Network analysis based on bag-of-paths: classification, node criticality and randomized policies

Author:
Bertrand LEBICHOT

Advisor:
Prof. Marco SAERENS

Thesis submitted in fulfillment
of the requirements for the degree of

Doctor of Engineering Science

of the Université catholique de Louvain

PhD jury:

Abstract

Network analysis and graph mining are used in many fields such as chemistry, biology, physics, human sciences and (computer) engineering, but also in everyday life from route planners to recommendation systems and social networks tools.

In this context, nodes represent concepts of interest and edges represent a given relation between nodes. In this elegant abstraction, finding the shortest path is probably the most well-known problem and is common to all graphs textbooks. This problem consists in the identification of the shortest (in terms of cost or length), most effective path, between two nodes.

However, the shortest path forgets about the rest of the network and therefore fails to report some important information present in this network. For example, it does not integrate the degree of connectivity between nodes. In many applications, nodes connected by many indirect paths should be considered as "closer" than nodes connected by only a few paths. On the other hand, measures taking connectivity into account (such as the commute-time distance) have their own drawbacks, especially when networks become large.

The bag-of-paths framework defines a family of distances interpolating between the shortest path and the commute-time distances. In doing so, the framework takes into account both proximity between nodes in the network and amount of connectivity. It is also possible, via a temperature parameter, to emphasize more on network exploitation or on network exploration.

This thesis is based on the bag-of-paths framework and introduces several of its applications: a new classifier based on a bag-of-paths group betweenness for graph-based semi-supervised classification, a new graph criticality measure and an algorithm to determine an optimal, mixed, policy for Markov decision processes. Other applications, closely related to the bag-of-paths framework, are also proposed.

Acknowledgements

I would like to acknowledge a lot of people (not necessarily in that order) for their contribution, passive or active, to this thesis:

- My advisor M. Saerens, for answering my questions and guiding me. His advices, kindness and availability really made this long-term work easier on a daily basis.
- I would like to thank the jury members: Prof. M. Saerens, Prof. M. Verleysen and Prof. F. Rossi who significantly improved this work with their remarks and suggestions from the beginning. I also thank Prof. F. Fouss and Dr. O. Caelens for accepting to be part of the jury. Finally, I thank Prof. C. Pecheur for accepting to be the President.
- I express my gratitude to my colleages of UCL and co-authors : Prof. F. Fouss, Prof. P. Franck, Prof. M. Kolp, Prof. J. Vanderdonckt, Dr. I. Kivimaki, Ir. K. Francoisse, Dr. G. Guex, Dr. M. Zen, Dr. M. Madany, Dr. S. G. Diez, Dr. T.-D. Nguyen, Dr F. Sommer, Ir V. Vandenbulcke, Ir. E. Freiss, Dr. R. Devooght, Dr. S. Heng. and many more. Our exchanges went from rigorous explanations to (less studious) great moments. I thank all of them for those enjoyable six years at the Louvain School of Management.
- Another part of my feelings goes to my family and friends support, for all those kind chats about the progression of the current thesis. I want to especially thank my parents (I obviously wouldn't be there without them) and my beloved wife, who actually often understood what I did (something quite rare for a PhD student). Also, a special thank goes to my daughters Emilie and Louise for substantially increasing the time required for this thesis...
- This work was partially supported by the Elis-IT project funded by the "Région wallonne" and the Immediate and the Brufence projects funded by "InnovIris" (Brussels Region). I acknowledge those institutions for giving us the opportunity to conduct both fundamental and applied research.

Contents

List of Figures

List of Tables

List of Symbols

(Inspired by [90])

s, x, y, z, etc	scalar or random variables, depending on the context.
$\mathbf{x}, \mathbf{y}, \mathbf{z}$, etc	random vectors.
t	the time (either discrete or continuous).
α, θ, etc	scalar parameters.
$\mathcal{S}$	a set, in calligraphic letters.
$\varnothing$	the empty set.
$\lvert\mathcal{S}\rvert = \#\mathcal{S}$	the number of elements, or cardinality, of a set $\mathcal{S}$.
P(some predicate)	the probability that the predicate containing random variables is true.
$\hat{\text{P}}$(some predicate)	an estimate of the probability based on empirical data.
$\text{P}(s = i)$	the probability that the discrete random variable s takes value i.
$\text{P}(s = i, e = j)$	the probability of drawing a path starting in node i and ending in node j from the virtual bag-of-paths.
$p_{ij} = \text{P}(s_t = j \mid s_{t-1} = i)$	the elements of the transition probability matrix in a Markov chain.
$\mathbb{E}[s]$	the expectation with respect to a random variable s.
$\mathbf{M}$	a matrix (always uppercase bold).
$\mathbf{M}^{\text{T}}$	the transpose of matrix $\mathbf{M}$ containing elements $m^{\text{t}}_{ij} = m_{ji}$.
$\mathbf{M}^{q}$	the matrix q-power of $\mathbf{M}$
$\mathbf{M}^{(q)}$	the Hadamard (elementwise) q-power of $\mathbf{M}$ containing elements m^{q}_{ij}.
$\mathbf{M} \circ \mathbf{N}$	the Hadamard (elementwise) matrix multiplication providing elements $m_{ij}n_{ij}$.
$\mathbf{M} \div \mathbf{N}$	the Hadamard (elementwise) matrix division providing elements m_{ij}/n_{ij}.

