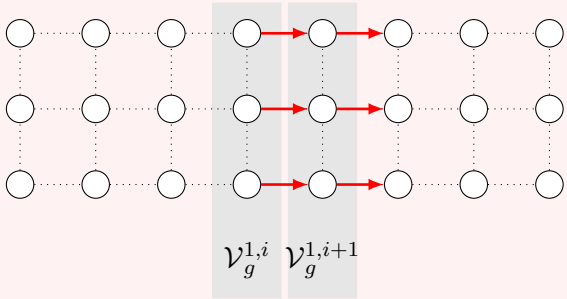
Let $\mathcal{G}_g$ be a grid graph respecting assumption (137), and let $\psi \in \Psi_{\mathcal{G}_g}$ be a minimal translation. If $\mathcal{V}_g^{1,i}$ and $\mathcal{V}_g^{1,i+1}$ are two slices containing no vertex which image by ψ is $\perp$, then:

$$\psi(\mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1}) \not\subset \mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1} . \tag{140}$$

Proof: Lemma 3

Let us consider a vertex $\mathbf{v}_1 \in \mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1}$. Proposition 8 tells us that there are two cases to consider:

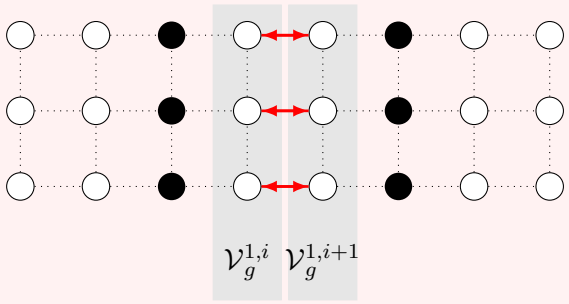
1. $\exists n : \psi^n(\mathbf{v}_1) = \perp$. Since no vertex in $\mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1}$ has its image being $\perp$, the sequence $(\psi^n(\mathbf{v}_1))_n$ necessarily contains a vertex $\mathbf{v}_2 \notin \mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1}$;
2. $\exists n : \psi^n(\mathbf{v}_1) = \mathbf{v}_1$. In this case, we distinguish the following situations, illustrated on a $\begin{bmatrix} 8 \\ 3 \end{bmatrix}$ grid graph:
   (a) Every vertex from $\mathcal{V}_g^{1,i}$ is sent to its neighbor in $\mathcal{V}_g^{1,i+1}$, and every vertex from $\mathcal{V}_g^{1,i+1}$ is sent to its neighbor in $\mathcal{V}_g^{1,i+2}$.



   In this situation, there cannot exist a cycle such that $\psi^n(\mathbf{v}_1) = \mathbf{v}_1$ due

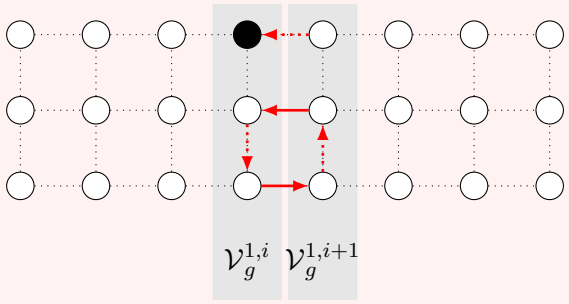
to injectivity of ψ. Note that this situation also applies in the case where every vertex from $\mathcal{V}_g^{1,i}$ is sent to its neighbor in $\mathcal{V}_g^{1,i-1}$, and every vertex from $\mathcal{V}_g^{1,i+1}$ is sent to its neighbor in $\mathcal{V}_g^{1,i}$;

(b) Every vertex from $\mathcal{V}_g^{1,i}$ is sent to its neighbor in $\mathcal{V}_g^{1,i+1}$, and every vertex from $\mathcal{V}_g^{1,i+1}$ is sent to its neighbor in $\mathcal{V}_g^{1,i}$.



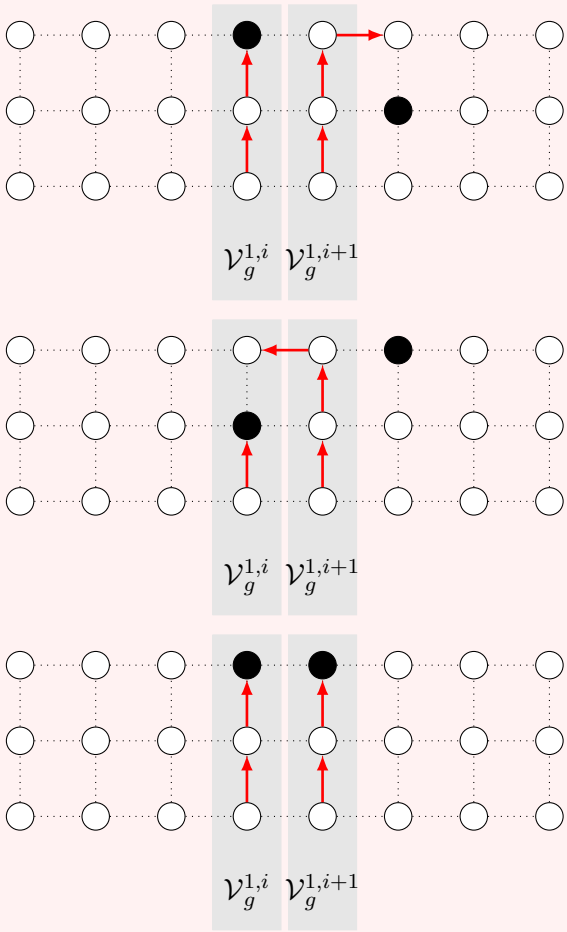
This causes all vertices in $\mathcal{V}_g^{1,i-1} \cup \mathcal{V}_g^{1,i+2}$ to be sent to $\perp$, leading to a loss twice higher than the upper bound for minimal translations given in Proposition 17. Therefore we reach a contradiction;

(c) There exists a vertex $\mathbf{v}_2 \in \mathcal{V}_g^{1,i}$ such that $\psi(v_2) \in \mathcal{V}_g^{1,i+1}$, and a vertex There exists a vertex $\mathbf{v}_3 \in \mathcal{V}_g^{1,i+1} \cap \mathcal{N}(\psi(v_2))$ such that $\psi(v_3) \in \mathcal{V}_g^{1,i}$. Necessarily, $\psi^2(\mathbf{v}_2) = \mathbf{v}_3$ and $\psi^2(\mathbf{v}_3) = \mathbf{v}_2$, causing apparition of a cycle of 4 vertices.



Then, since all dimensions are larger than 3, at least one of the vertices from this cycle has a neighbor $\mathbf{v}_4 \in \mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1}$ for which neighborhood cannot be preserved. As a consequence, there exists a vertex in $\mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1}$ that has its image equal to $\perp$, and we reach a contradiction. Additionally, note that in the case where the diedges of opposite directions are not adjacent, there is necessarily at least a vertex of which image is $\perp$ between them;

(d) Every vertex from $\mathcal{V}_g^{1,i}$ (resp. $\mathcal{V}_g^{1,i+1}$) is sent to a neighbor in $\mathcal{V}_g^{1,i}$ (resp. $\mathcal{V}_g^{1,i+1}$). If the corresponding diedges are of opposite directions, this eventually leads to a *turn*, in this case situation (c) concludes. If they take the same direction, then due to border effects, at least a vertex in $\mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1}$ has its image equal to $\perp$, leading to a contradiction.



Note that all these situations lead to a contradiction. Therefore, a minimal translation ψ cannot lead to the creation of a cycle such that $\psi^n(\mathbf{v}_1) = \mathbf{v}_1$. As a consequence, only case 1. applies, and $\psi(\mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1}) \not\subset \mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1}$.

The enumeration of cases in the proof of Lemma 3 gives us a characterization of the orientation of the edges between these vertices:

Corrolary 2

Let $\mathcal{G}_g$ be a grid graph respecting assumption (137), and let $\psi \in \Psi_{\mathcal{G}_g}$ be a minimal translation. Let $\mathcal{V}_g^{1,i}$ and $\mathcal{V}_g^{1,i+1}$ be two slices containing no vertex of which image through ψ is $\perp$. Every vertex from $\mathcal{V}_g^{1,i}$ is sent to its neighbor in $\mathcal{V}_g^{1,i+1}$, and every vertex from $\mathcal{V}_g^{1,i+1}$ is sent to its neighbor in $\mathcal{V}_g^{1,i+2}$.

Proof: Corrolary 2

This is a direct consequence of the proof of Lemma 3. Any other case corresponds to the situations described by cases 2.(b), 2.(c) and 2.(d) of the proof of Lemma 3, leading to existence of vertices of which image through ψ is $\perp$ in $\mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1}$.

Using these results, we can show that all vertices located *before* these two adjacent slices do not have their image equal to $\perp$:

Lemma 4

Let $\mathcal{G}_g$ be a grid graph respecting assumption (137), and let $\psi \in \Psi_{\mathcal{G}_g}$ be a minimal translation. Let $\mathcal{V}_g^{1,i}$ and $\mathcal{V}_g^{1,i+1}$ be two slices containing no vertex of which image by ψ is $\perp$:

$$\forall j \in [\![1, i+1]\!] : \forall \mathbf{v} \in \mathcal{V}_g^{1,j} : \psi(\mathbf{v}) \neq \perp \, . \tag{141}$$

Proof: Lemma 4

From the proof of Lemma 3, there cannot exist any cycle including vertices in $\mathcal{V}_g^{1,i} \cup \mathcal{V}_g^{1,i+1}$. As a consequence, for every vertex $\mathbf{v}_1 \in \mathcal{V}_g^{1,i}$, the sequence $(\psi^n(\mathbf{v}_1))_n$ eventually leads to $\perp$. Since the cardinal of $\mathcal{V}_g^{1,i}$ is $\prod_{j \in [\![2,\, D]\!]} \mathbf{d}[j]$, there cannot exist a vertex $\mathbf{v}_2$ in slices $\mathcal{V}_g^{1,j}$ $(j < i)$ such that $\psi(\mathbf{v}_2) = \perp$, since ψ would not be minimal, thus leading to a contradiction.

Using this result, it follows that translations by Dirac vectors along the first dimension of the grid are minimal:

Proposition 19

Let $\psi_1 \in \Psi_{\mathcal{G}_g}$ be the Euclidean translation by $\boldsymbol{\delta}_1$ as introduced in Proposition 17 on a grid graph $\mathcal{G}_g$ respecting assumption (137). We have that ψ_1 is minimal.

Proof: Proposition 19

Let $\psi_2 \in \Psi_{\mathcal{G}_g}$ be a minimal translation on $\mathcal{G}_g$. Let $\mathcal{V}_g^{1,i}$ and $\mathcal{V}_g^{1,i+1}$ be two slices containing no vertex of which image through ψ_2 is $\perp$. Lemma 4 tells us that no vertex $\mathbf{v}$ in slices $\mathcal{V}_g^{1,j}$ $(j \leq i+1)$ has its image equal to $\perp$. Additionally Lemma 3 and Corrolary 2 indicate that for these vertices, $\psi_2(\mathbf{v}) = \mathbf{v} + \boldsymbol{\delta}_1$.

Now, let us consider the minimum $k > i+1$ such that $\exists \mathbf{v}_1 \in \mathcal{V}_g^{1,k} : \psi_2(\mathbf{v}_1) = \perp$. Corrolary 2 tells us that every vertex from $\mathcal{V}_g^{1,k-1}$ has its image through ψ_2 in $\mathcal{V}_g^{1,k}$. Now, let us distinguish two cases:

1. If $k = \mathbf{d}[1]$, then $\psi_2 = \psi_1$, for which the loss is equal to the upper bound on losses for minimal translations;
2. If $k < \mathbf{d}[1]$, then we proceed by contradiction. By Corrolary 2, we have that $\psi_2(\mathbf{v}_1 - \boldsymbol{\delta}_1) = \mathbf{v}_1$. Since $\mathcal{N}(\mathbf{v}_1 - \boldsymbol{\delta}_1) \cap \mathcal{N}(\mathbf{v}_1 + \boldsymbol{\delta}_1) = \{\mathbf{v}_1\}$, and since $\psi_2(\mathbf{v}_1) = \perp$, we obtain that $\mathbf{v}_1 + \boldsymbol{\delta}_1$ cannot be the image of any vertex. As a consequence, we have a sequence $(\psi_2^n(\mathbf{v}_1 + \boldsymbol{\delta}_1))_n$ that ends with $\perp$. Therefore, the loss of ψ_2 is at least $1 + \prod_{j \in [\![2,\, D]\!]} \mathbf{d}[j]$, and ψ_2 is not minimal.

Similarly, we can show that translations by Dirac vectors along the other dimensions of the grid are pseudo-minimal:

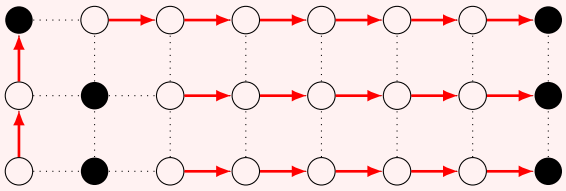
Corrolary 3

Let $\mathcal{G}_g$ be a grid graph respecting assumption (137). Euclidean translations by $\boldsymbol{\delta}_i$ ($i \in [\![1, D]\!]$) are pseudo-minimal.

Proof: Corrolary 3

Let us denote by ψ_i the translation of every vertex by $\boldsymbol{\delta}_i$ as introduced in Proposition 17. Proposition 19 shows that ψ_1 is minimal, hence pseudo-minimal.

Now, let us consider a translation $\psi \in \Psi_{\mathcal{G}_g}$ such that $\forall \mathbf{v} \in \mathcal{V}_g : \psi(\mathbf{v}) \neq \mathbf{v} + \boldsymbol{\delta}_1$. For such translations, we can perform the same reasoning as above, leading to ψ_2 being pseudo-minimal. However, in the case where such a vertex exists, we can have the situation where $\psi_2 \prec \psi$. The following illustration depicts such a possible ψ:



In this situation, ψ is not pseudo-minimal since $\psi \prec \psi_1$. It follows that ψ_2 is pseudo-minimal. The same reasoning can be made for all higher dimensions.

Finally, recall from Conjecture 1 that we believe all the results in this section apply for grid graphs such that $\forall i \in [\![1, D]\!] : \mathbf{d}[i] \geq 6$. Interestingly, for smaller dimensions, counterexamples as depicted in Figure 47 can be found. The depicted translations are minimal, while not being Euclidean translations by $\boldsymbol{\delta}_i$ as introduced in Proposition 17:

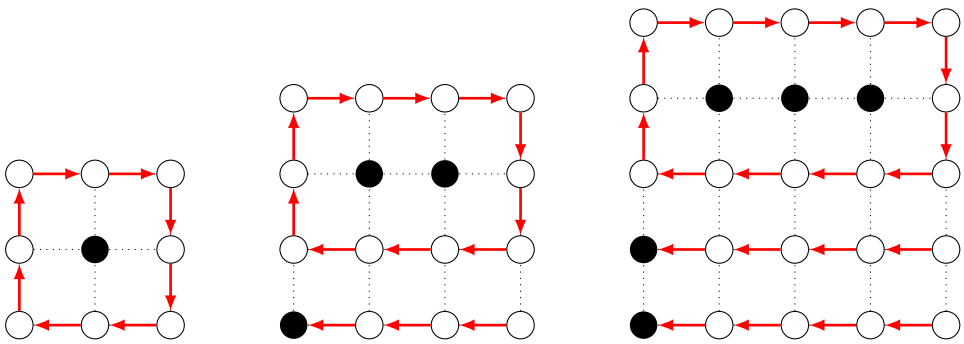


Figure 47 – Counterexamples for grid graphs of dimensions $\begin{bmatrix}3\\3\end{bmatrix}$ (left), $\begin{bmatrix}4\\4\end{bmatrix}$ (middle) and $\begin{bmatrix}5\\5\end{bmatrix}$ (right). For such graphs, the translations that are depicted are minimal, while not being Euclidean translations by $\boldsymbol{\delta}_i$ as introduced in Proposition 17.

4.3.4 Extension to generic graphs

In Proposition 4.3.1, we have shown that finding translations is an NP-complete problem. However, Section 4.3.2 and Section 4.3.3 have shown that in some particular cases, identifying the minimal or pseudo-minimal translations is possible in a minimum amount of time.

In this section, we are interested in generalizing the results we obtained for grid graphs to generic graphs. To do so, let us first show the following result:

Proposition 20

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a graph of adjacency matrix $\mathbf{A}$, and let $\psi \in \Psi_{\mathcal{G}}$ be a translation on $\mathcal{G}$. Let $\overrightarrow{\mathcal{G}}^{\psi} = \langle \mathcal{V}, \overrightarrow{\mathcal{E}}^{\psi} \rangle$ be the digraph associated with ψ, of adjacency matrix $\mathbf{A}_{\psi}$. The following propositions are equivalent:

1. For every connected component of vertices $\mathcal{V}_1 \subseteq \mathcal{V}$ in $\mathcal{G}$, we have either $\forall v \in \mathcal{V}_1 : \psi(v) = \perp$, or $\forall v \in \mathcal{V}_1 : \psi(v) \neq \perp$;
2. $\mathbf{A}\mathbf{A}_{\psi} = \mathbf{A}_{\psi}\mathbf{A}$.

Proof: Proposition 20

Without loss of generality, let us consider that $\mathcal{G}$ is connected, *i.e.*, has only one connected component.

1. $\Rightarrow$ 2.:
   - If $\psi = \phi_{\perp}$, then $\mathbf{A}_{\psi}$ is a matrix full of zeros, and the equality in 2. holds;
   - If ψ is lossless, let us consider a vertex $v_1 \in \mathcal{V}$. The equality in 2. is the matrix representation of $\mathcal{N}(\psi(v_1)) = \{\psi(v_2) \mid v_2 \in \mathcal{N}(v_1)\}$. Since translations are SNP, necessarily we have that $\{v_1, v_2\} \in \mathcal{E} \Leftrightarrow \{\psi(v_1), \psi(v_2)\} \in \mathcal{E}$, unless $\psi(v_1) = \perp$ or $\psi(v_2) = \perp$. Since we consider here lossless translations, we have the equivalence. As a conclusion, for any vertex $v_1 \in \mathcal{V}$, the translation of neighbors of v_1 (which in matrix notation translates to $\mathbf{A}_{\psi}\mathbf{A}$) are the neighbors of the translation of v_1 (which translates to $\mathbf{A}\mathbf{A}_{\psi}$);

2. $\Rightarrow$ 1.: If ψ is not lossless, there exists v such that $\psi(v) = \perp$. To verify equality 2., all neighbors of v must have their image equal to $\perp$. By contamination, $\psi = \phi_{\perp}$.

Following Proposition 20, and except for translation $\phi_{\perp}$, it is interesting to note that the number of non-null entries of $\mathbf{A}\mathbf{A}_{\psi} - \mathbf{A}_{\psi}\mathbf{A}$ is low when the loss of ψ is low. In particular, we remark the following properties:

1. For every vertex $v_1 \in \mathcal{V}$ such that $\nexists v_2 \in \mathcal{V} : \psi(v_2) = v_1$, we have:

$$\forall v_3 \in \mathcal{V} : (\mathbf{A}\mathbf{A}_{\psi})[v_3, v_1] = 0 \; ; \tag{142}$$

2. For every vertex $v_1 \in \mathcal{V}$ such that $\psi(v_1) = \perp$, we have:

$$\forall v_2 \in \mathcal{V} : (\mathbf{A}_{\psi}\mathbf{A})[v_1, v_2] = 0 \; . \tag{143}$$

Basically, $\mathbf{A}\mathbf{A}_{\psi}$ is a reorganization of the columns of $\mathbf{A}$, except for vertices that start a path, for which the associated column becomes null (item 1). Similarly, $\mathbf{A}_{\psi}\mathbf{A}$ is a reorganization of the rows of $\mathbf{A}$, except for vertices that have their image equal to $\perp$, for which the associated row becomes null (item 2). It follows that there are a number

of non-null entries in $\mathbf{A}\mathbf{A}_\psi - \mathbf{A}_\psi\mathbf{A}$ proportional to the number of neighborhoods that are lost (due to a neighbor having its image equal to $\perp$) or created (due to a neighbor having no inverse image by ψ).

From these remarks, we can derive a method to estimate translations that minimize the loss, through the following optimization problem, where $\mathbf{A}^*_\psi$ is the adjacency matrix of the digraph associated with the target translation, C is a constant, and $\|\cdot\|_1$ is the entrywise ℓ_1 norm for matrices:

$$\mathbf{A}^*_\psi = \underset{\mathbf{A}_\psi \in \{0,1\}^{N\times N}}{\operatorname{argmin}} \|\mathbf{A}\mathbf{A}_\psi - \mathbf{A}_\psi\mathbf{A}\|_1 \quad s.t. \quad \begin{cases} \forall i,j \in [\![1,N]\!] : \mathbf{A}_\psi[i,j] \leq \mathbf{A}[i,j] \\ \forall j \in [\![1,N]\!] : \sum_{i=1}^{N} \mathbf{A}_\psi[i,j] \leq 1 \\ \forall i \in [\![1,N]\!] : \sum_{j=1}^{N} \mathbf{A}_\psi[i,j] \leq 1 \\ \|\mathbf{A}_\psi\|_1 = C \end{cases} . \tag{144}$$

The first constraint imposes non-null entries of $\mathbf{A}_\psi$ to be a subset of non-null entries of $\mathbf{A}$, thus enforcing the EC property of translations. Additionally, the constraints enforcing rows and columns to sum to at most 1 force injectivity of the solution. Finally, the fourth constraint is necessary to avoid the trivial solution where $\mathbf{A}_\psi$ is a matrix full of zeros. Since ψ is injective, we have $C \in [\![0,N]\!]$.

Note that no constraints enforce the SNP property of the solution, which can thus be a transformation that is not a translation. Still, minimizing the sacrifice of this property is the objective of the problem, as explained above. The solution of (144) is therefore an EC transformation that is as close as possible to being SNP. We will thus call solutions to (144) ***pseudo-translations***.

Due to the sacrifice of SNP, and because it is necessary to fix a norm for $\mathbf{A}_\psi$, we need a measure to describe which value of C is the correct one. In practice, we solve (144) for every possible value of C in $[\![0,N]\!]$, and keep the associated pseudo-translation ψ_C that minimizes the following quantity:

$$|\mathcal{E}_{\text{lost}} \cup \mathcal{E}_{\text{created}}| , \tag{145}$$

with:

$$\mathcal{E}_{\text{lost}} = \{\{v_1,v_2\} \in \mathcal{E} \mid (\psi_C(v_1) = \perp) \vee (\psi_C(v_2) = \perp) \vee (\{\psi_C(v_1),\psi_C(v_2)\} \notin \mathcal{E})\} , \tag{146}$$

and:

$$\mathcal{E}_{\text{created}} = \{\{v_1,v_2\} \notin \mathcal{E} \mid (\psi_C(v_1) \neq \perp) \wedge (\psi_C(v_2) \neq \perp) \wedge (\{\psi_C(v_1),\psi_C(v_2)\} \in \mathcal{E})\} . \tag{147}$$

The quantity measured by (145) is the exact error of the pseudo-translation ψ_C, *i.e.*, it is the total number of neighborhoods that are lost or created by ψ_C, and should not be. Note that this quantity does not depend on C. Therefore, it provides a way of selecting the best pseudo-translation found, without a bias related to the imposed norm in (144).

Once a pseudo-translations ψ_1 is found by solving (144) with the value of C minimizing (145), we add additional constraints to (144) in order to prevent subsequent solutions to share edges with ψ_1. While this prevents some pseudo-minimal translations to be found in the general case — consider for instance the case of the complete graph — this gives us a greedy and efficient algorithm to find a set of pseudo-translations, by increasing progressively the set of constraints to restrict the possible neighborhood of

previously found pseudo-translations. In more details, let $\{\psi_1, \dots, \psi_i\}$ be the i first translations found by solving (144). For an optimization variable $\mathbf{A}_\psi$, such constraints are as follows:

$$\forall \psi \in \{\psi_1, \dots, \psi_i\}, \forall v \in \mathcal{V} : \mathbf{A}_\psi[v, \psi(v)] = 0 \ . \tag{148}$$

Algorithm 1 summarizes the whole process to identify pseudo-translations on a graph:

Algorithm 1: $findPseudoTranslations\ (\mathcal{G})$

```
result := {};
do
    ψ_best := φ_⊥;
    error_best := ∞;
    foreach C ∈ [[0, N]] do
        ψ := solve (144) with ‖A_ψ‖_1 = C;
        error := compute (145) for ψ and C;
        if error ≤ error_best then
            ψ_best := ψ;
            error_best := error;
    result := result ∪ {ψ_best};
    Update (144) with constraints (148) using result;
while ψ_best ≠ φ_⊥;
return result;
```

Although we will not study the following, it is interesting to remark that the constraints in (144) can be relaxed more to sacrifice the EC property in some extent. This would allow pseudo-translations to artificially *create* edges in the graph in order to minimize the overall error. In such case, the corresponding optimization problem can be written as follows:

$$\mathbf{A}_\psi^* = \underset{\mathbf{A}_\psi \in \{0,1\}^{N \times N}}{\operatorname{argmin}} \|\mathbf{A}\mathbf{A}_\psi - \mathbf{A}_\psi\mathbf{A}\|_1 + \ \alpha\|\mathbf{A} - \mathbf{A}_\psi\|_1$$

$$s.t. \begin{cases} \forall j \in [\![1, N]\!] : \sum_{i=1}^{N} \mathbf{A}_\psi[i,j] \leq 1 \\ \forall i \in [\![1, N]\!] : \sum_{j=1}^{N} \mathbf{A}_\psi[i,j] \leq 1 \\ \|\mathbf{A}_\psi\|_1 = C \end{cases}, \tag{149}$$

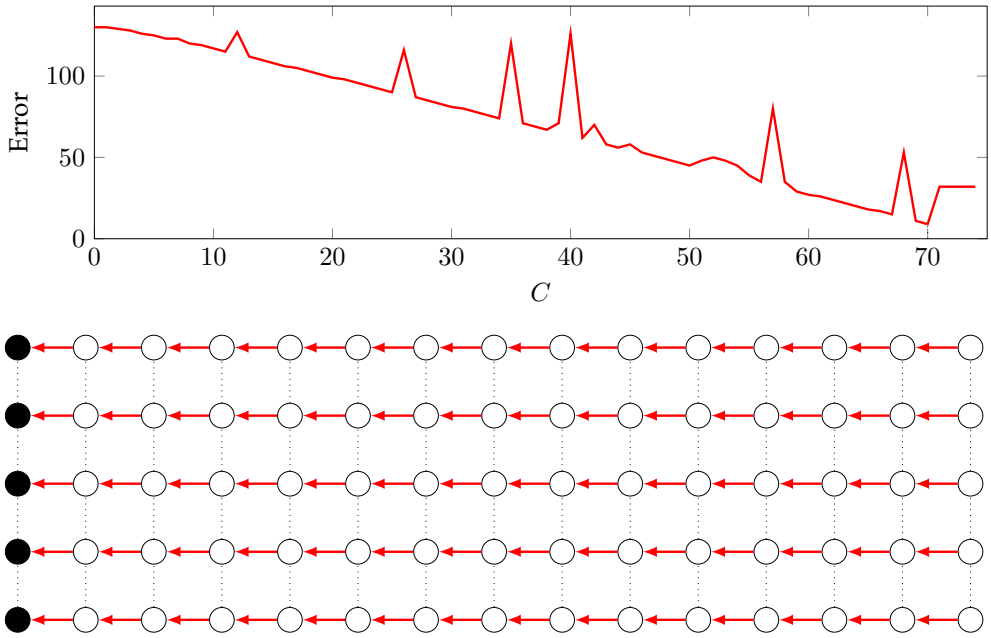
where α is a regularization parameter, and $\|\mathbf{A} - \mathbf{A}_\psi\|_1$ replaces the constraint enforcing non-null entries of $\mathbf{A}_\psi$ to be a subset of those of $\mathbf{A}$, by encouraging it in the objective function. Note that it also requires to update (145) to take the number of violations of the EC property into consideration.

4.4 Finding translations on complex graphs

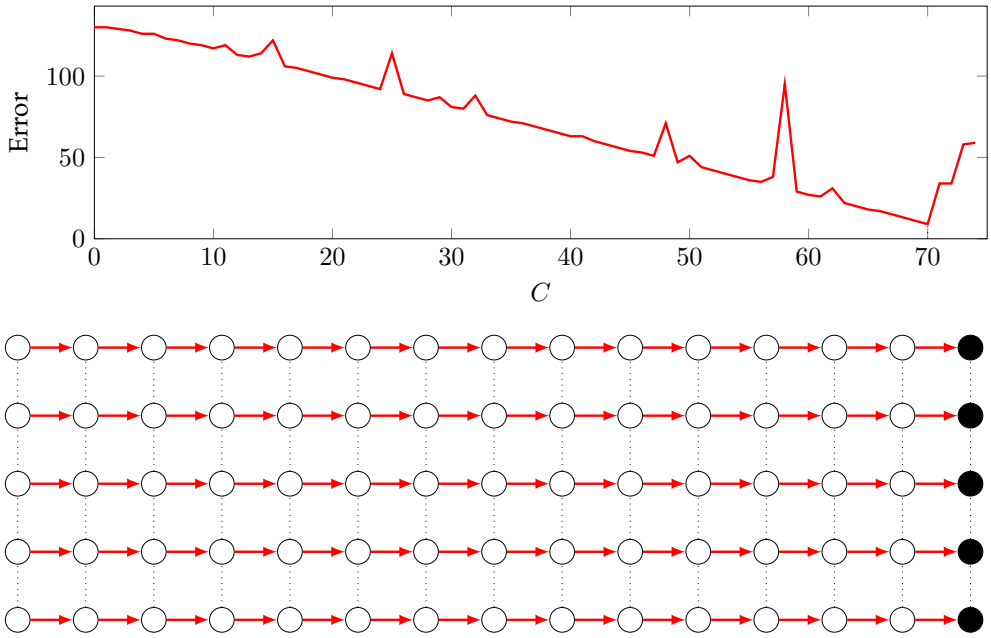
In this section, we first evaluate the results obtained with Algorithm 1 for various families of graphs. We illustrate that it finds the pseudo-minimal translations on a grid graph, and then we study the impact of small transformations of such graph on the translations that are found. Finally, we apply Algorithm 1 to identify translations on randomly generated graphs following a Watts-Strogatz model. In our experiments, Algorithm 1 is implemented using CVX [GBY08] package for MATLAB [MAT12].

4.4.1 Experiments on the grid graph

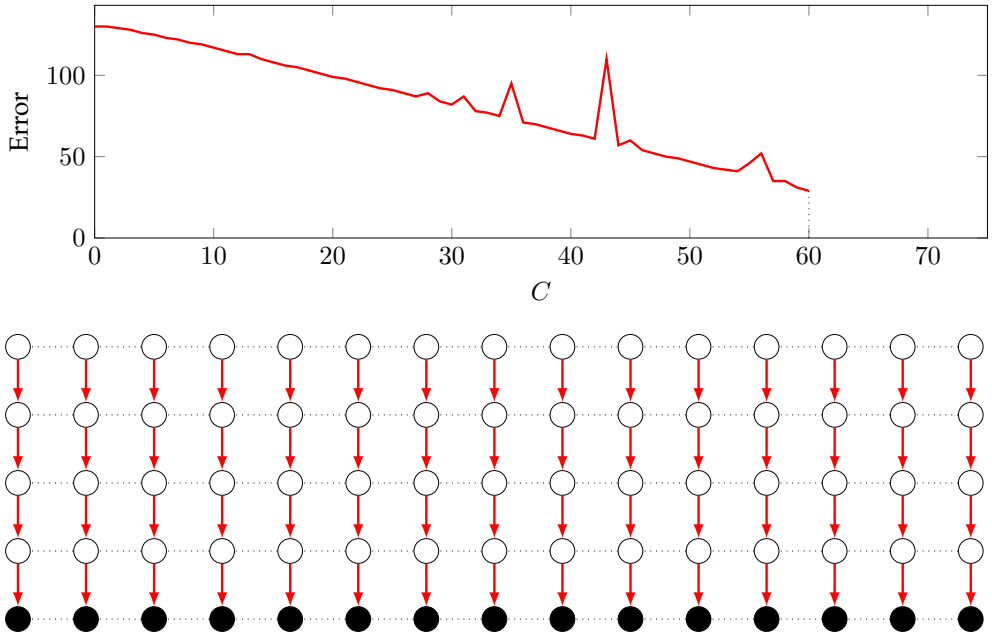
First, we consider a grid graph of dimensions $\mathbf{d} = \begin{bmatrix} 15 \\ 5 \end{bmatrix}$ respecting assumptions (137). Figure 48 depicts the errors (145) obtained for each value of C on this graph, as well as the pseudo-translations that correspond to the norm minimizing this error:



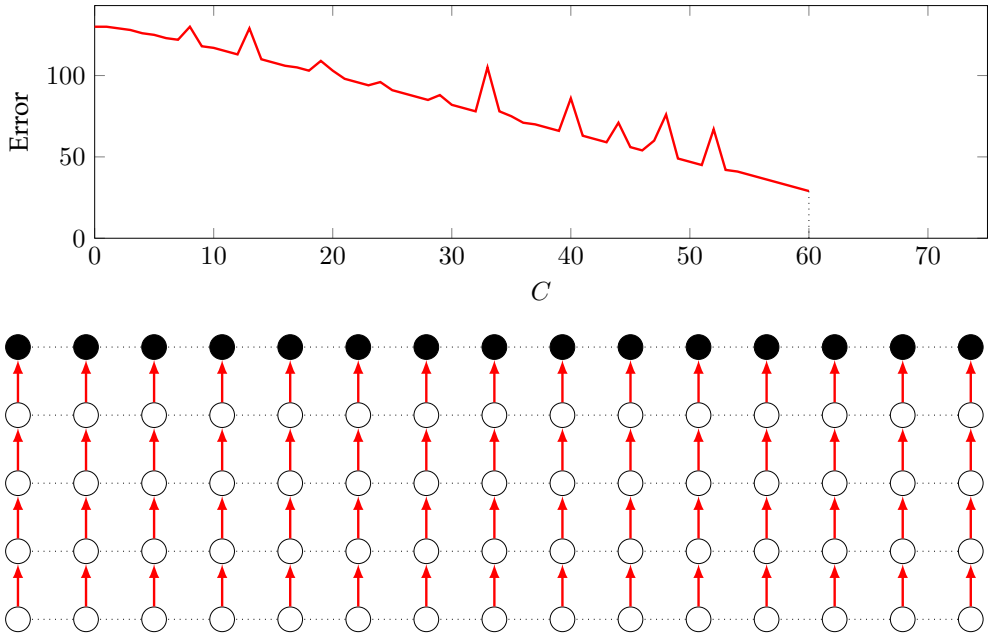
(a) First pseudo-translation found by Algorithm 1 for $C = 70$: ψ_1.



(b) Second pseudo-translation found by Algorithm 1 for $C = 70$: ψ_1^{-1}.



(c) Third pseudo-translation found by Algorithm 1 for $C = 60$: ψ_2.



(d) Fourth pseudo-translation found by Algorithm 1 for $C = 60$: ψ_2^{-1}.

Figure 48 – Pseudo-translations found by Algorithm 1 on a grid graph of dimensions $\mathbf{d} = \begin{bmatrix} 15 \\ 5 \end{bmatrix}$. The fifth pseudo-translation found, $\phi_\perp$, is not depicted. The left column depicts the error in (145) as a function of C. Values of (145) for which no solution exists are not depicted on the curve. The right column represents the translation associated with the value of C minimizing this quantity.

As expected, the algorithm finds all the pseudo-minimal translations on the grid graph. Similar results have been observed for grid graphs of different dimensions, provided that all dimensions are larger than 6, corresponding to Conjecture 1.

To evaluate whether the algorithm is robust to small variations of the graph, we consider a deformation of the grid graph $\mathcal{G}_g = \langle \mathcal{V}_g, \mathcal{E}_g \rangle$ of dimensions $\mathbf{d} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$. More precisely, we build a deformed grid graph $\mathcal{G}_{g\sigma} = \langle \mathcal{V}_{g\sigma}, \mathcal{E}_{g\sigma} \rangle$ as follows:

- $\forall \mathbf{v} \in \mathcal{V}_g : \mathbf{v} + \boldsymbol{\epsilon} \in \mathcal{V}_{g\sigma}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}\left(0, \sigma^2 \mathbf{I}_D\right)$;
- $\forall \{\mathbf{v}_1, \mathbf{v}_2\} \in \binom{\mathcal{V}_{g\sigma}}{2} : (\{\mathbf{v}_1, \mathbf{v}_2\} \in \mathcal{E}_{g\sigma}) \Leftrightarrow \left(d(\mathbf{v}_1, \mathbf{v}_2) < \frac{\sqrt{2}+1}{2}\right)$, where $d(\mathbf{v}_1, \mathbf{v}_2)$ is the Euclidean norm between vertices $\mathbf{v}_1$ and $\mathbf{v}_2$.

Then, we solve (144) once on this deformed grid to find a pseudo-translation ψ, of associated matrix $\mathbf{A}_\psi$. Let ψ_i be a translation by $\boldsymbol{\delta}_i$ on a non-deformed grid graph as introduced in Proposition 17, of associated matrix $\mathbf{A}_{\psi_i}$. The following quantity measures the difference between ψ and the closest translation ψ_i (or ψ_i^{-1}):

$$\min_{i \in [\![1, D]\!], j \in \{-1, 1\}} \left\| \mathbf{A}_{\psi_i^j} - \mathbf{A}_\psi \right\|_1 \tag{150}$$

where j allows the consideration of inverse Dirac translations.

Figure 49 depicts the difference (150) between ψ and the closest translation by a Dirac, as a function of the standard deviation σ controlling the deformation of the grid. For every value of $\sigma \in [0.03, 0.12]$ with a range of 0.005, we compute the mean difference for 100 randomly deformed grid graphs:

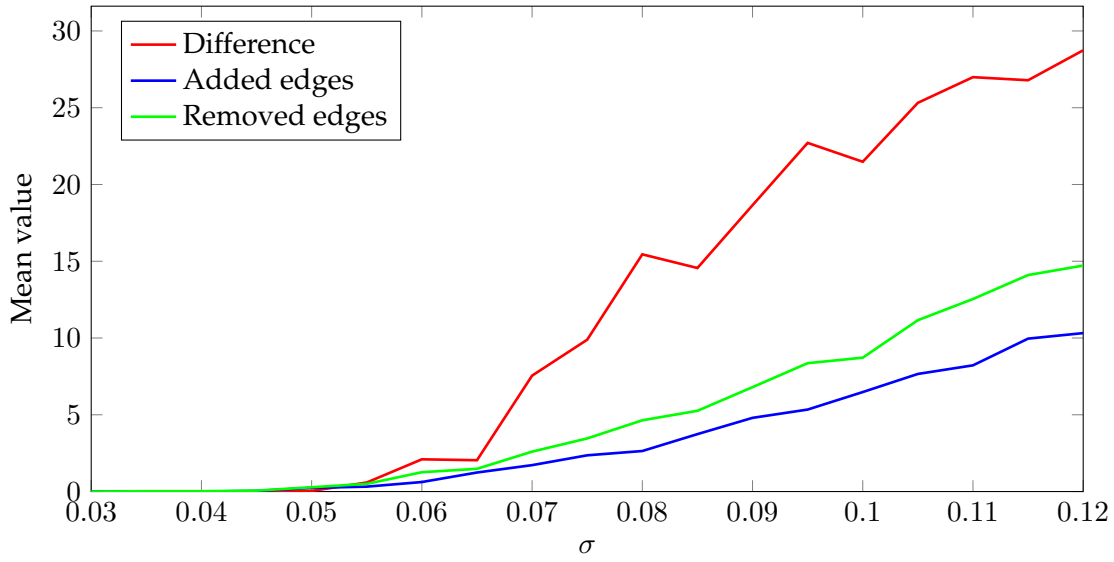


Figure 49 – Mean difference (150) between the first solution of (144) and a translation by a Dirac vector, as a function of the deformation of a grid of dimensions $\mathbf{d} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$.