$\mathbf{M}^{\div}$	the elementwise reciprocal of $\mathbf{M}$ containing elements $m_{ij}^{\div} = 1/m_{ij}$.
$\mathbf{M}^{+}$	the Moore-Penrose pseudoinverse of $\mathbf{M}$.
$\exp[\mathbf{M}]$	the elementwise exponential of matrix $\mathbf{M}$.
$\mathrm{expm}[\mathbf{M}] = e^{\mathbf{M}}$	the matrix exponential of matrix $\mathbf{M}$.
$\mathbf{I}$	the identity matrix.
$m_{ij} = [\mathbf{M}]_{ij}$	the element i, j (in the ith row and the jth column) of matrix $\mathbf{M}$.
$m_{ij}^{(q)} = [\mathbf{M}^q]_{ij}$	the element i, j of $\mathbf{M}^q$, the matrix q-power of $\mathbf{M}$.
$m_{i\bullet} = \sum_j m_{ij}$	the sum of the elements of the ith row of the matrix $\mathbf{M}$ (row sum).
$m_{\bullet j} = \sum_i m_{ij}$	the sum of the elements of the jth column of the matrix $\mathbf{M}$ (column sum).
$m_{\bullet\bullet} = \sum_{ij} m_{ij}$	the sum over all the elements of the matrix $\mathbf{M}$.
$\mathbf{Diag}(\mathbf{M})$	the diagonal matrix containing the diagonal of the square matrix $\mathbf{M}$.
$\mathbf{vec}(\mathbf{M})$	an operator stacking the columns of $\mathbf{M}$ in a column vector.
$\mathbf{v}$	a column vector (always lowercase bold).
$\mathbf{v}^{\mathrm{T}}$	a row vector; the transpose of column vector $\mathbf{v}$.
$\mathbf{diag}(\mathbf{M})$	the column vector containing the diagonal of the square matrix $\mathbf{M}$.
$\mathbf{Diag}(\mathbf{v})$ or $\mathbf{Diag}(v_i)$	the diagonal matrix containing the vector $\mathbf{v}$ on its diagonal.
$v_i^c = [\mathbf{v}_c]_i$	ith element of column vector $\mathbf{v}_c$.
$\mathbf{e}$	a unit column vector full of 1's of the appropriate size.
$\mathbf{0}$	a null column vector full of 0's of the appropriate size.
$\mathbf{e}_i$	ith column of $\mathbf{I}$, containing zero everywhere except on row i.
$\mathbf{K}$	a $n \times n$ kernel on the graph G, or $n_s \times n_s$ kernel obtained from matrix $\mathbf{X}$.
Δ_{ij}	the dissimilarity (or distance) between node i and node j.
$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^{\mathrm{T}}\mathbf{y}$	the inner product between vector $\mathbf{x}$ and vector $\mathbf{y}$.
$\mathbf{E} = \mathbf{e}\mathbf{e}^{\mathrm{T}}$	a matrix full of 1's.
$\mathbf{O}$	a matrix full of 0's.
$\mathbf{H} = \mathbf{I} - \dfrac{\mathbf{E}}{n}$	the $n \times n$ centering matrix. $\mathbf{Hx}$ provides a centered vector whose entries sum to 0.
δ_{kl}	the Kronecker delta whose value is 1 if $k = l$, and 0 otherwise.