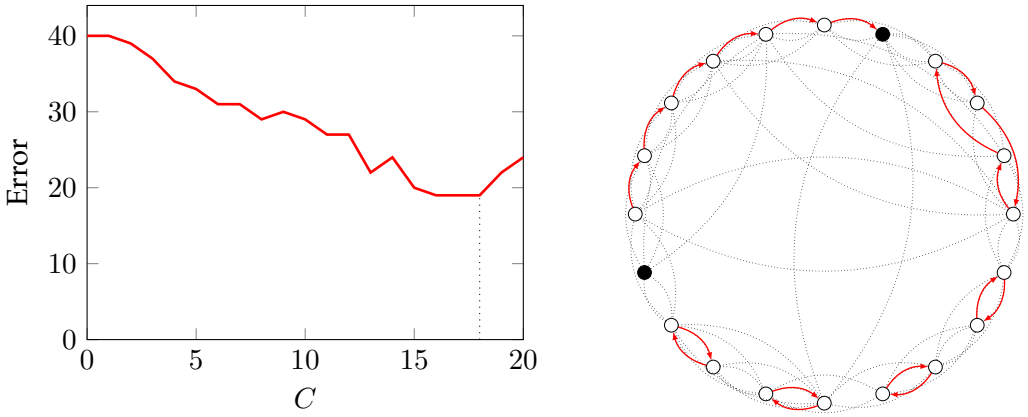
From Figure 49, we can see that small modifications on the grid do not strongly impact the translations that are found on it. In particular, around locations in the graph where an edge has been suppressed, we have observed that the solution of (144) tends to favor sending close vertices to $\perp$ rather than modifying the whole translation. However, when the modifications become too strong, the best solution becomes a translation that is not related to any translation by a Dirac vector, and we observe a strong increase of the difference (150).

In our observations, we have noticed that addition of edges has a smaller impact on the difference than suppression. This can easily be explained by the fact that relaxation of the problem by solving (144) sacrifices the SNP property, but still enforces pseudo-translations that are found to be EC. Therefore, when an edge is added, it results in a slightly larger loss in neighborhood, that may not change the overall result. On the contrary, when an edge is removed, as the EC property must be met, this implies sending vertices to $\perp$ or changing the whole solution.

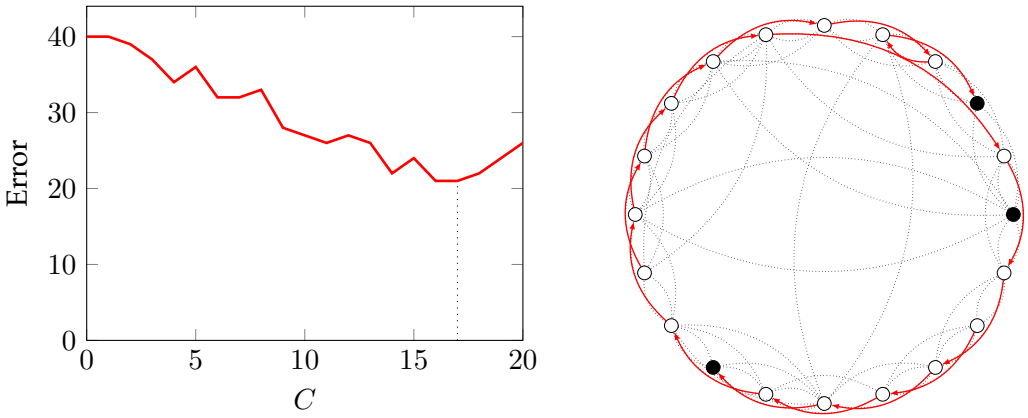
4.4.2 Translations on random graphs

In the previous experiments, we considered grid graphs, or small variations of it. In this section, we apply Algorithm 1 to randomly generated graphs in order to find pseudo-translations on different structures.

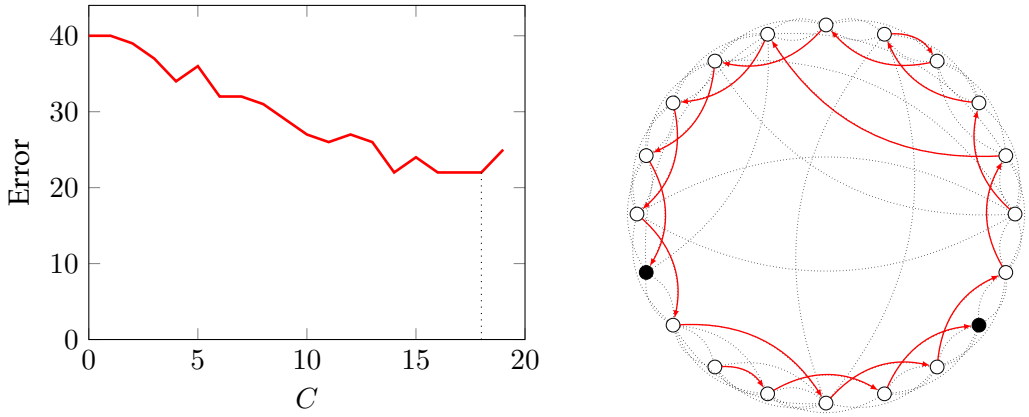
More particularly, we study the Watts-Strogatz model (see Definition 31). Figure 50 depicts the pseudo-translations that are found for a Watts-Strogatz graph of $N = 20$ vertices with $K = 2$ and $P = 0.1$:



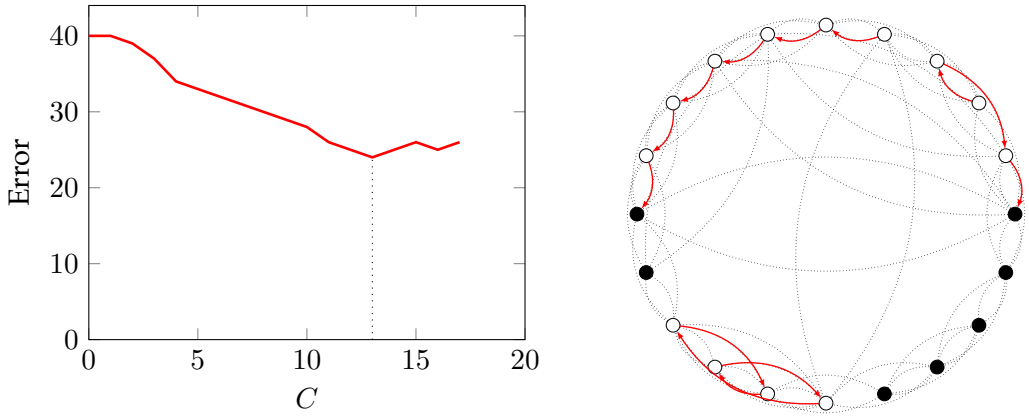
(a) $P = 0.1$, first pseudo-translation found for $C = 18$.



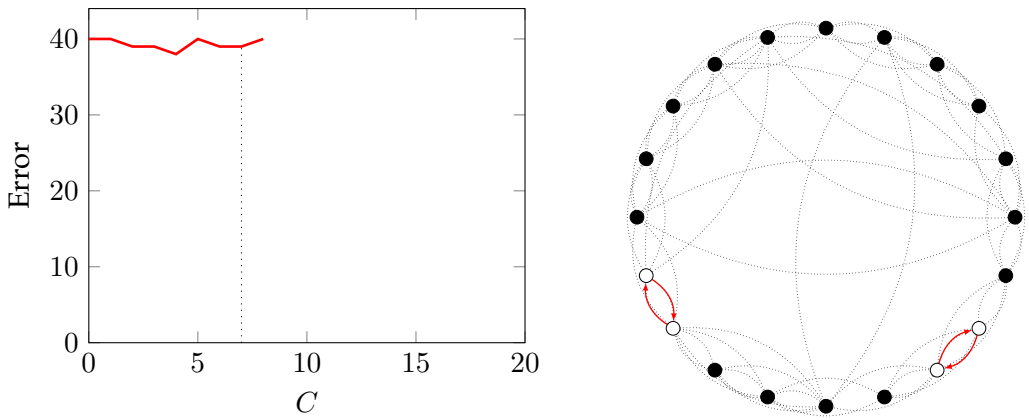
(b) $P = 0.1$, second pseudo-translation found for $C = 17$.



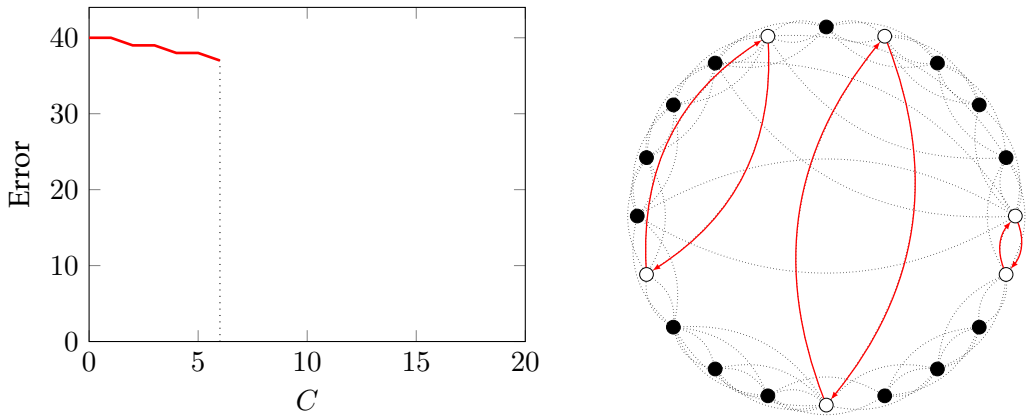
(c) $P = 0.1$, third pseudo-translation found for $C = 18$.



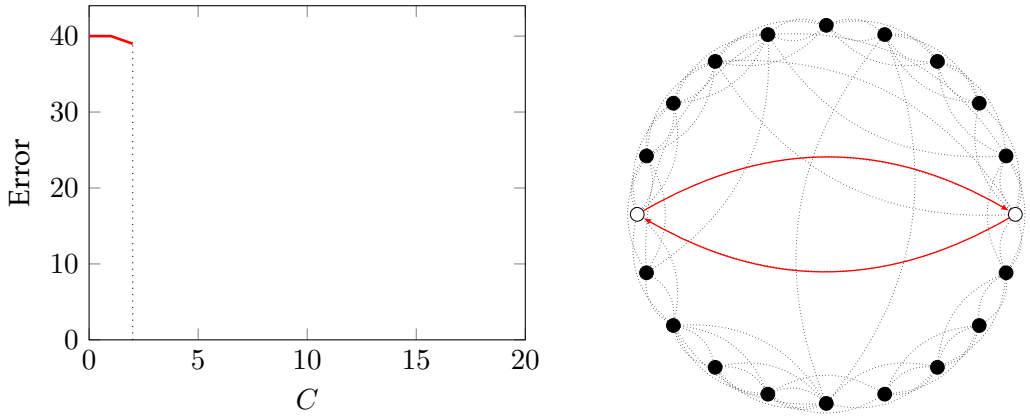
(d) $P = 0.1$, fourth pseudo-translation found for $C = 13$.



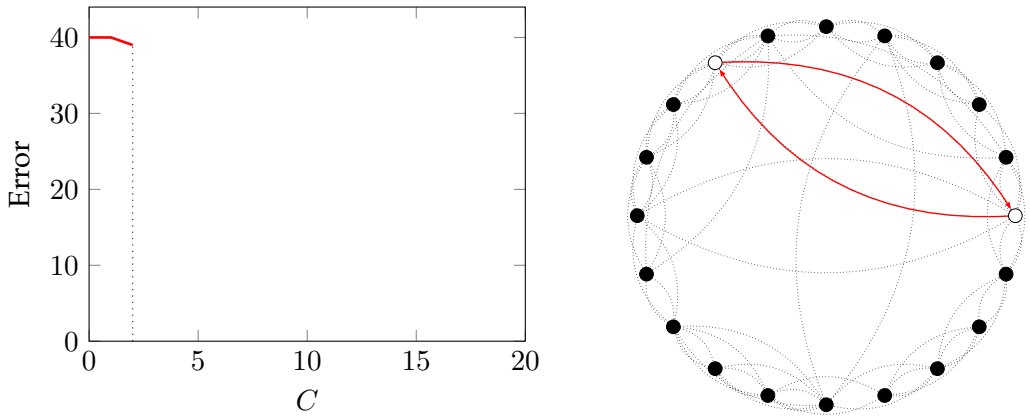
(e) $P = 0.1$, fifth pseudo-translation found for $C = 4$.



(f) $P = 0.1$, sixth pseudo-translation found for $C = 6$.



(g) $P = 0.1$, seventh pseudo-translation found for $C = 2$.



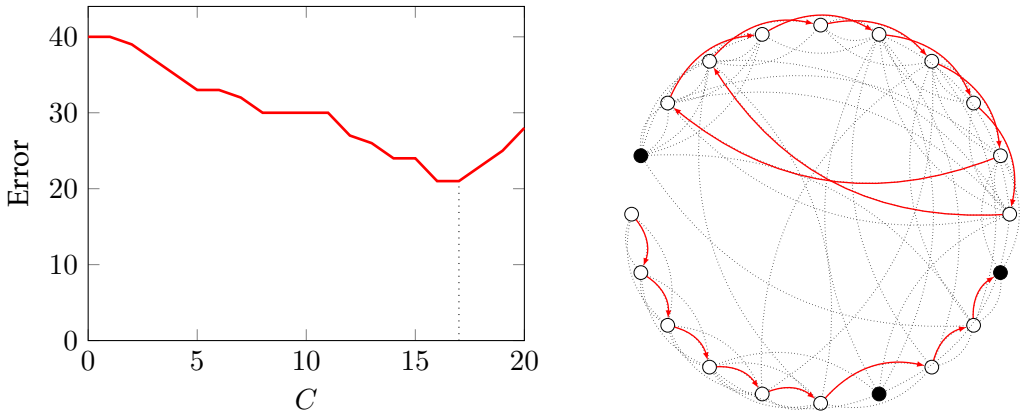
(h) $P = 0.1$, eighth pseudo-translation found for $C = 2$.

Figure 50 – Pseudo-translations found for random graphs following a Watts-Strogatz model with parameters $K = 2$ and $P = 0.1$. Error is computed according to (145). The translation that sends every vertex to $\perp$ is not depicted.

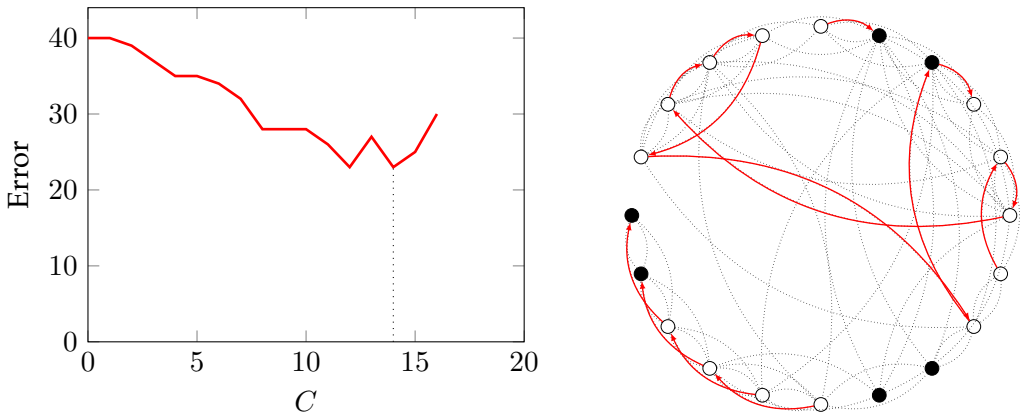
It appears that Algorithm 1 first finds pseudo-translations that are very significant. In particular, since P is low, the graph is close to a ring graph, and the first pseudo-translations that are found tend to follow edges along the border of the graph. After the

first few, pseudo-translations that are found appear to be *residuals*, as they can only use the edges that do not appear in the previously found ones, due to the greedy strategy of Algorithm 1. Still, these residual pseudo-translations favor edges linking vertices that belong to subsets of higher connectivity when it is possible, resulting in appearance of small cycles or small paths.

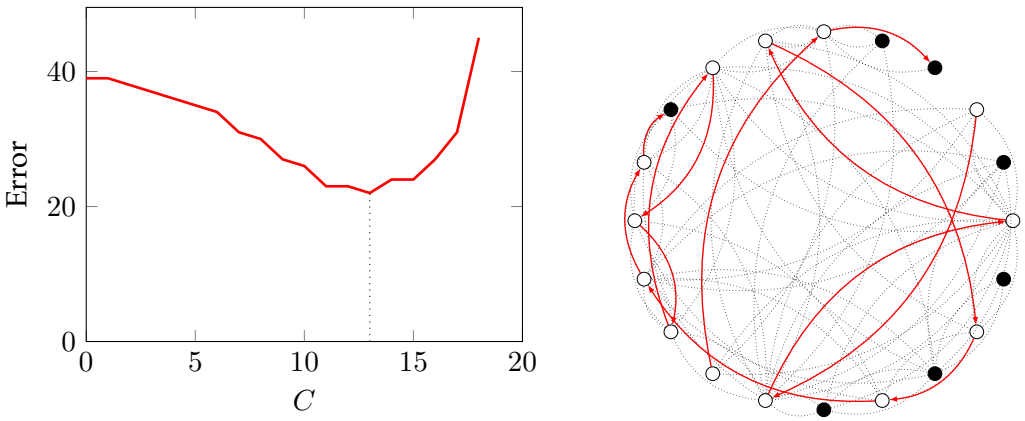
Figure 51 depicts the results we obtain for Watts-Strogatz graphs associated with larger probabilities $P \in \{0.3, 0.5, 0.7, 0.9\}$. Obviously, the results are not as good as for low values of P, as it can be seen from the curves in Figure 51. However, it is interesting to notice that the pseudo-translations continue to consist of paths in most cases and therefore make sense, as vertices sharing common neighbors tend to be sent from one to another:



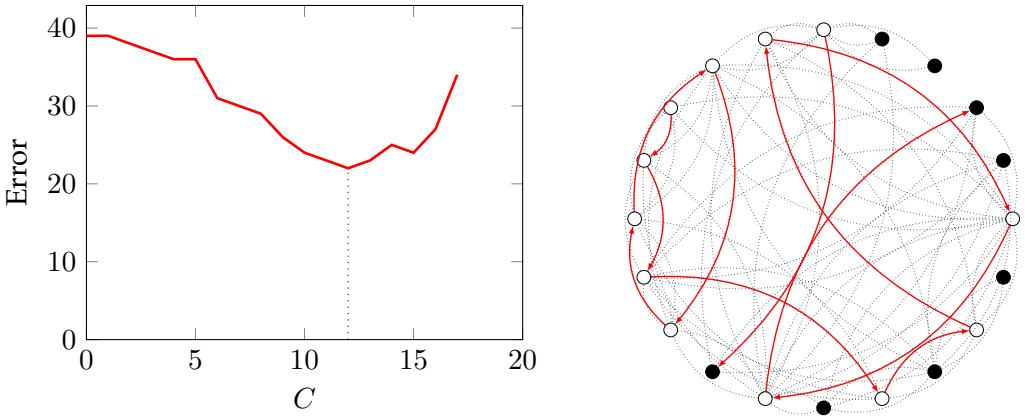
(a) $P = 0.3$, first pseudo-translation found for $C = 17$.



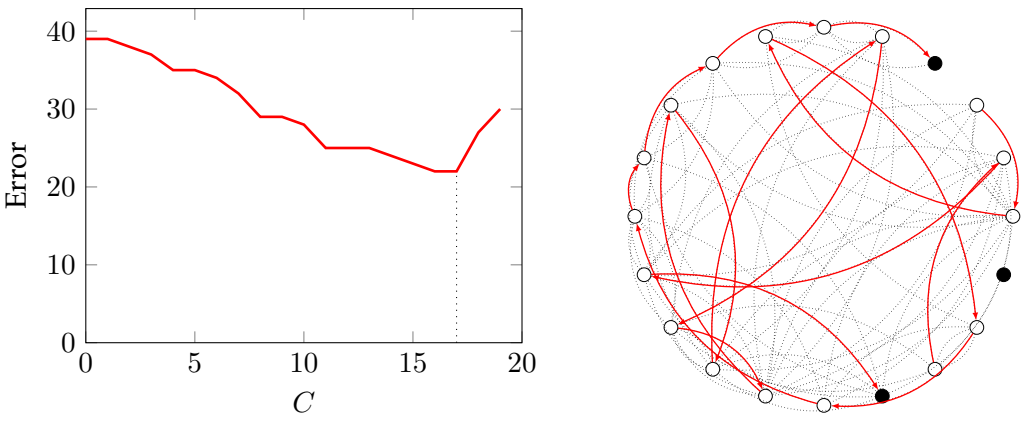
(b) $P = 0.3$, third pseudo-translation found for $C = 14$. The second pseudo-translation found was the inverse of the first.



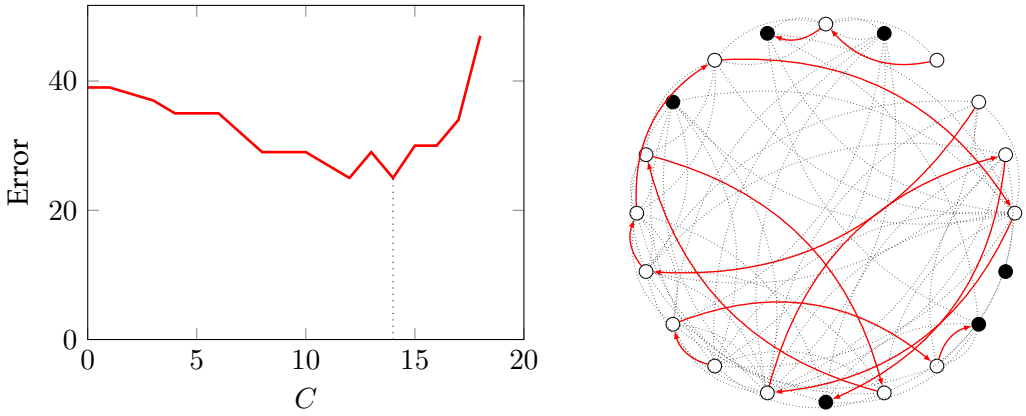
(c) $P = 0.5$, first pseudo-translation found for $C = 13$.



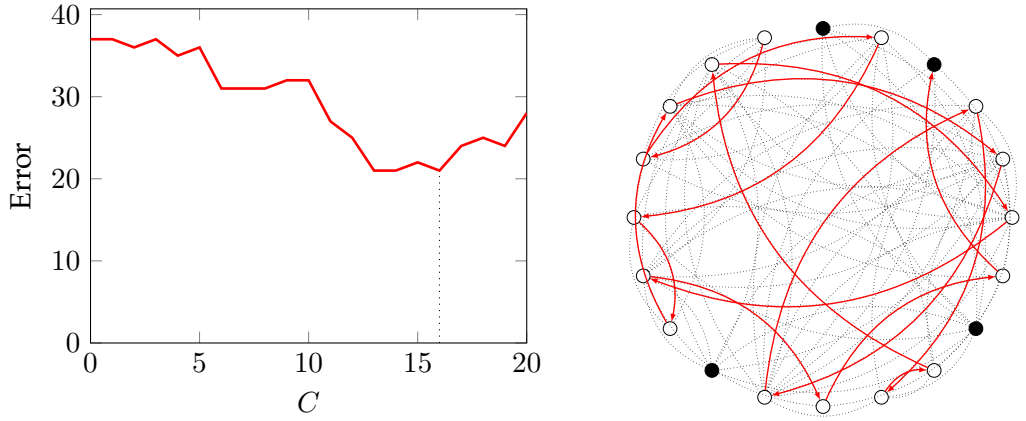
(d) $P = 0.5$, second pseudo-translation found for $C = 12$.



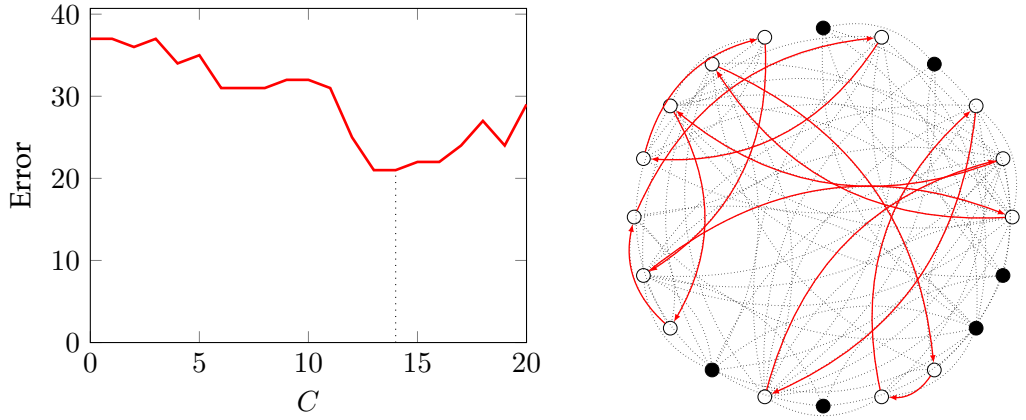
(e) $P = 0.7$, first pseudo-translation found for $C = 17$.



(f) $P = 0.7$, third pseudo-translation found for $C = 14$. The second pseudo-translation found was the inverse of the first.



(g) $P = 0.9$, first pseudo-translation found for $C = 16$.



(h) $P = 0.9$, second pseudo-translation found for $C = 14$.

Figure 51 – Pseudo-translations found for random graphs following a Watts-Strogatz model with parameters $K = 2$ and $P \in \{0.3, 0.5, 0.7, 0.9\}$. Error is computed according to (145). For each value of P, only the two first distinct pseudo-translations that are found are depicted.

4.5 Summary of the chapter

In this chapter, we have proposed a definition of translations on graphs that, contrary to existing definitions, matches the intuitive translations on Euclidean spaces. Such translations are defined to match some neighborhood-preserving properties, thus keeping the overall coherence of the signal that is translated, if possible without altering it. When the signal cannot be altered, we have relaxed our definition to allow the disappearance of some signal entries, as one would expect when translating an image on a non-toric grid of pixels for instance.

However, we have shown in this chapter that identification of these translations on graphs is an NP-complete problem. To circumvent this, we have introduced the notion of pseudo-translations, that sacrifice as little neighborhoods as possible. Finding these pseudo-translations can be done by solving an optimization problem for various matrix norms, and keeping the best norm found. Algorithm 1 details the whole process to identify them.

Finally, we have illustrated these translations on various graphs. On grid graphs, Algorithm 1 succeeds at finding the Euclidean translations introduced in Definition 4.3.3. Also, when studying small variations of the grid, our experiments have shown that the same translations could still be found, making them resistant to small noise. Finally, translations on partially regular graphs — such as Watts-Strogatz random graphs with low rewiring probability — appear to be quite regular, which is a desirable behavior.

We believe this work to have applications in tasks such as classification. Identifying translations on arbitrary graphs could allow the definition of adapted kernels for convolutional neural networks, in order to identify a same object at various locations of the graph. This idea is explored in the next chapter.

Chapter 5

Application to signals classification

Contents

In this chapter, we explore the combination of the two previous chapters to improve classification of signals on graphs.

Classification is a classical task in machine learning, consisting in associating a label — from a given set of labels — with a signal to classify. In more details, classification is an instance of ***supervised learning*** problems. In supervised learning, a function is learned from some labeled data (the ***training set***), and is then applied to new unlabeled examples (the ***test set***). In the case of classification, the function learned is generally called a classifier, as it takes a signal as an input and outputs a label for it indicating what the signal represents.

Examples of applications of classification are numerous, and are widely diffused in scientific vulgarization platforms nowadays. A well known example of classifier is Google Inception [Sze+16]. This classifier is based on *deep learning*, a particular subfamily of *artificial neural networks*, consisting in numerous layers as depicted in Figure 52:

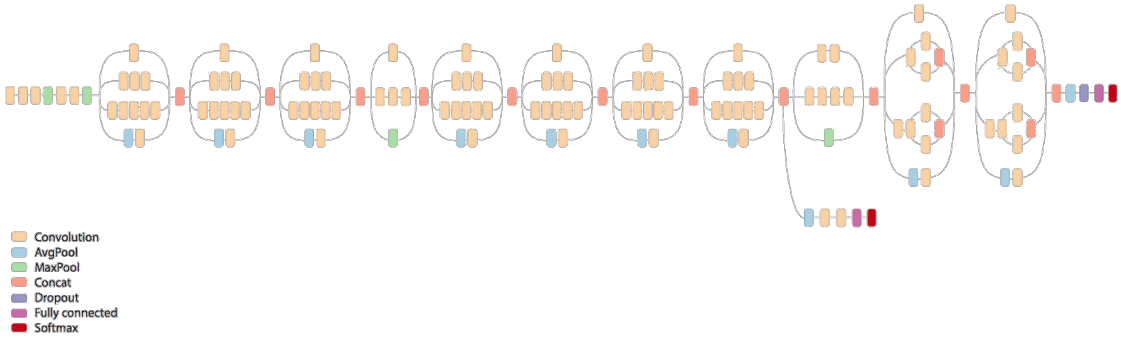


Figure 52 – Deep convolutional neural network used by the Google Inception classifier [Sze+16]. Figure taken from [Shl16].

In this chapter, a subfamily of neural networks we are particularly interested in are *convolutional neural networks* [LB+95]. These networks have proven to be very efficient in object classification thanks to the ability to learn representations from different locations in input signals, which through a slight misuse of language is generally called *shift invariance*.

The idea of this chapter is to explore how graph inference from data, followed by translations identification on the inferred graph, can help in a classification procedure of the initial signals. In few words, convolutional neural networks perform their detection by translating a kernel on the image under study. Identifying translations on a non-regular graph thus allows the same approach for signals evolving on complex topologies.

This chapter is organized as follows:

1. First, we introduce convolutional neural networks. In particular, we explain how they perform and introduce some related work on extending such networks to study data on irregular graphs;
2. Then, we study a complete example, in which a graph is inferred from images taken from the CIFAR-10 dataset [KH09]. Translations on this graph are then identified (see Chapter 4), and are used to move a kernel on the graph to define an adapted convolutional neural network. We show that in the case of image signals, our approach performs classification better than other methods that also do not use the information that signals are images. Additionally, we highlight that the performance we obtain does not suffer from shuffling the pixels of the input images, contrary to classical convolutional neural networks.

5.1 Convolutional neural networks

5.1.1 Some methods for classification of signals

Methods for classifying signals are numerous, as the problem has received a lot of interest since its inception in the work of Fisher [Fis36; Fis38]. Among the most classical methods (see [Kot07] for a more complete review of existing methods), we distinguish the following ones:

- ***K-nearest neighbors (K-NN)***: For each signal to classify, we find the K closest signals from the training set according to a certain distance function. The label associated with this signal is then the one appearing most in these close samples;
- ***Support vector machines (SVM)***: This method aims at finding an hyperplane in the space where data is defined, to create a delimitation between samples of the various classes. Signals to classify are then associated with a label by considering the side of the hyperplane they lie on;
- ***Classification trees***: The training data is recursively partitioned into homogeneous subsets, thus forming a tree of which leaves correspond to the various classes. When a signal is to be classified, search is performed starting from the root of the tree, and edges are taken iteratively in the direction of the subset that corresponds the most to the data to classify. Finally, once a leave is reached, the corresponding label is associated with the signal;
- ***Random forests***: This is an extension of classification trees. A certain number of trees are — partly randomly — created from the training data. Classification of a signal is then performed on each tree, and a majority vote decides the final label to associate with the signal;
- ***Naive Bayes***: This approach computes the probability for each feature of the training data to be representative of each class. Classification of unknown examples is then performed by finding the class with the maximum likelihood for every signal to classify.

There are numerous other methods, and additionally dozens of declinations of these methods. Additionally, learning can also be done using ***cross-validation***, meaning that the training set is splitted into two parts: the training set and the ***validation set***. In this case, learning is performed on a smaller set of samples, but the ability of the classifier to generalize to unseen examples is evaluated on the validation set.

Another family of classifiers we are interested in is the family of ***artificial neural networks***. In their most simple form, such classifiers consist of three main parts:

- An ***input layer***, consisting of as many vertices — sometimes called ***neurons*** [1] due to the analogy with biological neural networks — as there are entries in the data to classify;
- Some ***hidden layers***, of which number and sizes — *i.e.*, numbers of neurons per layer — are chosen arbitrarily;
- An ***output layer***, generally [Tig+16] consisting of one neuron per class.

1. When referring to vertices in layers of neural networks, we will use this term not to confuse with vertices of a graph.

Classically, each neuron is connected to all neurons of the next and previous layers with edges. Figure 53 presents an example of an artificial neural network, built to classify data defined in $\mathbb{R}^8$ into 6 distinct groups:

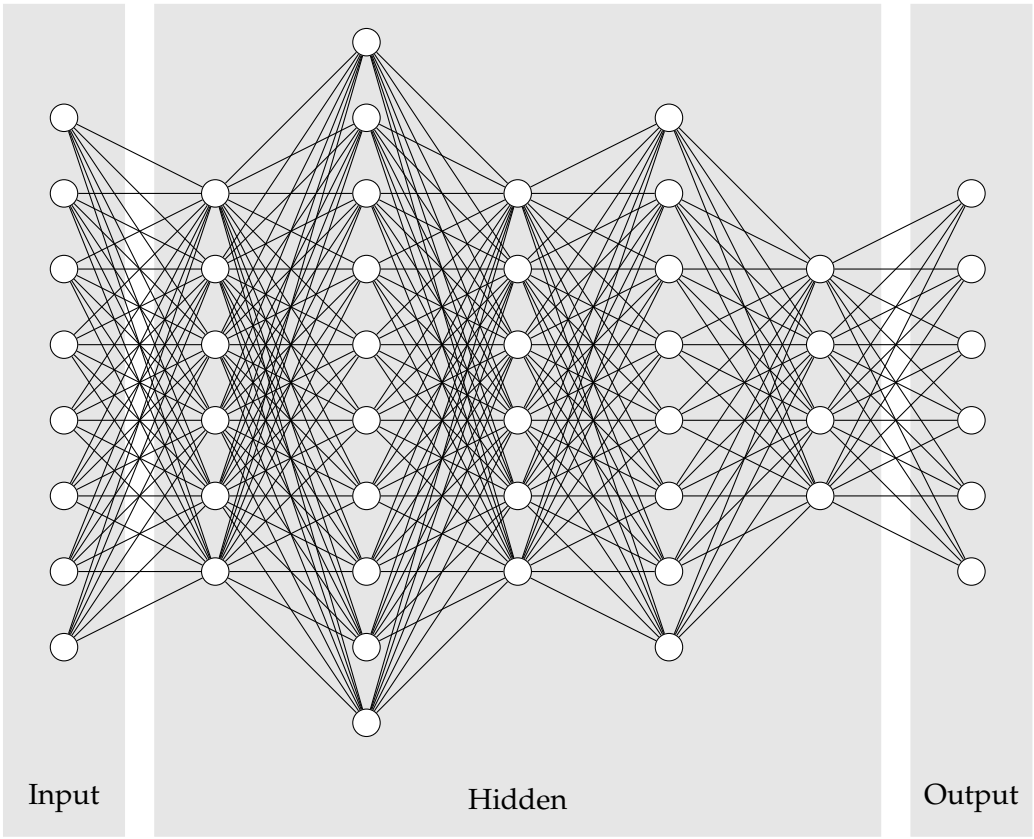


Figure 53 – Example of an artificial neural network. The input layer corresponds to signals in $\mathbb{R}^8$, which are to be classified between six labels. The network consists of four hidden layers, each containing a variable number of neurons.

Training of such network is performed by associating weights to the various edges. Generally, this is done using a method called ***back-propagation***, consisting in a correction of the weights to reduce the classification error during the training process.

Then, a signal is classified by feeding it to the network, and advancing from the input layer to the output one. At each layer, each neuron computes a linear combination of the signal entries at the previous layer, weighted using the connections between these layers. Once the output layer is reached, the neuron that has the maximum value activates, which gives the label to associate with the input signal.

Note that the description above is a highly simplified presentation of the process. As a matter of fact, there are numerous other specificities, such as the activation function used, or the presence of non-linearities when computing the activation score. Additionally, the initial weights distribution, or the number and sizes of layers, as well as the stop condition for the back-propagation also impact the classifier's performance.

5.1.2 Convolutional neural networks

Convolutional neural networks are in many aspects similar to the generic description of artificial neural networks given in Section 5.1.1. Such networks were designed to

process images, and therefore use the information that input signals are defined on a grid of pixels to drastically reduce the number of edges in the network, hence the number of variables.

Convolutional neural networks use the property that *objects* in images are very localized. Therefore, detection of a pattern can be performed by shifting a small, localized kernel on the image. In such networks, neurons from convolutional layers represent a local point of view. Therefore, a complete connection pattern as in Figure 53 is not necessary, which reduces the connectivity between all convolutional layers in the network.

As an example, Figure 54 depicts a portion of a convolutional neural network, obtained by translation of a 3×3 kernel on an input image:

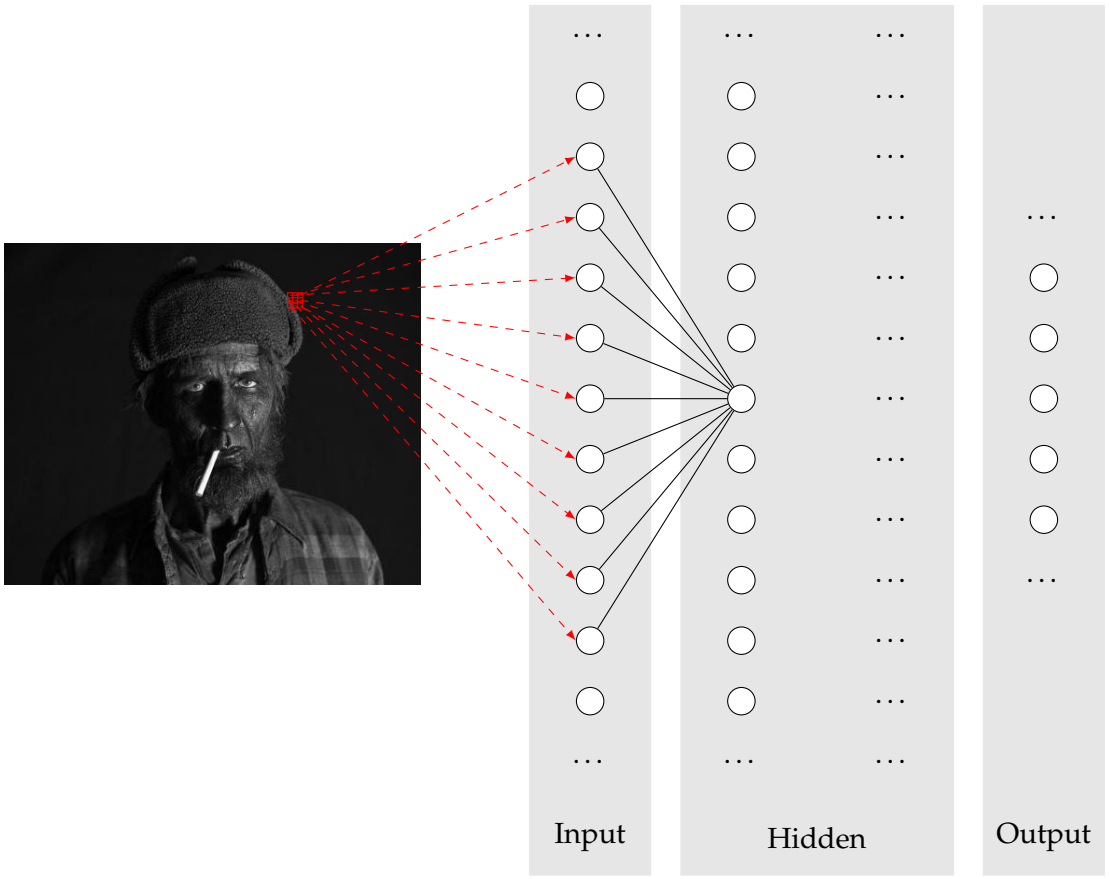


Figure 54 – Example of a portion of a convolutional neural network. Connections between the input layer and the first hidden layer are defined such that every neuron from the input layer associated with a pixel located in a 3×3 kernel is connected to the neuron in the first hidden layer associated with the center of the kernel.