δ(some predicate)	equals 1 if the predicate is true, and 0 otherwise.
$\mathscr{L}$	the Lagrange function.
$\mathcal{L}(\boldsymbol{\theta})$	the likelihood function.
G, H	a graph or a subgraph (connected or not, weighted or not, directed or not, containing self-loops or not). By extension, the set of nodes and edges composing G, H.
$G \setminus i$	the subgraph of G obtained by deleting node i from G, as well as its adjacent edges.
$\mathcal{V}$, $\mathcal{V}(G)$, or simply G	the set of nodes (or vertices) of a graph or subgraph G.
$v \in \mathcal{V}$	a node or vertice belonging to the set $\mathcal{V}$.
$\mathcal{E}$ or $\mathcal{E}(G)$	the set of arcs (or edges, links) of a graph or subgraph G.
v_i, node i, or simply i	the node whose index is i (ith node of G).
$i \to j$	an edge or link connecting node i and node j.
$i \rightsquigarrow j$	a path or walk connecting node i and node j.
$\mathcal{N}(k)$	the set of neighbors of node k; these nodes are also called adjacent or incident nodes.
$\mathcal{N}^{(t)}(k)$	the set of t-steps neighbors of node k.
$n = \lvert\mathcal{V}(G)\rvert = n(G)$	the number of nodes of graph G.
$e = \lvert\mathcal{E}(G)\rvert$	the number of edges of graph G.
$\mathbf{A}$	the adjacency matrix of G: $a_{ij} = w_{ij}$ when there is an edge between i and j; $a_{ij} = 0$ otherwise.
$\mathbf{D} = \mathbf{Diag}(\mathbf{Ae})$	the diagonal degree matrix of G containing the $a_{i\bullet}$ on its diagonal. Also sometimes called generalized degree or strength of the nodes for weighted graphs.
$\mathbf{d} = \mathbf{Ae}$	the $n \times 1$ degree vector of the undirected graph G containing the $a_{i\bullet} = a_{\bullet i}$.
$\mathbf{C}$	the cost matrix associated to a graph G containing transition costs c_{ij}.
$\mathbf{P}$	the $n \times n$ transition matrix of a time-independent Markov chain. Contains the elements $p_{ij} = \mathrm{P}(s(t) = j \mid s(t-1) = i)$ where $s(t)$ is a random variable representing the state of the Markov chain at time step t.
$\mathbf{W} = \mathbf{P}^{\text{ref}} \circ \exp[-\theta\mathbf{C}]$	a $n \times n$ transition matrix.
$\mathbf{X}$	the $n_s \times m$ data matrix containing n_s samples on its rows and m features on its columns.

$\mathbf{P}^{\text{ref}}$	the probability of following the path $\wp$ when walking according to the natural random walk reference distribution.
$\tilde{\pi}(\wp)$	the likelihood of the path $\wp$.
$\tilde{\pi}^{\text{ref}}$	the probability of following path $\wp$ when walking according to the natural random walk reference distribution. The product of the transition probabilities p_{ij}^{ref} along the path $\wp$. The likelihood of the path $\wp$ according to the natural random walk reference distribution.
$\boldsymbol{\pi}$	the stationary distribution of a primitive Markov chain.
$\mathbf{L} = \mathbf{D} - \mathbf{A}$	the $n \times n$ Laplacian matrix of the undirected graph G.
$\mathbf{Q}$	the $n \times n$ modularity matrix of G.
$\mathbf{y}$	the column-vector containing the target variable.
$\mathbf{y}^c$	the column-vector containing the binary memberships of the nodes to class c.
$\mathbf{Y}$	the matrix containing $\mathbf{y}_c$ for all possible c.
bet(k)	the betweenness of node k.
$\mathbf{bet}$	the betweenness vector.
cr(G)	the criticality of graph G.
cr$(G)_i$	the node criticality of node i on graph G.
$\mathcal{C}_k$	the set containing all samples of class k.
$\mathcal{P}$	the set of all possible paths.
$\mathcal{P}_{ij}$	the set of all possible paths starting from node i and ending in node j. including loops.
$\mathcal{P}_{ij}^{\text{h}}$	the set of all possible hitting paths starting from node i and ending in node j. including loops.
$\mathcal{P}_{ij}(t)$	the set of all t-steps paths of G, starting from node i and ending in node j, including loops.
$\wp$	a particular path of G.
$\wp^{\text{h}}$	a particular hitting path of G.
$\wp_{ij}$	a particular path starting in node i and ending in node j
$\wp_{ij}^{\text{h}}$	a particular hitting path starting in node i and ending in node j
J_0	a fixed relative entropy for the bag-of-paths framework.
$T = 1/\theta$	the temperature of the system; θ being the inverse temperature.
ϕ	the free energy.

$\tilde{c}(\wp)$	the total cost along path $\wp$. That is, the sum of the individual transition costs c_{ij} along $\wp$.
$\mathcal{Z}$	the partition function.
I	Moran's index (see Subsection 6.3.1).
c	Geary's c (see Subsection 6.3.1).
$\triangleq$	equal and defined as.