Again, there are numerous details that will not be introduced in this manuscript. The interested reader may consult Stanford's CS231n course on convolutional neural networks for visual recognition [LJY17] for a more complete description of the mechanisms used in convolutional neural networks.

Still, it is important to notice that in many convolutional neural networks, there are some non-linearities between layers, that have proven to improve the learning process of the network. In particular, ***max-pooling*** is a non-linear, sample-based discretization process that allows one to reduce the dimension of information between two consec-

utive layers. When it comes to images, the sliding kernel is generally a rectangle of a few pixels, that can easily be divided into smaller rectangles. Then, for each of these smaller windows, the maximal output of the covered pixels is kept as a single value representing the window. Figure 55 illustrates this notion:

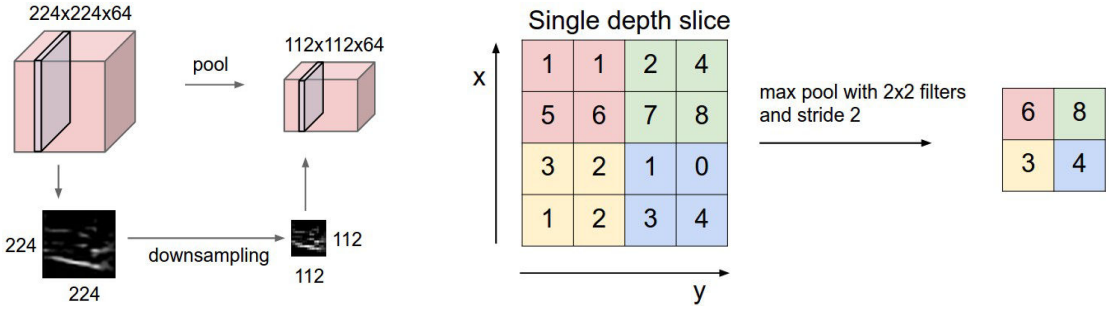


Figure 55 – Pooling reducing the dimension of the image from 224×224 pixels to 112×64 pixels, while keeping the same number of attributes per pixel (left). Max-pooling, in which a 4×4 window is reduced to 2×2, with entries represented by the maximal value in a part of the larger window (right). Figure taken from [LJY17].

Max-pooling however is not easy to define if the sliding kernel is not regular, which will be the case when we replace square kernels with custom ones on irregular graphs later in this chapter. An interesting option based on graph signal processing tools would be to use ***downsampling***, which is a common dimensionality reduction technique consisting in selecting a subset of the vertices of a graph. However, this is left for future work, and we choose in this chapter to consider convolutional neural networks that do not feature any max-pooling function.

The parameters defining the structure of a neural network are generally found using a ***grid search***, which can be understood as a near-exhaustive search in the space of parameters. For cost and time reasons, we do not have the ability to perform this search in our experiments. Therefore, in the subsequent sections, we perform our tests on a convolutional neural network found *by hand*, by tweaking the various network parameters. Our results appear to be interesting enough, but we believe they could be improved using a more complex architecture found through a grid search on the parameters.

5.1.3 Related work in extending convolutional neural networks to irregular domains

Being able to extend convolutional networks to generic graphs is currently receiving a lot of interest, as the potential applications are countless due to the observed performance of convolutional architectures for images or time signals. For this reason, some methods have recently been developed (see [Bro+17] for an overview of methods), that we group according to the convolution used. Note that none of the following methods defines translations on the graph, and therefore need a new definition for the convolution operator that allows the creation of the convolutional neural network. Since our method uses translations inferred on the graph, convolution can be done exactly as in the classical case [VGM16].

Convolution in the spectral domain

The missing results that would allow a proper definition of convolutional layers for graphs are a correct translation, or a proper convolution operator. We have shown in Section 2.3.1 that such operators were defined analogously to some properties in classical signal processing. As a reminder, translation is defined as a convolution with a Dirac delta signal on the graph, and convolution of signals on graphs is performed by doing an entrywise product in the spectral domain.

Using these graph signal processing tools in place of their Euclidean equivalents is the approach taken by Bruna *et al.* in [Bru+13], followed by Henaff *et al.* in [HBL15]. While their work provides an interesting idea on how to generalize convolutional neural networks to irregular data, the convolution is done using the spectral domain [Shu+13], and does not preserve the values and locality of the filter, as illustrated in Figure 41.

Following the same line, Defferrard *et al.* [DBV16] correct this locality problem by considering polynomial filters, which have the property to keep the localization properties after application of the translation operator presented in Definition 47. Still, the convolution used does not enforce the entries of the filters used to remain identical, as operations are performed in the spectral domain.

Convolution in the graph domain

More recently, Puy *et al.* [PKP17] have proposed a method that performs in the graph domain only. Convolution is performed by considering small patches around a vertex and performing a scalar product in the manner of [NAK16; Mon+16].

Translation however is not performed, and kernels are generated in various places of the graph by considering a given number of hops around a chosen center vertex. This is a problem, since by doing this, the authors cannot make a mapping between the vertices that belong to the various kernels, thus making the method not invariant by translation. Such a mapping can be decided arbitrarily [AT15], with random walks [HCQ17], or can be learned simultaneously [VGC17].

Puy *et al.* [PKP17] consider a slightly different problem, where the graph can change over time, and its various forms are known *a priori*. For this reason, they use a convolutional neural network architecture that takes a signal to classify as an input, as well as a graph on which the signal is defined. Other methods exist that only take a graph as an input [Duv+15; NAK16], which is a different problem.

In our approach, we do not assume knowledge of the underlying graph, which we infer from the data to classify. The convolutional neural network we define has therefore a more classical shape, as it takes a signal to classify as an input, and outputs a label for it. Also, the mapping is directly implied by the translation, which makes it equivalent to classical convolution on images.

The method we propose

Contrary to the methods above, the one we propose consists in inferring a graph, then identifying translations on the graph, and using these translations to determine a convolution operator for a convolutional neural network in the manner of classical images [VGM16]. It has the following properties:

- It performs in the graph domain only;
- Since our translations enforce neighborhood preservation properties, a translated kernel keeps the same shape and the same mapping, except for pixels having no image through the translation. It is then possible to define a tensor representing a convolution-like operator exactly as one would do for images.

5.2 A complete example on images

In this section, we apply our method in Section 5.1.3 to improve classification of signals. We consider here the CIFAR-10 dataset [KH09], consisting of a training set of 50.000 images of 32×32 images, and evaluated on a test set of 10.000 images. In this dataset, there are ten classes of objects, as summarized in Figure 56:

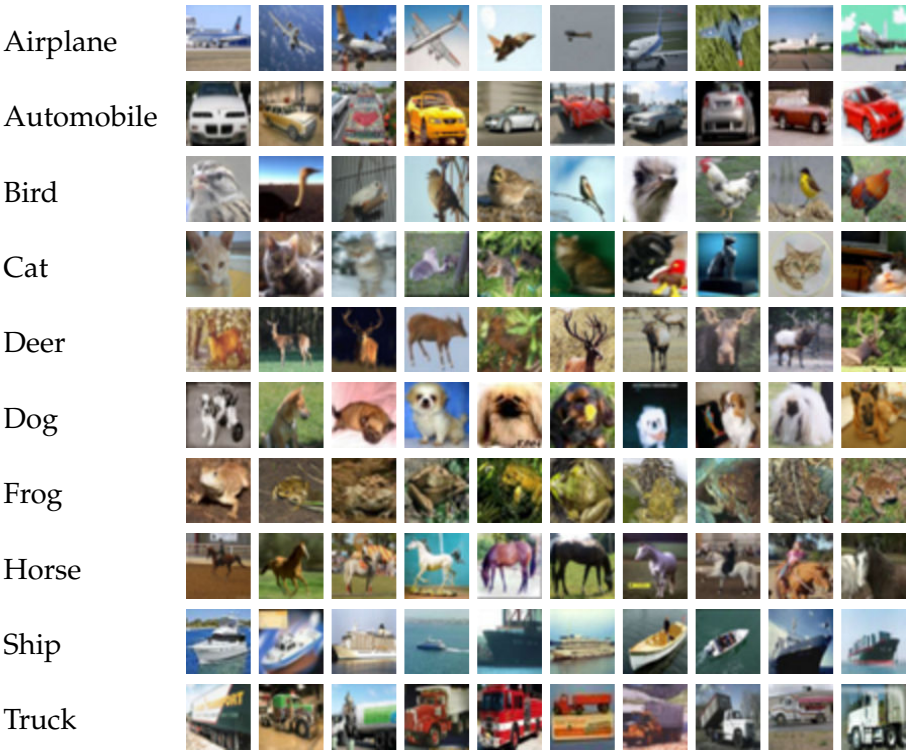


Figure 56 – Examples of images representing the ten classes from the CIFAR-10 dataset.

On this dataset, when using the information that signals are images, numerous solutions exist that reach high correct classification rates (96.53% for [Gra14], or more recently, 97.28% for [DPQ17]). However, state of the art methods that do not use this information reach a correct classification rate of 72.7% at most [CCM16]. Methods that do not use convolutions can reach 78% accuracy when adding deformations on the images, but reach 70% only when they do not consider any prior on the signals [LMK15].

In our approach, we do not use the information that signals are images. For this reason, we aim to show that inferring the underlying topology, and using translations to create an adapted convolutional neural network can improve classification of signals without using prior information on their nature.

5.2.1 Graph inference from data

Since we do not use the assumption that signals are images, a first step in our method is to find the structure on which the signals evolve. We choose here to consider four graphs, inferred with various graph inference techniques.

Direct use of the sample covariance matrix

The sample covariance matrix gives interesting information on the signal entries that tend to be similar. When considering the task of finding *objects* in signals, it is in many cases safe to assume that such elements to find are spread on multiple vertices. As an example, an object in an image will be spread across multiple pixels of the image.

We proceed by setting the four highest off-diagonal entries per row of the sample covariance matrix of the training set images to 1, and dropping the others. The matrix is then symmetrized so that the obtained graph is undirected [2]. This binarization method favors the creation of a graph with a minimal degree per vertex. In the remaining of this chapter, this graph will be denoted by $\mathcal{G}_{\widetilde{\mathbf{\Sigma}}}$.

Using the method of Kalofolias

Since the method of Kalofolias [Kal16] introduced in Section 3.1.3 promotes smoothness of the signals on the inferred graph, it should provide a graph in which vertices carrying smooth patterns are linked together. Therefore, for the same reasons as above, it may provide a graph on which a localized kernel covers an object to find in the signals.

In the experiments we perform in this chapter, we consider a graph inferred using this method, obtained for values $\alpha = \beta = 10^{-3}$. Then, we keep the four highest entries per row as in Section 5.2.1 and symmetrize the matrix. This graph will be denoted by $\mathcal{G}_{\mathrm{K}}$.

Additionally, we also consider the regularized version of this matrix for stationary signals as introduced in Section 3.4. It is obtained by selecting the closest point from the polytope to the matrix inferred using the method by Kalofolias (before binarization). The same vertex selection process as for other methods is then performed to obtain a binary matrix. We denote this latter matrix by $\mathcal{G}^*_{\mathrm{K}}$.

Using our *Sparse* method

Finally, we also evaluate one of our graph inference methods on this dataset. We first infer a graph using the eigenvectors of the sample covariance matrix of the images from the training set, and perform the same selection of entries as for the two other inference methods to be able to find translations on sparse, unweighted graphs. The graph inferred using the *Sparse* method in (110) will be denoted by $\mathcal{G}_{\mathrm{sparse}}$.

Interestingly, all four methods manage to find graphs that are highly similar to a grid graph. As we have observed in Figure 49 that translations on slightly deformed grids were still very close to Euclidean translations, we expect the convolutional neural network we define using these graphs to perform quite well on the CIFAR-10 dataset.

2. Note that our algorithm to search for translations does not require the graph to be undirected. Still, we want to be able to slide a kernel to every possible vertex of the graph, which may not be possible if considering a directed graph.

5.2.2 Identification of translations on the graph

We have shown in Proposition 4.3.1 that identification of translations on an arbitrary graph is an NP-complete problem. In the current experiment however, we are trying to translate some localized kernels, that we define as follows:

Definition 80: Convolutional kernel on a graph

A ***convolutional kernel*** of parameter K on a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, centered on a vertex $v_1 \in \mathcal{V}$, is the subset of vertices $\mathcal{V}^- \subset \mathcal{V}$ such that:

$$\forall v_2 \in \mathcal{V} : v_2 \in \mathcal{V}^- \Leftrightarrow d_{\text{geo}}(v_1, v_2) \leq K \ . \tag{151}$$

For this reason, we consider induced subgraphs of limited orders (large enough to include the convolutional kernel), centered on the vertices of the graph, on which we compute translations. This gives us sufficient information to translate a kernel as, for a vertex v, both the kernel centered on v and its translations do have null entries on any vertex that does not belong to the induced subgraph centered on v.

5.2.3 Definition of the convolution matrix of a convolutional neural network

To define the edges between two layers, we perform as for classical convolutional neural networks, by instantiating a convolutional kernel on the graph. We choose to initially localize this kernel to the most central vertex of the graph, considering the ***proximity centrality*** of vertices [Bav50; Sab66]. A vertex v_1 is said to be the most central according to this measure if it maximizes the following quantity:

$$\mathrm{C}(v_1) = \frac{1}{\sum\limits_{v_2 \in \mathcal{V}} d_{\text{geo}}(v_1, v_2)} \ . \tag{152}$$

Then, we translate the kernel to all other vertices upon saturation. If a kernel is centered on a vertex v, then we apply all possible translations previously found on the induced subgraph centered on v. This gives us new possible localizations of the kernel, with centers on the neighbors of v. We then apply this procedure upon convergence, keeping the kernel minimizing the loss in case of multiple kernels centered on a single vertex.

Finally, once the kernel has been translated to every possible vertex, we build a convolutional layer in our network by adding an edge between neurons associated with all vertices under the kernel range and the neuron associated with the center of the kernel, as presented in Figure 54. Implementation of the convolution operator using our translations is made as in Vialatte *et al.* [VGC17].

Using this procedure with convolutional kernels of parameter $K = 3$, we obtain the Python/Keras [Cho15] code given in Appendix A, corresponding to the network depicted in Figure 57:

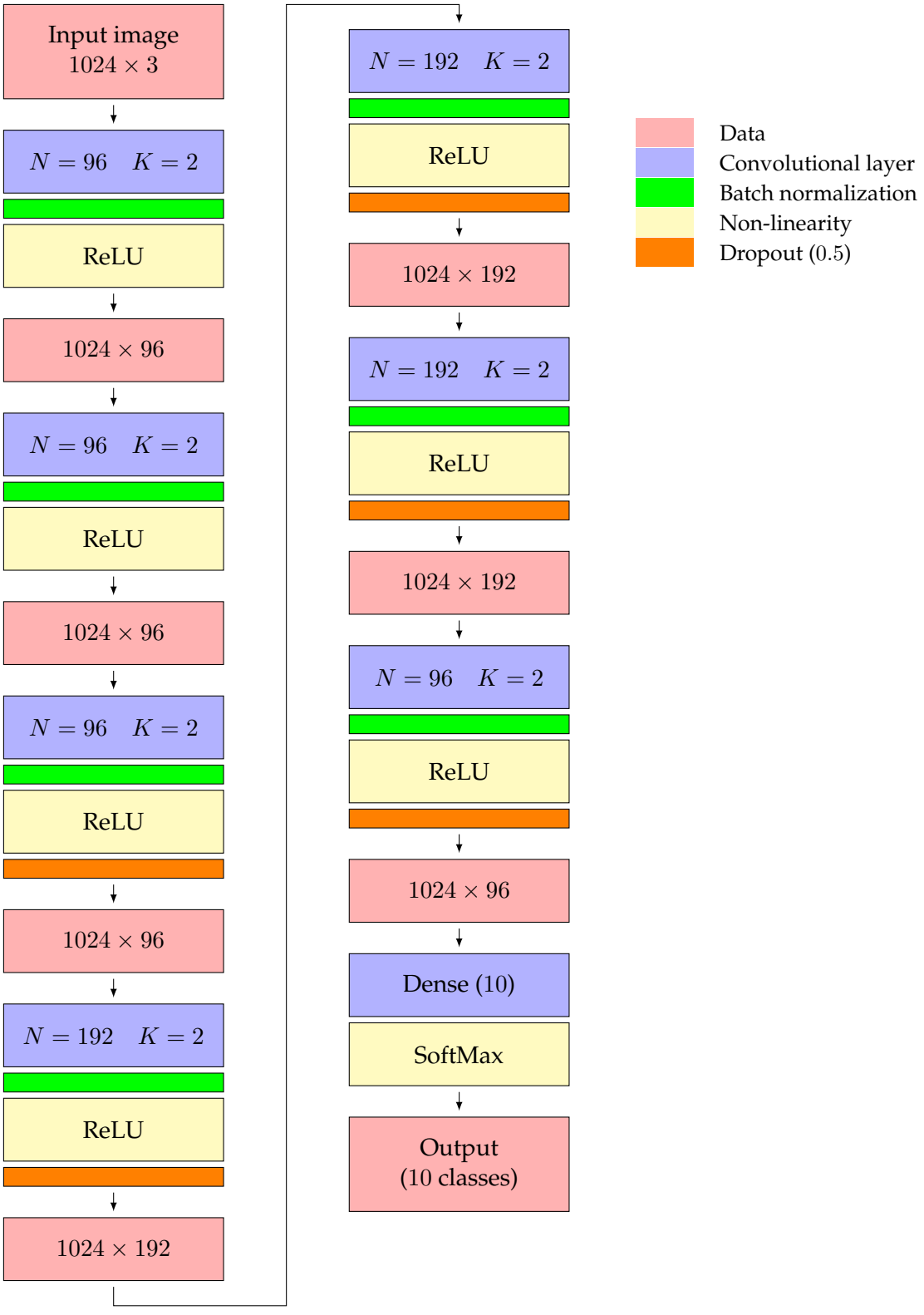


Figure 57 – Graphical representation of the convolutional neural network implemented by the Python/Keras [Cho15] code given in Appendix A. The network consists of seven convolutional layers using our translations — N denotes the number of filters, and K the number of hops in Definition 80 defining the kernel — followed by one output dense — fully connected — layer. Dropout and batch normalization are added in order to prevent ***overfitting***, *i.e.*, to favor generalization to unseen examples.