To my family.

Chapter 1

Introduction

Except if he lives in a cave, every single person will agree that nowadays Information and Communications Technologies (ICT) take a greater and greater importance in our lives: Internet, social networks, Internet of Things (IoT), Artificial Intelligence (AI) are concepts that everybody should at least be aware of.

Some authors (see [212] as an example) even assimilate ICT to a Fourth Industrial Revolution (or third, according to other authors, since the border between the third and the fourth is fuzzy):

- The First Industrial Revolution took place from the 18th to 19th centuries in Europe and America. It was characterized by a conversion from an agrarian, rural society to an industrial and urban one. Iron, steam engine, and textile industries were the big improvements of this period, which stays associated to the Victorian age in the United Kingdom.
- The Second Industrial Revolution took place in the late 19th, begin 20th centuries. Steel, mass production, oil (and chemical industry), and electrical power were the major improvements at that time. The society became more specialized and still more urban (as industries are located near cities).
- The (not universally accepted) Third Industrial Revolution started in mid 20th century and is still going on. Analogical and mechanical electronic devices let the place to the digital technology available today. For this reason, it is also called the Digital Revolution. This era advancements were, among other, computers, personal computers, cellular phones, and the Internet.
- The Fourth Industrial Revolution builds on the Digital Revolution after the beginning of the 21th century. The technology becomes connected, embedded within society and even within human body. This Revolution keywords are robotics, artificial intelligence, nanotechnology, biotechnology, Internet of Things, the Cloud, and autonomous vehicles.

Like revolutions that preceded it, the Fourth Industrial Revolution has the potential to increase global income levels and improve the quality of life for populations around the world [212]. But those rapid changes also left some people behind.

This thesis is a small contribution to this revolution, developing data mining tools in this context. The volume of data generated by internet and social networks is increasing every day, and there is a clear need for efficient ways of extracting useful information from them.

However, this work is not a reflexion on how those tools should be used (indeed, a few more theses could be written on this subject, especially in our fast evolving society). The main concern is about designing new computer-based tools to solve practical problems.

The whole thesis is devoted to networked (or graph-based) data. Such representation is a natural abstraction for a lot of interesting concepts:

- ▶ Social networks are people's profiles linked by friendship or professional links.
- ▶ Maps are cities linked by roads.
- ▶ Internet is composed of files linked by hyperlinks.
- ▶ (Bio)-chemical databases consist of molecules linked by chemical interactions.
- ▶ Power grid are producers and consumers linked by power lines.

Of course some of those concepts can be studied outside of graph theory, but in this thesis, this framework was systematically chosen.

In particular, we are interested into automatic graph-based classification (Chapter 5 and Chapter 6), graph-based criticality (Chapter 7), automatic fraud detection (Chapter 8) and Markov decision processes (Chapter 9). We worked on making accurate predictions, and on handling large amounts of data.

The rest of this introduction is composed of a plan of the thesis in Section 1.1 and of a description of the different chapters and associated publications in Section 1.2.

1.1 Outline

This thesis is organized as follows:

- ▶ Chapter 2 reviews the underlying concepts of graph theory.
- ▶ Chapter 3 follows the same objective for semi-supervised learning.

- Chapter 4 introduces the bag-of-paths framework, which is used in Chapter 5, 6, 7 and 9.
- Chapter 5 derives a classifier based on a Bag-of-Path group betweenness for graph-based semi-supervised classification.
- Chapter 6 investigates different semi-supervised classification methods to compare feature based approaches, graph structure based approaches, and dual approaches combining both information sources.
- Chapter 7 derives a graph criticality measure based on the bag-of-paths framework.
- Chapter 8 tackles fraud detection based on a graph structure and semi-supervised learning.
- Chapter 9 studies Markov decision processes based on the randomized shortest path framework (RSP), which is really close to the bag-of-paths framework.
- Finally, chapter 10 concludes this thesis.

1.2 Associated publications

Here is the list of scientific publications made in the context of this work:

1. The work presented in Chapter 5 has been published in:

   Bertrand Lebichot, Kevin Francoisse, Ilkka Kivimaki and Marco Saerens. Semi-Supervised Classification through the Bag-of-Paths Group Betweenness. In *IEEE Transactions on Neural Networks and Learning Systems* 25 (6 2014), pp. 1173–1186.

   - Bertrand Lebichot is with LOURIM and ICTEAM from the Université catholique de Louvain, Louvain-la-Neuve, Belgium.
   - Kevin Francoisse is with ICTEAM from the Université catholique de Louvain, Louvain-la-Neuve, Belgium.
   - Ilkka Kivimaki is with ICTEAM from the Université catholique de Louvain, Louvain-la-Neuve, Belgium and with the Department of Computer Science Aalto University, Helsinki, Finland.
   - Marco Saerens is with ICTEAM from the Université catholique de Louvain, Louvain-la-Neuve, Belgium.

2. The work presented in Chapter 6 has been submitted as:

   Bertrand Lebichot and Marco Saerens. An experimental study of graph-based semi-supervised classification with additional node information. *Currently in reviewing (second round) in Information Fusion*.

3. The work presented in Chapter 7 has been published in:

   Bertrand Lebichot and Marco Saerens. A Bag-of-Paths Node Criticality Measure. In *Neurocomputing, Volume 275 (January 2018), pp. 224–236*.

4. The work presented in Chapter 8 has been published in:

   Bertrand Lebichot, Fabian Braun, Olivier Caelens and Marco Saerens. A graph-based, semi-supervised, credit card fraud detection system. *In: Complex Networks & Their Applications V: Proceedings of the 5th InternationalWorkshop on Complex Networks and their Applications (COMPLEX NETWORKS 2016).* Ed. by H. Cherifi et al. Cham: Springer International Publishing, 2017, pp. 721–733.

   - Fabian Braun is with the R&D department from Worldline GmbH, Germany.
   - Olivier Caelens is with the R&D department from Worldline SA/NV, Belgium.

5. The work presented in Chapter 9 has to be submitted soon as: Bertrand Lebichot, Guillaume Guex, Ilkka Kivimaki and Marco Saerens. Constrained Randomized Shortest Path Problems. *To be submitted soon*.

   - Guillaume Guex is with ICTEAM from the Université catholique de Louvain, Louvain-la-Neuve, Belgium.
   - We also thank Benjamin Blaise, a former Master student, who helped us to investigate the randomized Markov decision processes in his Master thesis.

Chapter 2

Graphs and networks

This chapter introduces graphs and related concepts that are used throughout this thesis. The following topics are covered: Section 2.1 is a short introduction. Section 2.2 presents the different types of graphs. Sections 2.3 and 2.4 introduces respectively basic notations and matrix notations. Section 2.5 focuses on paths and cycles. Section 2.6 briefly discusses the shortest path problem. Section 2.7 reviews Markov chains. Finally, Section 2.8 is dedicated to random graph generators.

2.1 An introduction to graphs

Graph theory is a relatively recent field of mathematics. The first book [150] about it was published in 1936. However, Euler's work [84], on the seven bridges of Königsberg (in Prussia, later renamed Kaliningrad) is often considered as the first work on graphs, and was published two centuries earlier [110, 204, 4, 90].

Graph theory covers well known problems such as the Traveling Saleman Problem, Eulerian paths, and Graph Coloring to name a few [110, 204, 252, 42, 4]. More recently, thanks to computer developments and the availability of large networks, a lot of new work emerged, often linked with other fields such as chemistry, biology, physics, human sciences, and engineering [90].

This chapter, largely inspired by [90] but also by [204, 110, 42, 183], can be seen as a reminder about graph theory, although only the concepts used later in this thesis are covered. For a wider review, a lot of detailed textbooks are available (see for example [110, 204, 252, 42, 4, 90, 38]).

2.2 Graph types

Graphs are discrete structures composed of nodes and edges connecting those nodes. A graph or network G (both terms will be used interchangeably) can be formally defined by two sets:

- a finite non-empty set $\mathcal{V}(G) = \mathcal{V}$ whose elements are called **nodes** (sometimes vertices).
- a set $\mathcal{E}(G) = \mathcal{E}$ whose elements are links between pairs of nodes of $\mathcal{V}$ and are called **edges** (sometimes arcs or simply links). An edge somehow is said to connect its endpoint nodes.

A graph is therefore a collection of nodes linked by edges, $(\mathcal{V}, \mathcal{E})$.

Graphs are often used as a convenient way for representing pairwise relations between objects of interest: nodes represent the objects or entities and edges represent binary relations between those objects. For instance, in a social network, nodes can represent people and edges can be a friendship link between them. The number of nodes will be noted $n = |\mathcal{V}|$ and the number of edges will be noted $e = |\mathcal{E}|$. An edge connecting a certain node i to a certain node j will be noted $i \to j$.