5.2.4 Classification results

Results on CIFAR-10

As indicated earlier in this chapter, state of the art methods that do not use the information that signals are images reach a correct classification rate of 72.7% [CCM16]. Using the network in Appendix A, we obtain the correct classification rates in Table 3, depending on the graph inference method used:

$\mathcal{G}_g$	$\mathcal{G}_{\widetilde{\mathbf{\Sigma}}}$	$\mathcal{G}_{\mathrm{K}}$	$\mathcal{G}_{\mathrm{K}}^*$	$\mathcal{G}_{\mathrm{sparse}}$
85.98% (193)	85.03% (246)	85.33% (195)	85.81% (277)	77.54% (169)

Table 3 – Correct classification rates obtained with our method, as a function of the graph inference technique used. Values into parentheses indicate the number of the epoch at which the best value was found. The classification result obtained for a grid graph $\mathcal{G}_g$ of dimensions $\begin{bmatrix} 32 \\ 32 \end{bmatrix}$ indicates an upper bound on the results we expect to obtain with our architecture when using inferred graphs.

For the convolutional neural network architecture used, it is interesting to notice that using the graphs that encode some smoothness assumptions — *i.e.*, $\mathcal{G}_{\widetilde{\mathbf{\Sigma}}}$, $\mathcal{G}_{\mathrm{K}}$ and $\mathcal{G}_{\mathrm{K}}^*$ — perform best. This is an expected result, as elements to classify are generally smooth across adjacent pixels in images. Additionally, it appears that regularization of the method of Kalofolias — to have it match a stationarity assumption on the signals — improves the results, which makes it an interesting assumption to consider in addition to smoothness.

The *Sparse* method however does not make any particular prior on the smoothness of the signals, and performs slightly worse than the two other graphs. Still, the correct classification rate remains higher than the results of Cheng *et al.* in [CCM16]. When looking in details at the kernel after translation, it appears that it reduces more in size than for the other graphs as it is translated, due to a larger loss of the translations found. In details, Table 4 gives the mean number of vertices covered by the kernel (per kernel center) for all considered methods:

	$\mathcal{G}_g$	$\mathcal{G}_{\widetilde{\mathbf{\Sigma}}}$	$\mathcal{G}_{\mathrm{K}}$	$\mathcal{G}_{\mathrm{K}}^*$	$\mathcal{G}_{\mathrm{sparse}}$
Initial kernel size	13 (495)	13 (527)	13 (528)	13 (495)	13 (530)
Mean kernel size	12.433	11.523	11.27	12.194	8.137

Table 4 – Mean number of vertices covered by a 2-hops convolutional kernel, after translation to all possible vertices. The initial kernel size indicates the number of vertices covered by the kernel at initialization, *i.e.*, when created on the most central vertex, given into parentheses.

These results illustrate that inference of a graph from data — even if not the best graph — allows the definition of a convolutional neural network that improves classification of this data. Also, it highlights that the prior used to infer the graph has a strong importance in the process. As a matter of fact, if the translations found on the graph necessarily have a large loss, it has some impact on the performance, as the convolutional kernel cannot be efficiently translated to all vertices.

Still, in all cases, our method manages to perform better than state of the art methods that do not use the information that signals are images, even with a simple convolutional neural network architecture found *by hand*. While the gain in performance is already quite high ($+13.11$ points when using $\mathcal{G}_{\mathrm{K}}^{*}$), we believe that it could be increased with a more complex architecture found thanks to a grid search over all parameters. In particular, when considering a perfect grid graph to define the connections of a convolutional neural network, we obtain a classification rate of 85.98% with our network architecture, which is nearly achieved using the graph $\mathcal{G}_{\mathrm{K}}^{*}$. As more complex architectures can reach 97.28% correct classification when using the information that signals are images, we believe that using an inferred graph with such networks could reach slightly lower performance, while not using the information that signals are images.

Behavior of the method on scrambled CIFAR-10

Scrambled CIFAR-10 is a transformed version of the CIFAR-10 dataset. In this dataset, every pixel is associated with a new location, chosen randomly. Note that all images are shuffled using the same permutation of pixels. Therefore, objects to recognize are no longer localized in the image. On such scrambled images, the classical convolutional neural networks do not work anymore, as the sliding windows now cover pixels that can belong to various elements of the images.

As an example, we consider in Figure 58 a random permutation of the pixels of all images in the CIFAR-10 dataset. Replacing the convolutional layers in Figure 57 with regular 2D convolutional layers using kernels of size 3×3, we obtain a correct classification rate of 56.74% (obtained at epoch 187):

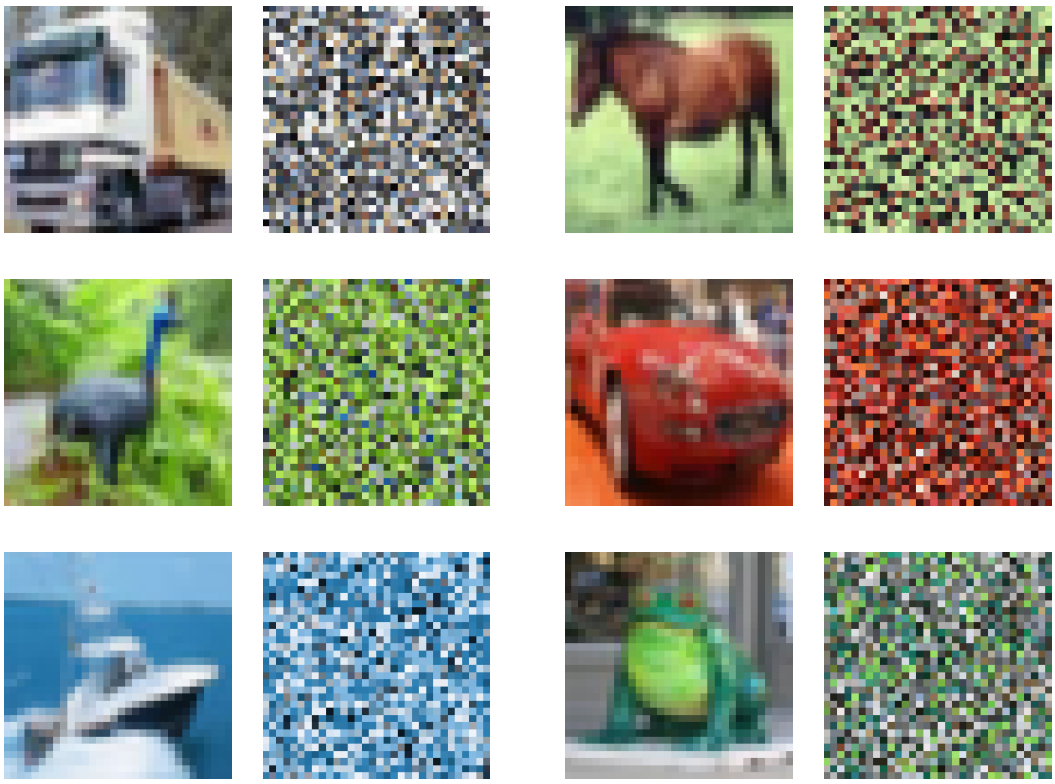


Figure 58 – Some images from the CIFAR-10 training set, given in their normal form (left) and in their scrambled version using a random permutation matrix (right). The same permutation is applied to all images of the training and test sets.

An interesting thing about our approach is that the edges in the inferred graph depend on the graph inference method, and not on the localization of the pixels. If all images are shuffled identically, then the sample covariance matrix remains exactly the same except for a relabeling of vertices.

Therefore, the various graphs computed in Section 5.2.1 remain exactly the same as those obtained when studying the regular CIFAR-10 dataset. Since the graphs are the same, so are the translations found, thus the convolutional layers. As a conclusion, our method performs exactly the same on the scrambled version of the dataset as it does on the regular one, provided that the graph inference method makes use of the covariance matrix or another matrix that does not depend on the labeling.

5.2.5 Perspectives: the interest of identifying circulant matrices

In some of the experiments we performed on other type of data — that are not depicted here — we have observed that strong irregularities in the inferred graph can prevent the translations from being computed. As a matter of fact, neighboring vertices that do not share any other neighborhood cause the kernel to reduce to its center only once translated, thus causing a high loss in the signal entries. Due to this high loss, translation of a convolutional kernel to all vertices of the graph is not possible without losing most of the kernel energy. This leads to the impossibility to define the convolutional layers of the neural network.

We believe that a possible solution to this problem lies in the properties of circulant matrices. As a matter of fact, if a matrix is circulant, it corresponds to a graph that is highly regular. A simple example is to consider the adjacency matrices of the ring graph and of the two-dimensional torus graph, given in Figure 59:

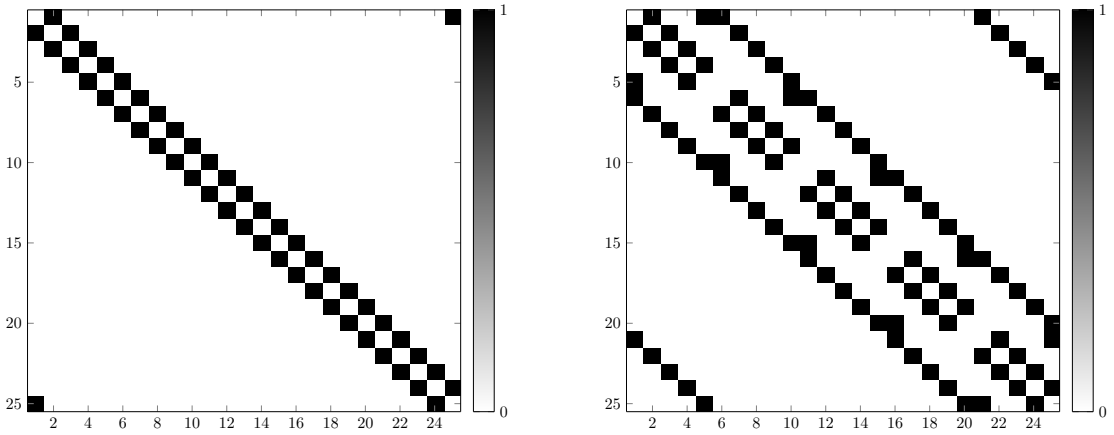


Figure 59 – Adjacency matrices of a ring graph of $N = 25$ vertices (left), and of a torus graph of dimensions vector $\mathbf{d} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$ (right). The first matrix is circulant, and the second one is circulant by blocks.

Inferring translations on circulant matrices is a particular case of inferring translations on graphs, as one can simply follow the labeling order to obtain lossless translations. Therefore, a possible idea to apply our method on irregular graphs would be to proceed in three steps:

1. Given the binary adjacency matrix $\mathbf{A}$ of a graph, find a binary circulant matrix $\mathbf{C}$ such that $\|\mathbf{A} - \mathbf{C}\|_{0,1}$ is minimized;

2. Infer translations on $\mathbf{C}$;
3. Either directly use these translations to define a convolutional neural network, or use an adaptation of these translations to account for the existing edges in $\mathbf{A}$.

The main idea behind this possible solution is to approximate the irregular graph with a close regular topology to simplify the problem. However, finding this circulant matrix appears to be a very difficult problem, due to two main limitations:

1. Let us consider the adjacency matrix of the ring graph in Figure 59. This matrix is circulant, due to a correct labeling of the various vertices. However, for a different labeling, it is no longer circulant, as the i^{th} is no longer a shift of the $i-1^{\text{th}}$ one. Still, we know there exists a permutation of the labels such that the matrix is circulant, thus solving our problem. However, there is no obvious strategy to find this particular permutation;
2. Another problem is that, in the cases we want to consider, the adjacency matrix is not circulant. Therefore, for a chosen labeling, we need to be able to compute the closest circulant matrix, in the sense of edge addition or subtraction.

At the moment, it is not clear for us if these two problems should be considered sequentially, of simultaneously. Additionally, we may also be interested in finding matrices that are circulant by blocks as in Figure 59 (right), which complicates the problem even more. Still, this is a very promising direction for future work, as it would provide new methods to infer translations on irregular graphs.

We believe this problem to have connections with (and applications to) other domains, such as the ***cyclic bandwidth sum*** problem, which consists in finding a labeling of vertices such that the total difference between labels of adjacent vertices is minimized [Jia01]. Interestingly, circulant matrices appear to have a low such sum, which makes us believe we can make profit of the various heuristics used in this field.

5.3 Summary of the chapter

In this chapter, we have explored how combination of the previous two chapters could help in machine learning tasks such as classification. We have proposed a method that, for a given training set of signals, performs as follows:

1. Infer a graph from the signals;
2. Infer translations on this graph;
3. Using these translations, shift a convolutional kernel — initialized at the center of the graph — to all possible vertices;
4. Define a convolutional neural network accordingly;
5. Train the convolutional neural network using the training set.

Our method has proven to be very effective on images, as inference methods could retrieve a graph that is very close to a grid graph. Translations on the inferred graph are also very close to Euclidean translations, which allows the creation of a convolutional neural network that resembles a network one could build when using the information that signals are images. Using our method, we have achieved a correct classification of 85.81% on CIFAR-10, where state of the art methods reached 72.7%. Additionally, since our approach infers a graph from data, it performs as well on scrambled datasets as on their regular versions, contrary to standard convolutional neural networks.

Most importantly, our approach is a real generalization of convolutional neural networks to irregular domains. While most existing approaches use a convolution operator that performs in the spectral domain and thus does not preserve the shape of the convolutional kernel, our approach work in the graph domain only. Once translations are inferred, then classical convolution can be performed again, making convolutional neural networks on images a particular case of our model.

Our work has a wide range of possible improvement, among which the following:

- The architecture we considered in Figure 57 — with code given in Appendix A — was found *by hand*. A grid search on the various parameters would probably improve the results consequently;
- We only considered one kernel size. Numerous convolutional neural networks vary the size of the convolutional kernel used at each layer. This is another parameter that should be taken into consideration in the grid search;
- There are no max-pooling or stride layers. Again, these operators appear to highly improve the classification task in classical convolutional neural networks. Finding equivalents to these operators is one of the main directions of our future work;
- We have observed that identification of translations on highly irregular topologies is a problem. We believe the approach detailed in Section 5.2.5 to be a promising direction to tackle this problem. This would allow the application of our methodology to more complex datasets.

Chapter 6

Conclusions

Contents

6.1 Summary of the manuscript

The work presented in this Ph.D. manuscript focuses on some aspects of graph signal processing, a subfield of signal processing that emerged a few years ago from the will to generalize Fourier analysis to signals evolving on irregular domains. In this context, numerous tools were defined to process such signals, based on an analogy between the Fourier modes in classical signal processing and the eigenvectors of the Laplacian matrix of the ring graph. In details, these eigenvectors define an adapted spectral basis for signals evolving on the associated graph, and projection of signals on these eigenvectors gives a spectral representation of them. The notion of frequencies from classical signal processing can be found again in these spectral signals, as the entries associated with the lowest eigenvalues encode regularity of the signals on the graph, while the highest correspond to strong variations. This observation allows definition of filters on graphs, analogously to those issued from classical signal processing, as well as numerous other operators on signals.

6.1.1 From signals to graphs...

Knowledge of the graph on which the observed signals evolve is a cornerstone of graph signal processing. While one can intuitively think of a grid graph to model the support of images, or a ring graph to model periodical time, in most situation however such information is not available. Examples include electroencephalographic sensors placed on the head of a patient, or seismic sensors thrown on a field, for which the notion of vertex is intuitive, but edges are not easy to choose. Under such settings, graph signal processing tools cannot be applied anymore, and one needs to infer a graph to enable their use.

In some situations, information on the vertices — such that their geographical location — can be used to define a graph, for example by linking vertices under a certain radius with an edge. In some other cases however, this is not possible, or it may not reflect the real topology on which data evolves. A possible solution is therefore to infer a graph from the observed data. This problem is however ill-posed, as there is a huge number of potentially admissible graphs matching the given signals. One thus needs to add some priors on the graph to infer (such as simplicity, sparsity, binarity of the weights...), or on the signals (including stationarity, observation after diffusion of Dirac deltas on the graph, smoothness on the graph...).

6.1.2 ...to translations...

Once a graph is inferred from data, it can then be used to process the signals. In particular, a very common task on signals is to classify them. Applications of classification are countless, and some well-known examples include face recognition, heart attack detection, as well as more common tools such as Google Images.

In this context, knowledge of the underlying topology can help a lot in the classification task. In the simple case of images, the underlying grid indicates that two adjacent pixels are generally similar, and local kernels can be translated on the image to detect some patterns in the signal. In the more general case when the graph is inferred from data, translation of such kernels is not an easy task, since notions such as *to the left*, or *forward in time* do not exist due to the absence of an underlying metric space.

Inferring translations on such graphs that are similar to translations on metric spaces is a possible method to allow this kernel-based approach again. As a matter of fact, Euclidean translations of signals on a graph can be understood as shifting the observer's point of view to a different place: when translated, the signal should globally *look the same* as before. This implies that, when a signal entry is translated from a vertex to another, the signal entries carried by neighboring vertices should also be translated to neighbors of the target vertex.

6.1.3 . . . to signals classification

Objects to identify in signals — for instance, a cat in a picture, or a particular activity in the brain — are generally dispatched on multiple vertices, that we expect to be adjacent — or at a short distance considering the number of edges — from one another. This is the reason why localized kernels in images perform well in classification tasks, and this is the approach taken by convolutional neural networks. When considering images, the connections in such networks are created by translating a localized kernel on the image. This encodes some locality properties in the network structure, which has proven to improve classification of objects in images.

In this manuscript, we have extended this principle to signals defined on irregular graphs. Once a graph is inferred from data, and once translations are found on this graph, we use these translations to shift a local kernel, in order to define the connections between the layers of a convolutional neural network.

6.2 Contributions

This section summarizes the contributions of the main articles from which this manuscript is built. Additional contributions to the field of graph signal processing, that are not directly related to the main line of this manuscript, are not detailed. Still, information on the uncertainty principle [Pas+15b; Pas+16] can be found in Section 2.3.3, and application of graph inference to neural signals classification through dimensionality reduction can be found in [Mén+17].

6.2.1 Graph inference from signals

Graph inference from signals is the main concern of Chapter 3. We have studied the particular case of stationary signals, and have proposed methods to infer a graph shift operator adapted to such signals, which is equivalent to finding the edges of the graph.

In particular, we have shown that any graph shift operator adapted to stationary signals should have the same eigenvectors as the covariance matrix of the signals. These eigenvectors define a particular convex set of eigenvalues, from which every element is a possible solution. The problem then becomes how to select a point from this set, using additional criteria.

We have first studied a simplified problem, where every signal is diffused once [Pas+15a], for which we have shown that we could find the eigenvalues of the ground truth matrix. Then, we have extended this work to the more general case of signals diffused by any graph filter [Pas+17a], and have proposed approaches enforcing simplicity or sparsity of the solution. Finally, we have introduced a third approach allowing the

correction of the solution provided by a given inference method, to have it match the stationarity assumption on signals.

Experiments were performed on synthetic data, as well as on real temperature measurements in Brittany. The graphs inferred using our strategies have appeared to feature the properties corresponding to the selection criteria used Additionally, correction of other methods also provided interesting results, with possible applications to graph selection from a set of candidates.

6.2.2 Translations identification on graphs

Identification of translations on an arbitrary graph has been presented in details in Chapter 4. We have introduced numerous functions on graphs, that have eventually led to translations on graphs. Such translations have the interesting property to retain the neighborhoods of all vertices when translating them, thus causing a signal to *look similar* once it is shifted on the graph.

In addition to introducing such translations in [Gre+16], we have shown in [Pas+17b] that their identification on an arbitrary graph is an NP-complete problem. To cope with this complexity issue, we have relaxed the problem to allow small deformations of the translated signal, and have proposed an approximate algorithm based on an optimization problem.

This algorithm has proven to be able to find the expected translations on grid and torus graphs, and has allowed the identification of pseudo-translations on irregular graphs, such as random graphs following a Watts-Strogatz model. Additionally, our experiments have shown that, under small deformations of a grid graph, the pseudo-translations found were very close to Euclidean translations on the grid.