In this thesis, different kinds of graphs are used and it is important to keep in mind the properties of those graphs: Links can be oriented (or not), loops can be present (or not), and multiple edges can be present between two nodes... Sadly, there is no unified way to name graphs according to their properties. This lack of uniformity is somehow due to the relatively modern interest in graph theory, and because it has applications in a wide variety of disciplines. Table 2.1 summarized all possible type of graph. Nevertheless, there are three main properties that are commonly accepted [204]:

- Multiple edges between two nodes can be allowed. For example, three edges can be present between two nodes, encoding different weights compared to the case where there would be only one edge. In this case, the graph is called a **multiple graph** or **multigraph**. If edges between two nodes are restricted to be unique, or absent (kind of boolean) and that no loops are present, the graph is called a **simple graph**. If loops are allowed it is called a **simple graph without loops**.
- Those edges can be oriented or not. If an edge (or path, see Section 2.5) is oriented, the relationship is directed. A transition on that edge is only allowed in one direction and the graph is called a **directed graph** (or **digraph**). Note that a second edge can link the two same nodes in the opposite direction. A graph with no oriented edges is called **undirected**.
- The presence of **(self-)loops**, i.e. edges that start and end on the same node, can be allowed or forbidden. Graphs that may include loops, and possibly multiple edges connecting the same pair of vertices or a vertex to itself, are sometimes called **pseudographs**. A graph without self-loops and without multiple edges is called a **simple graph**.

TABLE 2.1: Graph nomenclature used this work. Notice that the status about the weights on the edges must also be specified using weighted or unweighted.

Loops forbidden		**Multiple edges ?**	
		no	yes
Directed edges ?	no	undirected simple graph	undirected multiple graph undirected multigraph
	yes	directed simple graph simple digraph	directed multiple graph directed multigraph

Loops allowed		**Multiple edges ?**	
		no	yes
Directed edges ?	no	undirected simple graph with loops undirected pseudograph with loops	undirected multiple graph with loops undirected pseudograph with loops
	yes	directed simple graph with loops directed pseudograph with loops	directed multiple graph with loops directed pseudograph with loops

Furthermore, weights on edges can be defined: as opposed to simple boolean edges, and the interpretation differs whether or not the graph is directed.

- In the case of a **weighted undirected graph**, a non-negative symmetric **weight** w_{ij} (with $w_{ij} = w_{ji}$), quantifying the degree of "affinity", the degree of "similarity" or the "closeness" between the two nodes i and j, is associated to each edge. In a co-authorship network, weights could be set, e.g., to the number of papers co-signed by two authors. An undirected graph can be considered as a directed graph with each edge having the same weight and being bi-directional. A graph without weights assigned to edges is called **unweighted**.
- If the graph is **weighted** and **directed**, the (directed) weight w_{ij} can usually be interpreted as a degree of endorsement, credit, reward, or dependency of object i towards object j. It roughly defines a binary relation between pairs of nodes in which the starting node delivers some kind of "credit" to the ending node. This happens, e.g. in a citation network where papers are citing other papers. On the contrary, for some directed networks, the weights on the edges could instead reflect a relation of dominance or influence of i on j.

2.3 Basic graph definitions

Two nodes from $\mathcal{V}$ are **adjacent** if there is an edge in $\mathcal{E}$ connecting these two nodes. Additionally, a node and an edge are **incident** if the edge is connected to the node.

A graph H is a **subgraph** of G if $\mathcal{V}(H) \subseteq \mathcal{V}(G)$ and $\mathcal{E}(H) \subseteq \mathcal{E}(G)$. Notice that $\mathcal{V}(H)$ and $\mathcal{E}(H)$ must be coherent. Those subgraphs can have other properties: centered around a particular node, for instance obtained from a cluster of G, or composed of multiple disconnected subgraphs,... Many computational tasks involve identifying subgraphs of various types.

The concept of degree is essential in graph theory. It depends on the type of graph but is always related to the number of edges incident to a considered node:

- For unweighted undirected graph, the **degree** of a node is the number of edges incident with it or, equivalently, the number of nodes adjacent to it. A node with a degree equal to 0 is called an isolated node. Every finite (i.e. with a finite number of nodes) undirected graph has an even number of vertices with odd degree (the handshaking lemma [204]).
- For weighted graph, the **generalized degree**, or simply the **degree**, of a node is the sum of the weights (the total weight) of the edges incident with it.
- For directed graphs, **indegrees** and **outdegrees** must be introduced: the indegree of a node is the number of ingoing edges (or the total weight for a weighted graph) ending at the considered node while its outdegree is the number of outgoing edges (or the total weight for a weighted graph) starting from the considered node.

A **bipartite graph** (or bigraph) is a graph whose nodes can be divided into two disjoint subsets $\mathcal{X}$ and $\mathcal{Y}$ such that each edge links a node in $\mathcal{X}$ to a node in $\mathcal{Y}$, or vice-versa. In other words, for all edges $i \to j \in \mathcal{E}$, either $i \in \mathcal{X}$ and $j \in \mathcal{Y}$, or $i \in \mathcal{Y}$ and $j \in \mathcal{X}$, with $\mathcal{X} \cap \mathcal{Y} = \varnothing$ and $\mathcal{X} \cup \mathcal{Y} = \mathcal{V}$. Consequently, there are no edge connecting two nodes in $\mathcal{X}$ or connecting two nodes in $\mathcal{Y}$.

Similarly, a **tripartite graph** is a graph whose nodes can be divided into three disjoint subsets $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ (with $\mathcal{X} \cap \mathcal{Y} = \varnothing$, $\mathcal{Y} \cap \mathcal{Z} = \varnothing$, $\mathcal{X} \cap \mathcal{Z} = \varnothing$, and $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z} = \mathcal{V}$) such that there are no edges connecting two nodes belonging to the same subset $\mathcal{X}$, $\mathcal{Y}$, or $\mathcal{Z}$ and the starting and ending nodes of each edge belong to two different subsets ($\mathcal{X}$, $\mathcal{Y}$, or $\mathcal{Z}$) .

2.4 Matrix representation

Matrices play an important role in graph theory [110, 204, 252, 42, 4]. **Adjacency matrix A** of a n-nodes graph G is a $n \times n$ matrix where off-diagonal elements a_{ij} are the (integer) number of edges from node i to node j. The on-diagonal elements a_{kk} are the number of self-loops from and to node k. i, j, and k are restricted to be $\in [1, \dots, n]$), and elements a_{ij} of **A** are restricted to be ≥ 0
Certain other restrictions may apply:

- If multiple edges (and weights) are forbidden, a_{ij} is restricted to be zero or one, $\forall i$ and $\forall j$
- If self-loops are forbidden, $a_{kk} = 0, \forall k$.
- If the graph is undirected, $\mathbf{A}$ is symmetric.
- If the graph is weighted, then multiple edges are forbidden and a_{ij} (and potentially a_{kk}) are allowed to be non-negative real values.

In addition to providing an algebraic representation of graph G, adjacency matrix $\mathbf{A}$ of G can be used to easily extract information about the underlying graph and its structure. For example, for an unweighted graph, the element in the ith row and jth column of matrix $\mathbf{A}^t$ ($\mathbf{A}$ to the power t) provides the number of paths of length t between node i and node j in G .

The sum of all the elements of matrix $\mathbf{A}$ is called to the **volume** of the graph, $\text{vol}(G) = \sum_{i,j=1}^{n} a_{ij} = a_{\bullet\bullet}$.

Note that an adjacency matrix of a graph G is based on the ordering chosen for the nodes. Hence, there are $n!$ different adjacency matrices for a n-node graph, representing all possible permutations of the node indices.

Notice that in the case of an **undirected bipartite graph**, the adjacency matrix $\mathbf{A}$ is of the form

$$\mathbf{A}^{\text{bi}} = \begin{bmatrix} \mathbf{0}_{\mathcal{X}\times\mathcal{X}} & \mathbf{A}_{\mathcal{X}\times\mathcal{Y}} \\ \mathbf{A}^{\text{T}}_{\mathcal{X}\times\mathcal{Y}} & \mathbf{0}_{\mathcal{Y}\times\mathcal{Y}} \end{bmatrix} \tag{2.1}$$

where $\mathbf{O}_{\mathcal{X}\times\mathcal{X}}$ and $\mathbf{O}_{\mathcal{Y}\times\mathcal{Y}}$ are two matrices full of zeros (indicating that there is no links between elements of the same set ($\mathcal{X}$ or $\mathcal{Y}$)).

In some situations, instead of (or in addition to) affinities, nonnegative costs are assigned to the edges of G. The **cost matrix** is defined accordingly

$$[\mathbf{C}]_{ij} = \begin{cases} c_{ij} & \text{if } i \to j \in \mathcal{E} \\ \infty & \text{otherwise} \end{cases} \quad \text{(for a weighted graph)} \tag{2.2}$$

This number represents the **immediate cost of transition** from nodes i to j. If there is no link between i and j, the cost is assumed to take a large value, denoted by $c_{ij} = \infty$. The **cost matrix C** is an $n \times n$ matrix containing the c_{ij} as elements.