6.2.3 Classification of signals using graph signal processing and convolutional neural networks

In Chapter 4, we have proposed to use graph inference and translations identification to help classifying signals. We have designed a method based on graph signal processing to create a convolutional neural network that is adapted to the signals to classify. In few words, the method consists in the following steps:

1. From a set of signals to classify, infer a graph;
2. Find translations on this graph;
3. Use these translations to shift a local kernel on the graph, thus defining a connectivity tensor between the layers of a convolutional neural network;
4. Train this network with the initial signals.

Contrary to simple convolutional neural networks, our method does not use the coordinates of the vertices — for example a 3×3 sliding window on an image — to define the connections. Kernels are defined by considering a central vertex and a small portion of its neighborhood. Therefore, if the graph is a grid and signals to classify are images, we obtain results that are very close to those of a classical convolutional neural network.

When considering a permutation of the pixels of the images to classify, sliding windows from which convolutional neural networks are built cannot detect objects in images anymore, since they are no longer localized in the image. Our method however can find

the links between the initially adjacent pixels thanks to the graph inference method, and the retrieved graph is simply a relabeling of the one obtained from unshuffled images. Therefore, the network we build using our techniques performs as well as for regular images. More generally, since our method does not use prior information on the topology on which signals evolve, it allows the detection of patterns in signals evolving on complex topologies.

As a conclusion, our method is a generalization of convolutional neural network based on graph signal processing. Without any localization information on the vertices, it is able to detect *objects* in the input signals, whatever their location in the graph. This has numerous applications, including improvement of classification techniques, as well as fault detection in a network, or analysis of brain/seismic activity.

6.3 Perspectives

As indicated in the respective conclusions of the chapters of this manuscript, each part of our work can be extended in numerous ways.

Considering graph inference from signals, numerous other methods could be introduced to select a vector of eigenvalues from the polytope, enforcing properties such as binarity of the retrieved graph. Also, complexity of the definition of the polytope of acceptable solutions can be an issue, as the number of constraints delimiting it is of the order of N^2, among which some constraints are not informative, and could be removed if a method to identify were to be found. Then, the number of signals required to obtain a sufficiently correct approximate of the eigenvectors of the covariance matrix is quite high when using the sample covariance matrix as an estimator. Other covariance estimators could be studied to obtain a faster convergence of these eigenvectors. Finally, our experiments on graph selection from a set of candidates could be complemented by considering noisy versions of the ground truth matrix.

Our work on identification of translations also opens some new perspectives. NP-completeness of the problem of identifying minimal translations forced us to propose a relaxed version of the problem, through the resolution of an optimization problem. Alternative solutions could also be explored, for example with greedy approaches. Additionally, we have chosen in this relaxation to sacrifice the SNP property, while still enforcing pseudo-translations to be EC. Accepting a partial sacrifice of this property as well could lead to the definition of interesting pseudo-translations, especially when considering graphs inferred with some noise.

Finally, we believe our work on convolutional neural networks to have many applications in signals processing and analysis. Still, state of the art convolutional neural networks use more complex mechanisms, such as max-pooling or stride layers. Defining such operators is a great direction for future work, as it would probably improve the classification results significantly. Also, improving the translations detection method for highly irregular graphs is a challenging problem, that would allow the consideration of new problems with potential important applications.

Appendix A

Implementation of our convolutional neural networks

The following Python/Keras code [Cho15] is an implementation of the convolutional neural network using our translations. For space reasons, the files containing the translations found on the various graphs are not provided, but are available upon request:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
##########################################################################################################################
################################################## INITIALIZATION ########################################################
##########################################################################################################################

# Imports
import sys
import ast
import numpy
import tensorflow
from ternary_layer import *
from keras.models import Sequential
from keras.datasets import cifar10
from keras.layers import Reshape, Dense, Dropout, Flatten, Activation
from keras.layers.normalization import BatchNormalization
from keras.optimizers import Adam
from keras.utils.np_utils import to_categorical

# Arguments
if len(sys.argv) != 2 :
    print("Usage: " + sys.argv[0] + " <translationsFile>")
    quit()
translationsFile = sys.argv[1]

# Processing the translations file
with open(translationsFile, "r") as fileObject :
    fileContents = ast.literal_eval(fileObject.read())
nbVertices = len(fileContents)
maxFilterSize = max([value for value in max([list(cnnFilter.values()) for cnnFilter in fileContents])])

# Custom layer using these translations
S = numpy.zeros((maxFilterSize, nbVertices, nbVertices))
i = 0
for cnnFilter in fileContents:
    for vertex in cnnFilter :
        S[cnnFilter[vertex] - 1, vertex, i] = 1
    i += 1
initializer = tensorflow.constant_initializer(S)

# We load the dataset
(X_train, Y_train), (X_test, Y_test) = cifar10.load_data()
imgWidth = X_train.shape[1]
imgHeight = X_train.shape[2]
imgChannels = X_train.shape[3]
nbClasses = len(set([label[0] for label in Y_train]))

##########################################################################################################################
################################################## CNN DESCRIPTION #######################################################
##########################################################################################################################

# Input of the model
model = Sequential()
model.add(Reshape((nbVertices, imgChannels), input_shape=(imgWidth, imgHeight, imgChannels)))

# Custom layer 1
nbFilters = 96
model.add(TernaryLayer(maxFilterSize, nbVertices, nbFilters, scheme_initializer=initializer, train_scheme=False))
model.add(BatchNormalization())
```

```
model.add(Activation("relu"))

# Custom layer 2
nbFilters = 96
model.add(TernaryLayer(maxFilterSize, nbVertices, nbFilters, scheme_initializer=initializer, train_scheme=False))
model.add(BatchNormalization())
model.add(Activation("relu"))

# Custom layer 3
nbFilters = 96
model.add(TernaryLayer(maxFilterSize, nbVertices, nbFilters, scheme_initializer=initializer, train_scheme=False))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(Dropout(0.5))

# Custom layer 4
nbFilters = 192
model.add(TernaryLayer(maxFilterSize, nbVertices, nbFilters, scheme_initializer=initializer, train_scheme=False))
model.add(BatchNormalization())
model.add(Activation("relu"))

# Custom layer 5
nbFilters = 192
model.add(TernaryLayer(maxFilterSize, nbVertices, nbFilters, scheme_initializer=initializer, train_scheme=False))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(Dropout(0.5))

# Custom layer 6
nbFilters = 192
model.add(TernaryLayer(maxFilterSize, nbVertices, nbFilters, scheme_initializer=initializer, train_scheme=False))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(Dropout(0.5))

# Custom layer 7
nbFilters = 96
model.add(TernaryLayer(maxFilterSize, nbVertices, nbFilters, scheme_initializer=initializer, train_scheme=False))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(Dropout(0.5))

# Dense output layer
model.add(Flatten())
model.add(Dense(nbClasses, activation="softmax"))

#########################################################################################################################
#################################################### CNN TRAINING #######################################################
#########################################################################################################################

# First 150 epochs with learning rate of 0.001
model.compile(loss="categorical_crossentropy", optimizer=Adam(lr=0.001, decay=1e-6), metrics=["accuracy"])
model.fit(X_train / 255.0, to_categorical(Y_train), batch_size=32, shuffle=True, epochs=150, validation_data=(X_test
    ↪ / 255.0, to_categorical(Y_test)), verbose=1)

# Next 75 epochs with learning rate of 0.0001
model.compile(loss="categorical_crossentropy", optimizer=Adam(lr=0.0001, decay=1e-6), metrics=["accuracy"])
model.fit(X_train / 255.0, to_categorical(Y_train), batch_size=32, shuffle=True, epochs=75, validation_data=(X_test
    ↪ / 255.0, to_categorical(Y_test)), verbose=1)

# Final 75 epochs with learning rate of 0.00001
model.compile(loss="categorical_crossentropy", optimizer=Adam(lr=0.00001, decay=1e-6), metrics=["accuracy"])
model.fit(X_train / 255.0, to_categorical(Y_train), batch_size=32, shuffle=True, epochs=75, validation_data=(X_test
    ↪ / 255.0, to_categorical(Y_test)), verbose=1)
```

Additionally, the code below depicts the important part of the `TernaryLayer.py` file, implementing the propagation formula:

```
class TernaryLayer (Layer) :

    ...

    # Call method implementing the propagation formula
    def call (self, x, mask=None) :

        # x (b,n,p) (dot) S (w,n,m) = xS (b,p,w,m)
        xS = tf.tensordot(x, self.scheme, [[1],[1]])

        # xS (b,p,w,m) (dot) W (w,p,q) = y (b,m,q)
        y = tf.tensordot(xS, self.kernel, [[1,2],[1,0]])
        y.set_shape([x.shape[0], self.m, self.q])

        # If there is bias
        if self.use_neuron_bias:
            y += tf.reshape(self.neuron_bias, (1, 1, self.q))

        # Return with activation
        return self.neuron_activation(y)
```

Bibliography

[AK98] Edoardo Amaldi and Viggo Kann. "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems". In: *Theoretical Computer Science* 209.1-2 (1998), pp. 237–260.

[AL07] Manuel Appert and Chapelon Laurent. "Measuring urban road network vulnerability using graph theory: the case of Montpellier's road network". In: *La mise en carte des risques naturels* (2007), 89p.

[AL12] Ameya Agaskar and Yue M Lu. "Uncertainty principles for signals defined on graphs: Bounds and characterizations". In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE. 2012, pp. 3493–3496.

[AL13] Ameya Agaskar and Yue M Lu. "A spectral graph uncertainty principle". In: *IEEE Transactions on Information Theory* 59.7 (2013), pp. 4338–4356.

[AM12] James A Albano and David W Messinger. "Euclidean commute time distance embedding and its application to spectral anomaly detection". In: *Proc. SPIE*. Vol. 8390. 2012, 83902G.

[And63] T. W. Anderson. "Asymptotic Theory for Principal Component Analysis". In: *Ann. Math. Statist.* 34.1 (Mar. 1963), pp. 122–148. DOI: 10.1214/aoms/1177704248. URL: http://dx.doi.org/10.1214/aoms/1177704248.

[AT15] James Atwood and Don Towsley. "Search-Convolutional Neural Networks". In: *CoRR* abs/1511.02136 (2015). URL: http://arxiv.org/abs/1511.02136.

[Bav50] Alex Bavelas. "Communication patterns in task-oriented groups". In: *The Journal of the Acoustical Society of America* 22.6 (1950), pp. 725–730.

[Ber05] Dennis S Bernstein. *Matrix mathematics: Theory, facts, and formulas with application to linear systems theory*. Vol. 41. Princeton University Press Princeton, 2005.

[Bro+17] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[Bru+13] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral networks and locally connected networks on graphs". In: *arXiv preprint arXiv:1312.6203* (2013).

[CCM16] Xiuyuan Cheng, Xu Chen, and Stéphane Mallat. "Deep Haar scattering networks". In: *Information and Inference: A Journal of the IMA* 5.2 (2016), pp. 105–133.

[CDS01] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. "Atomic decomposition by basis pursuit". In: *SIAM review* 43.1 (2001), pp. 129–159.

[Che+17] Sundeep Prabhakar Chepuri, Sijia Liu, Geert Leus, and Alfred O Hero. "Learning sparse graphs under smoothness prior". In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE. 2017, pp. 6508–6512.

[Cho15] François Chollet. *Keras*. `https://github.com/fchollet/keras`. 2015.

[Chu97] Fan RK Chung. *Spectral graph theory*. 92. American Mathematical Soc., 1997.

[CL11] Tony Cai and Weidong Liu. "Adaptive thresholding for sparse covariance matrix estimation". In: *Journal of the American Statistical Association* 106.494 (2011), pp. 672–684.

[Cor09] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[CZ12] T Tony Cai and Harrison H Zhou. "Optimal rates of convergence for sparse covariance matrix estimation". In: *The Annals of Statistics* (2012), pp. 2389–2420.

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3844–3852.

[Dem72] Arthur P Dempster. "Covariance selection". In: *Biometrics* (1972), pp. 157–175.

[DJ+10] Xue Ding, Tiefeng Jiang, et al. "Spectral distributions of adjacency and Laplacian matrices of random graphs". In: *The annals of applied probability* 20.6 (2010), pp. 2086–2117.

[Don+16] Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. "Learning laplacian matrix in smooth graph signal representations". In: *IEEE Transactions on Signal Processing* 64.23 (2016), pp. 6160–6173.

[DPQ17] Anuvabh Dutt, Denis Pellerin, and Georges Quénot. "Coupled Ensembles of Neural Networks". In: *arXiv preprint arXiv:1709.06053* (2017).

[Dra05] Steve Draper. *How many senses do humans have?* `http://www.psy.gla.ac.uk/~steve/best/senses.html`. Accessed: 2017-07-27. Department of Psychology, University of Glasgow, 2005.

[Duv+15] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. "Convolutional networks on graphs for learning molecular fingerprints". In: *Advances in neural information processing systems*. 2015, pp. 2224–2232.

[EPO16] Hilmi E Egilmez, Eduardo Pavez, and Antonio Ortega. "Graph learning from data under structural and laplacian constraints". In: *arXiv preprint arXiv:1611.05181* (2016).

[ER59] P Erdős and A Rényi. "On Random Graphs I". In: *Publ. Math. Debrecen* 6 (1959), pp. 290–297.

[Esc+01] Bernard Escudié, Claude Gazanhes, Henri Tachoire, and Vincenç Torra. *Des cordes aux ondelettes. L'analyse en temps et en fréquence avant et après Fourier. Un inverseur de l'équation de la chaleur de Fourier : le calorimètre à conduction*. Publications de l'Université de Provence, 2001.

[FHT08] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Sparse inverse covariance estimation with the graphical lasso". In: *Biostatistics* 9.3 (2008), pp. 432–441.

[Fis36] Ronald A Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of human genetics* 7.2 (1936), pp. 179–188.

[Fis38] Ronald A Fisher. "The statistical utilization of multiple measurements". In: *Annals of Human Genetics* 8.4 (1938), pp. 376–386.

[Fou22] Joseph Fourier. *Theorie analytique de la chaleur, par M. Fourier*. Chez Firmin Didot, père et fils, 1822.

[Gan98] Feliks Ruvimovich Gantmakher. *The theory of matrices*. Vol. 131. American Mathematical Soc., 1998.

[GBY08] Michael Grant, Stephen Boyd, and Yinyu Ye. *CVX: Matlab software for disciplined convex programming*. 2008.

[GGF15] Benjamin Girault, Paulo Gonçalves, and Éric Fleury. "Translation on graphs: an isometric shift operator". In: *IEEE Signal Processing Letters* 22.12 (2015), pp. 2416–2420.

[Gir+16] Benjamin Girault, Paulo Gonçalves, Shrikanth Narayanan, and Antonio Ortega. "Localization bounds for the graph translation". In: *CoRR* abs/1609.08820 (2016). URL: http://arxiv.org/abs/1609.08820.

[Gir15a] Benjamin Girault. "Signal processing on graphs-contributions to an emerging field". PhD thesis. Lyon, École normale supérieure, 2015.

[Gir15b] Benjamin Girault. "Stationary Graph Signals using an Isometric Graph Translation". In: *Eusipco*. Nice, France, Aug. 2015, pp. 1531–1535. URL: https://hal.inria.fr/hal-01155902.

[Gir39] M. A. Girshick. "On the Sampling Theory of Roots of Determinantal Equations". In: *Ann. Math. Statist.* 10.3 (Sept. 1939), pp. 203–224. DOI: 10.1214/aoms/1177732180. URL: http://dx.doi.org/10.1214/aoms/1177732180.

[GN03] Rémi Gribonval and Morten Nielsen. "Sparse representations in unions of bases". In: *IEEE transactions on Information theory* 49.12 (2003), pp. 3320–3325.

[GO15] Inc. Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2015. URL: \url{http://www.gurobi.com}.

[GP10] Leo J Grady and Jonathan Polimeni. *Discrete calculus: Applied analysis on graphs for computational science*. Springer Science & Business Media, 2010.

[Gra14] Benjamin Graham. "Fractional Max-Pooling". In: *CoRR* abs/1412.6071 (2014). URL: http://arxiv.org/abs/1412.6071.

[Gre+16] Nicolas Grelier, Bastien Pasdeloup, Jean-Charles Vialatte, and Vincent Gripon. "Neighborhood-Preserving Translations on Graphs". In: *GlobalSIP 2016 : IEEE Global Conference on Signal and Information Processing*. Arlington, United States, 2016, pp. 410–414.

[GZ15] A. Gavili and X.-P. Zhang. "On the Shift Operator, Graph Frequency and Optimal Filtering in Graph Signal Processing". In: *ArXiv e-prints* (Nov. 2015). arXiv: 1511.03512 [math.SP].

[HBL15] Mikael Henaff, Joan Bruna, and Yann LeCun. "Deep convolutional networks on graph-structured data". In: *arXiv preprint arXiv:1506.05163* (2015).

[HCQ17] Yotam Hechtlinger, Purvasha Chakravarti, and Jining Qin. "A Generalization of Convolutional Neural Networks to Graph-Structured Data". In: *arXiv preprint arXiv:1704.08165* (2017).

[Hei25] W. Heisenberg. "Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik". In: *Zeitschrift für Physik* 33 (1925), pp. 879–893.

[HSW83] D. Hanson, K. Seyffarth, and J. H. Weston. "Matchings, Derangements, Rencontres". In: *Mathematics Magazine* 56.4 (1983), pp. 224–229. ISSN: 0025570X, 19300980. URL: http://www.jstor.org/stable/2689812.

[Hua+09] Shuai Huang, Jing Li, Liang Sun, Jun Liu, Teresa Wu, Kewei Chen, Adam Fleisher, Eric Reiman, and Jieping Ye. "Learning brain connectivity of Alzheimer's disease from neuroimaging data". In: *Advances in Neural Information Processing Systems*. 2009, pp. 808–816.

[HVG11] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. "Wavelets on graphs via spectral graph theory". In: *Applied and Computational Harmonic Analysis* 30.2 (2011), pp. 129 –150. ISSN: 1063-5203. DOI: http://dx.doi.org/10.1016/j.acha.2010.04.005. URL: http://www.sciencedirect.com/science/article/pii/S1063520310000552.

[IM02] Mads Ipsen and Alexander S Mikhailov. "Evolutionary reconstruction of networks". In: *Physical Review E* 66.4 (2002), p. 046109.

[Jia01] Hao Jianxiu. "Cyclic bandwidth sum of graphs". In: *Applied Mathematics-A Journal of Chinese Universities* 16.2 (2001), pp. 115–121. ISSN: 1993-0445. DOI: 10.1007/s11766-001-0016-0. URL: https://doi.org/10.1007/s11766-001-0016-0.

[Kal16] Vassilis Kalofolias. "How to learn a graph from smooth signals". In: *Artificial Intelligence and Statistics*. 2016, pp. 920–929.

[KH09] Alex Krizhevsky and Geoffrey Hinton. "Learning multiple layers of features from tiny images". In: (2009).

[Kot07] Sotiris Kotsiantis. "Supervised Machine Learning: A Review of Classification Techniques". In: 31 (Oct. 2007).

[KR93] Douglas J Klein and Milan Randić. "Resistance distance". In: *Journal of mathematical chemistry* 12.1 (1993), pp. 81–95.

[Lac+99] Jean-Philippe Lachaux, Eugenio Rodriguez, Jacques Martinerie, Francisco J Varela, et al. "Measuring phase synchrony in brain signals". In: *Human brain mapping* 8.4 (1999), pp. 194–208.

[Law56] D. N. Lawley. "Tests of Significance for the Latent Roots of Covariance and Correlation Matrices". In: *Biometrika* 43.1/2 (1956), pp. 128–136. ISSN: 00063444. URL: http://www.jstor.org/stable/2333586.

[LB+95] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[Lin12] Georg Lindgren. *Stationary stochastic processes: theory and applications*. CRC Press, 2012.

[LJY17] Fei-Fei Li, Justin Johnson, and Serena Yeung. *CS231n Convolutional Neural Networks for Visual Recognition*. http://cs231n.github.io/. Accessed: 2017-09-19. Stanford Vision Lab, 2017.

[LMK15] Zhouhan Lin, Roland Memisevic, and Kishore Reddy Konda. "How far can we go without convolution: Improving fully-connected networks". In: *CoRR* abs/1511.02580 (2015). URL: http://arxiv.org/abs/1511.02580.

[LT10] Brenden Lake and Joshua Tenenbaum. "Discovering structure by learning sparse graphs". In: *Proceedings of the Cognitive Science Society*. Vol. 32. 32. 2010.

[Mag] *Magic Card Market – Serra Angel (Alpha)*. https://fr.magiccardmarket.eu/Products/Singles/Alpha/Serra+Angel. Accessed: 2017-07-19. 2017.

[Mar+16] Antonio G. Marques, Santiago Segarra, Geert Leus, and Alejandro Ribeiro. "Stationary Graph Processes and Spectral Estimation". In: *CoRR* abs/1603.04667 (2016). URL: http://arxiv.org/abs/1603.04667.

[MAT12] MATLAB. *version 7.14.0 (R2012a)*. Natick, Massachusetts: The MathWorks Inc., 2012.

[MH12] Rahul Mazumder and Trevor Hastie. "The graphical lasso: New insights and alternatives". In: *Electronic journal of statistics* 6 (2012), p. 2125.

[MM15] Jonathan Mei and José MF Moura. "Signal processing on graphs: Estimating the structure of a graph". In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 5495–5499.

[Mon+16] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M Bronstein. "Geometric deep learning on graphs and manifolds using mixture model CNNs". In: *arXiv preprint arXiv:1611.08402* (2016).

[Mon13] Pierre Rémond de Montmort. *Essay d'analyse sur les jeux de hazard*. chez Jacque Quillau, imprimeur-juré-libraire de l'Université, rue Galande, 1713.

[MW47] Henry B Mann and Donald R Whitney. "On a test of whether one of two random variables is stochastically larger than the other". In: *The annals of mathematical statistics* (1947), pp. 50–60.

[Mén+17] Mathilde Ménoret, Nicolas Farrugia, Bastien Pasdeloup, and Vincent Gripon. "Evaluating Graph Signal Processing for Neuroimaging Through Classification and Dimensionality Reduction". In: *arXiv preprint arXiv:1703.01842* (2017).

[NAK16] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. "Learning convolutional neural networks for graphs". In: *International Conference on Machine Learning*. 2016, pp. 2014–2023.

[Nek07] Maziar Nekovee. "Worm epidemics in wireless ad hoc networks". In: *New Journal of Physics* 9.6 (2007), p. 189.

[Opt] *Map of the Internet*. http://www.opte.org/maps/. Accessed: 2017-07-04. 2005.

[Pas+15a] Bastien Pasdeloup, Michael Rabbat, Vincent Gripon, Dominique Pastor, and Grégoire Mercier. "Graph reconstruction from the observation of diffused signals". In: *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*. IEEE. 2015, pp. 1386–1390.

[Pas+15b] Bastien Pasdeloup, Réda Alami, Vincent Gripon, and Michael Rabbat. "Toward an uncertainty principle for weighted graphs". In: *Signal Processing Conference (EUSIPCO), 2015 23rd European*. IEEE. 2015, pp. 1496–1500.

[Pas+16] Bastien Pasdeloup, Vincent Gripon, Grégoire Mercier, and Dominique Pastor. "Towards a characterization of the uncertainty curve for graphs". In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 4558–4562.

[Pas+17a] Bastien Pasdeloup, Vincent Gripon, Grégoire Mercier, Dominique Pastor, and Michael G Rabbat. "Characterization and inference of graph diffusion processes from observations of stationary signals". In: *IEEE Transactions on Signal and Information Processing over Networks* (2017).

[Pas+17b] Bastien Pasdeloup, Vincent Gripon, Nicolas Grelier, Jean-Charles Vialatte, and Dominique Pastor. "Translations on graphs with neighborhood preservation". In: *arXiv preprint arXiv:1709.03859* (2017).

[Pea01] Karl Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572.

[Per+14] Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst, and David K Hammond. "GSPBOX: A toolbox for signal processing on graphs". In: *arXiv preprint arXiv:1408.5781* (2014).

[Pet98] Julius Petersen. "Sur le théoreme de Tait". In: *L'intermédiaire des Mathématiciens* 5 (1898), pp. 225–227.

[PK17] Bastien Pasdeloup and Marine Karmann. "PyRat - Un jeu sérieux pour l'enseignement de l'informatique". In: *QPES 2017 : Questions de Pédagogies dans l'Enseignement Supérieur*. Grenoble, France, 2017.

[PKP17] Gilles Puy, Srdan Kitic, and Patrick Pérez. "Unifying local and non-local signal processing with graph CNNs". In: *CoRR* abs/1702.07759 (2017). URL: `http://arxiv.org/abs/1702.07759`.

[PM06] Markus Püschel and José M. F. Moura. "Algebraic Signal Processing Theory". In: *CoRR* abs/cs/0612077 (2006). URL: `http://arxiv.org/abs/cs/0612077`.

[PMTF17] Hermina Petric Maretic, Dorina Thanou, and Pascal Frossard. "Graph learning under sparsity priors". In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. EPFL-CONF-224359. 2017.

[PO16] Eduardo Pavez and Antonio Ortega. "Generalized Laplacian precision matrix estimation for graph signal processing". In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 6350–6354.

[Pri90] Roland Priemer. *Introductory signal processing*. World Scientific Publishing Co Inc, 1990.

[PV17] Nathanaël Perraudin and Pierre Vandergheynst. "Stationary signal processing on graphs". In: *IEEE Transactions on Signal Processing* 65.13 (2017), pp. 3462–3477.

[Rab17] Michael G Rabbat. "Inferring sparse graphs from smooth signals with theoretical guarantees". In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE. 2017, pp. 6533–6537.

[RG14] Michael G Rabbat and Vincent Gripon. "Towards a spectral characterization of signals supported on small-world networks". In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 4793–4797.

[Rin09] Francesco Rinaldi. "Mathematical Programming Methods for minimizing the zero-norm over polyhedral sets". In: *Sapienza, University of Rome. url: http://www. math. unipd. it/~ rinaldi/papers/thesis0. pdf* (2009).

[RMC08] Angela Re, Ivan Molineris, and Michele Caselle. "Graph theory analysis of genomics problems: Community analysis of fragile sites correlations and of pseudogenes alignments". In: *Computers & Mathematics with Applications* 55.5 (2008), pp. 1034–1043.

[Rot+08] Adam J Rothman, Peter J Bickel, Elizaveta Levina, Ji Zhu, et al. "Sparse permutation invariant covariance estimation". In: *Electronic Journal of Statistics* 2 (2008), pp. 494–515.

[Sab66] Gert Sabidussi. "The centrality index of a graph". In: *Psychometrika* 31.4 (1966), pp. 581–603.

[SBDL16] Stefania Sardellitti, Sergio Barbarossa, and Paolo Di Lorenzo. "Graph topology inference based on transform learning". In: *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on*. IEEE. 2016, pp. 356–360.

[SBG17] Yanning Shen, Brian Baingana, and Georgios B Giannakis. "Topology inference of directed graphs using nonlinear structural vector autoregressive models". In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE. 2017, pp. 6513–6517.

[Seg+16] Santiago Segarra, Antonio G Marques, Gonzalo Mateos, and Alejandro Ribeiro. "Network topology identification from spectral templates". In: *Statistical Signal Processing Workshop (SSP), 2016 IEEE*. IEEE. 2016, pp. 1–5.

[Seg+17a] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro. "Network Topology Inference from Spectral Templates". In: *IEEE Transactions on Signal and Information Processing over Networks* PP.99 (2017), pp. 1–1. DOI: 10.1109/TSIPN.2017.2731051.

[Seg+17b] Santiago Segarra, Antonio G Marques, Gonzalo Mateos, and Alejandro Ribeiro. "Robust Network Topology Inference". In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[Sha+17] Rasoul Shafipour, Santiago Segarra, Antonio G Marques, and Gonzalo Mateos. "Network topology inference from non-stationary graph signals". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[Sha49] Claude Elwood Shannon. "Communication in the presence of noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.

[SHG12] Shiliang Sun, Rongqing Huang, and Ya Gao. "Network-scale traffic modeling and forecasting with graphical lasso and neural networks". In: *Journal of Transportation Engineering* 138.11 (2012), pp. 1358–1367.

[Shl16] Jon Shlens. *Train your own image classifier with Inception in TensorFlow*. https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html. Accessed: 2017-08-24. Google Research Blog, 2016.

[SHR15] Santiago Segarra, Weiyu Huang, and Alejandro Ribeiro. "Diffusion and superposition distances for signals supported on networks". In: *IEEE Transactions on Signal and Information Processing over Networks* 1.1 (2015), pp. 20–32.

[Shu+13] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE Signal Processing Magazine* 30.3 (2013), pp. 83–98.

[Sil+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489.

[SJ10] Pannagadatta K Shivaswamy and Tony Jebara. "Laplacian spectrum learning". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2010, pp. 261–276.

[SM13] Aliaksei Sandryhaila and José MF Moura. "Discrete signal processing on graphs". In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.

[SM14] Aliaksei Sandryhaila and Jose MF Moura. "Discrete Signal Processing on Graphs: Frequency Analysis." In: *IEEE Trans. Signal Processing* 62.12 (2014), pp. 3042–3054.

[SP13] Shahin Shahrampour and Victor M Preciado. "Reconstruction of directed networks from consensus dynamics". In: *American Control Conference (ACC), 2013*. IEEE. 2013, pp. 1685–1690.

[SP15] Shahin Shahrampour and Victor M Preciado. "Topology identification of directed dynamical networks via power spectral analysis". In: *IEEE Transactions on Automatic Control* 60.8 (2015), pp. 2260–2265.

[SRV12] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. "A windowed graph Fourier transform". In: *Statistical Signal Processing Workshop (SSP), 2012 IEEE*. Ieee. 2012, pp. 133–136.

[SSJ17] Santiago Segarra, Michael T. Schaub, and Ali Jadbabaie. "Network Inference from Consensus Dynamics". In: *arXiv preprint arXiv:1708.05329* (2017).

[Sze+16] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex A. Alemi. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *ICLR 2016 Workshop*. 2016. URL: https://arxiv.org/abs/1602.07261.

[TBDL16] Mikhail Tsitsvero, Sergio Barbarossa, and Paolo Di Lorenzo. "Signals on graphs: Uncertainty principle and sampling". In: *IEEE Transactions on Signal Processing* 64.18 (2016), pp. 4845–4860.

[Tha+16] Dorina Thanou, Xiaowen Dong, Daniel Kressner, and Pascal Frossard. "Learning heat diffusion graphs". In: *arXiv preprint arXiv:1611.01456* (2016).

[Tig+16] Philippe Tigréat, Carlos Rosar Kos Lassance, Xiaoran Jiang, Vincent Gripon, and Claude Berrou. "Assembly output codes for learning neural networks". In: *Turbo Codes and Iterative Information Processing (ISTC), 2016 9th International Symposium on*. IEEE. 2016, pp. 285–289.

[TTT99] Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. "SDPT3 — a MATLAB software package for semidefinite programming, version 1.3". In: *Optimization methods and software* 11.1-4 (1999), pp. 545–581.

[TV16] Oguzhan Teke and PP Vaidyanathan. "Discrete uncertainty principles on graphs". In: *Signals, Systems and Computers, 2016 50th Asilomar Conference on*. IEEE. 2016, pp. 1475–1479.

[TWS15] Kean Ming Tan, Daniela Witten, and Ali Shojaie. "The cluster graphical lasso for improved estimation of Gaussian graphical models". In: *Computational statistics & data analysis* 85 (2015), pp. 23–36.

[VGC17] Jean-Charles Vialatte, Vincent Gripon, and Gilles Coppin. "Learning Local Receptive Fields and their Weight Sharing Scheme on Graphs". In: *CoRR* abs/1706.02684 (2017). URL: http://arxiv.org/abs/1706.02684.

[VGM16] Jean-Charles Vialatte, Vincent Gripon, and Grégoire Mercier. "Generalizing the Convolution Operator to Extend CNNs to Irregular Domains". In: *CoRR* abs/1606.01166 (2016). URL: http://arxiv.org/abs/1606.01166.

[Wei58] Karl Weierstraß. *Über ein die homogenen Funktionen zweiten Grades betreffendes Theorem*. Monatsh. Akademie der Wissenschaften, Berlin, 1858.

[Wer76] Nanny Wermuth. "Analogies between multiplicative models in contingency tables and covariance selection". In: *Biometrics* (1976), pp. 95–108.

[WFS11] Daniela M Witten, Jerome H Friedman, and Noah Simon. "New insights and faster computations for the graphical lasso". In: *Journal of Computational and Graphical Statistics* 20.4 (2011), pp. 892–900.

[WP09] Wei Biao Wu and Mohsen Pourahmadi. "Banding sample autocovariance matrices of stationary processes". In: *Statistica Sinica* (2009), pp. 1755–1768.

[WS98] Duncan J Watts and Steven H Strogatz. "Collective dynamics of 'small-world'networks". In: *nature* 393.6684 (1998), pp. 440–442.

[XW+12] Han Xiao, Wei Biao Wu, et al. "Covariance matrix estimation for stationary time series". In: *The Annals of Statistics* 40.1 (2012), pp. 466–493.

[Yan+15] Sen Yang, Qian Sun, Shuiwang Ji, Peter Wonka, Ian Davidson, and Jieping Ye. "Structural graphical lasso for learning mouse brain connectivity". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2015, pp. 1385–1394.

Index

Abstract

* * *

This manuscript sums up our work on extending convolutional neural networks to irregular domains through graph inference. It consists of three main chapters, each giving the details of a part of a methodology allowing the definition of such networks to process signals evolving on graphs with unknown structures.

First, graph inference from data is explored, in order to provide a graph modeling the support of the signals to classify. Second, translation operators that preserve neighborhood properties of the vertices are identified on the inferred graph. Third, these translations are used to shift a convolutional kernel on the graph in order to define a convolutional neural network that is adapted to the input data.

We have illustrated our methodology on a dataset of images. While not using any particular knowledge on the signals, we have been able to infer a graph that is close to a grid. Translations on this graph resemble Euclidean translations. Therefore, this has allowed us to define an adapted convolutional neural network that is very close what one would obtain when using the information that signals are images. This network, trained on the initial data, has outperformed state of the art methods by more than 13 points, while using a very simple and easily improvable architecture.

The method we have introduced is a generalization of convolutional neural networks. As a matter of fact, they can be seen as a particularization of our approach in the case where the graph is a grid. Our work thus opens the way to numerous perspectives, as it provides an efficient way to build networks that are adapted to the data.

* * *

Keywords : Graph signal processing, graph inference, translations on graphs, convolutional neural networks.

Résumé

* * *

Ce manuscrit résume nos travaux sur l'extension des réseaux de neurones convolutifs à des domaines irréguliers par l'inférence de graphe. Il consiste en trois grands chapitres, chacun détaillant une partie d'une méthodologie nous permettant d'appliquer de tels réseaux à des signaux évoluant sur des graphes inconnus.

Tout d'abord, nous présentons des méthodes permettant d'inférer un graphe à partir de signaux, afin de modéliser le support des données à classifier. Ensuite, des translations préservant les voisinages des sommets sont identifiées sur le graphe inféré. Enfin, ces translations sont utilisées pour déplacer un noyau convolutif sur le graphe, afin de définir un réseau de neurones convolutif adapté aux données d'entrée.

Nous avons illustré notre méthodologie sur une base de données d'images. Sans utiliser de connaissances sur les signaux, nous avons pu inférer un graphe proche d'une grille. Les translations sur ce graphe sont proches des translations Euclidiennes, ce qui nous a permis de définir un réseau de neurones convolutif très similaire à ce que l'on aurait pu obtenir en utilisant l'information que les signaux sont des images. Ce réseau, entraîné sur les données initiales, a dépassé les performances des méthodes de l'état de l'art de plus de 13 points, tout en étant simple et facilement améliorable.

La méthode que nous avons introduite est une généralisation des réseaux de neurones convolutifs, car ceux-ci sont des cas particuliers de notre approche quand le graphe est une grille. Nos travaux ouvrent donc de nombreuses perspectives, car ils fournissent une méthode efficace pour construire des réseaux adaptés aux données.

* * *

Mots-clés : Traitement de signal sur graphe, inférence de graphe, translations sur graphes, réseaux de neurones convolutifs.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

N° d'ordre : 2017IMTA0048

IMT Atlantique
Technopôle Brest-Iroise CS 83818
29238 Brest Cedex 03, France

Résumé

Tout d'abord, nous présentons des méthodes permettant d'inférer un graphe à partir de signaux, afin de modéliser le support des données à classifier. Ensuite, des translations préservant les voisinages des sommets sont identifiées sur le graphe inféré. Enfin, ces translations sont utilisées pour déplacer un noyau convolutif sur le graphe, afin de définir un réseau de neurones convolutif adapté aux données d'entrée.

Nous avons illustré notre méthodologie sur une base de données d'images. Sans utiliser de connaissances sur les signaux, nous avons pu inférer un graphe proche d'une grille. Les translations sur ce graphe sont proches des translations Euclidiennes, ce qui nous a permis de définir un réseau de neurones convolutif très similaire à ce que l'on aurait pu obtenir en utilisant l'information que les signaux sont des images. Ce réseau, entraîné sur les données initiales, a dépassé les performances des méthodes de l'état de l'art de plus de 13 points, tout en étant simple et facilement améliorable.

La méthode que nous avons introduite est une généralisation des réseaux de neurones convolutifs, car ceux-ci sont des cas particuliers de notre approche quand le graphe est une grille. Nos travaux ouvrent donc de nombreuses perspectives, car ils fournissent une méthode efficace pour construire des réseaux adaptés aux données.

Mots clef : Informatique, Inférence de graphe, Traitement de signal 6 Apprentissage, Traitement de signal sur graphe, Réseaux de neurones, Machine Learning

Abstract

This manuscript sums up our work on extending convolutional neural networks to irregular domains through graph inference. It consists of three main chapters, each giving the details of a part of a methodology allowing the definition of such networks to process signals evolving on graphs with unknown structures.

First, graph inference from data is explored, in order to provide a graph modeling the support of the signals to classify. Second, translation operators that preserve neighborhood properties of the vertices are identified on the inferred graph. Third, these translations are used to shift a convolutional kernel on the graph in order to define a convolutional neural network that is adapted to the input data.

We have illustrated our methodology on a dataset of images. While not using any particular knowledge on the signals, we have been able to infer a graph that is close to a grid. Translations on this graph resemble Euclidean translations. Therefore, this has allowed us to define an adapted convolutional neural network that is very close what one would obtain when using the information that signals are images. This network, trained on the initial data, has outperformed state of the art methods by more than 13 points, while using a very simple and easily improvable architecture.

The method we have introduced is a generalization of convolutional neural networks. As a matter of fact, they can be seen as a particularization of our approach in the case where the graph is a grid. Our work thus opens the way to numerous perspectives, as it provides an efficient way to build networks that are adapted to the data.

Keywords: Graph signal processing, Graph inference, Translations on graphs, Convolutional neural networks

N° d'ordre : 2017IMTA0048

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

IMT Atlantique Bretagne-Pays de la Loire - www.imt-atlantique.fr

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 03
T +33 (0)2 29 00 11 11
F +33 (0)2 29 00 10 00

Campus de Nantes
4, rue Alfred Kastler - La Chantrerie
CS 20722
44307 Nantes Cedex 3
T +33 (0)2 51 85 81 00
F +33 (0)2 51 85 81 99

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
T +33 (0)2 99 12 70 00
F +33 (0)2 99 12 70 08