Costs are usually set independently of the adjacency matrix: they are quantifying the cost of a transition according to the problem at hand. Costs can, e.g., be set in function of some properties, or features, of the nodes or the arcs in order to bias the probability distribution of choosing a path. In the case of a social network, we may, for instance, want to bias the paths in function of the education level of the persons, therefore favoring paths visiting highly educated persons (see [94] for details). If there is no reason to introduce a cost, we simply set $c_{ij} = 1$ (paths are penalized by their length) or $c_{ij} = 1/a_{ij}$ (in this case, a_{ij} is viewed as a conductance and c_{ij} as a resistance). Notice that a weight matrix w_{ij} can be sometimes used instead of a_{ij}, if different) – this last setting is used in the experimental section.

Another useful matrix aiming at representing an undirected (or directed) graph and capturing its structure is the $n \times |\mathcal{E}|$ **incidence matrix J**. Each row corresponds to a node of G and each column to an edge of G. However, this representation is not used in this thesis. More information about the incidence matrix can be found in [221, 204, 23].

We finally introduce the $n \times n$ **Laplacian matrix L**, for an undirected graph without self-loop, defined as

$$\mathbf{L} \triangleq \mathbf{D} - \mathbf{A} \tag{2.3}$$

where $\mathbf{D} = \mathbf{Diag}(\mathbf{Ae})$ is the diagonal degree matrix of the graph G containing the $a_{i\bullet} = \sum_{j=1}^{n} a_{ij}$ on its diagonal and $\mathbf{e}$ is a column vector full of ones. $\mathbf{L}$ is positive semi-definite [42, 63]. Another of its properties is that its eigenvalues provide useful information about the connectivity of the graph [63]. The smallest eigenvalue of $\mathbf{L}$ is always equals to zero, and the second smallest one is equal to zero only if the graph is composed of at least two connected components (see Section 2.5). This last value is called the **algebraic connectivity**. Actually, the multiplicity of the zero eigenvalue of the Laplacian matrix is equal to the number of connected components of G [63].

Another important property of the Laplacian matrix is the following: For any vector of values $\mathbf{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$ defined on the nodes,

$$\mathbf{x}^{\mathrm{T}}\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{i,j=1}^{n} a_{ij}(x_i - x_j)^2 = \frac{1}{2}\sum_{i=1}^{n}\sum_{j \in \mathcal{N}(i)} a_{ij}(x_i - x_j)^2 \tag{2.4}$$

is a criterion measuring to what extent the value x_i on each node is close to the values of its neighbors. It is equal to zero (its minimum) when the values x_i are constant. Therefore, this criterion is a structural measure of the **smoothness** of the distribution of the values over the graph. It therefore measures the overall assortativity, or structural (auto)correlation in the graph (the fact that neighboring nodes take similar values).

The properties of $\mathbf{L}$ and its pseudoinverse $\mathbf{L}^{+}$, especially the properties of its eigenvalues and eigenvectors, are of particular importance for analyzing the structure of the underlying graph G. For instance, we already saw that the multiplicity of the zero eigenvalue of the Laplacian matrix is equal to the number of connected components of G [63]. Thus, $\mathbf{L}$ has at least one zero eigenvalue and is therefore rank-deficient.

However, in the Equation (2.4), the sum is run over all the edges which implicitly favors high-degree nodes (hubs) since they appear more frequently in the sum. In order to alleviate this effect, $\mathbf{x}$ can be rescaled, leading to the **normalized Laplacian matrix $\tilde{\mathbf{L}}$** [63, 161, 243]:

$$\tilde{\mathbf{L}} \triangleq \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{2.5}$$

Equation (2.5) immediately follows by substituting $\mathbf{D}^{-1/2}\mathbf{x}$ for $\mathbf{x}$ in Equation (2.4). This shows that the normalized Laplacian matrix is also positive semi-definite.

2.5 Paths and cycles

A **path** $\wp$ [110, 204, 252, 42, 4] (also called a **walk**) in a graph is a sequence of edges in which each successive node is adjacent to its predecessor in the path. A **cycle** or **loop** is a path whose starting node is equal to the ending node. The set of all possible paths of G starting from node i and ending in node j (including possible loops) is denoted $\mathcal{P}_{ij}$, and the set of all t-steps paths of G, starting from node i and ending in node j (including loops), is denoted by $\mathcal{P}_{ij}(t)$. A path between i and j will be denoted by $i \rightsquigarrow j$ or $\wp_{ij}$.

A graph is said to be **connected** (or **strongly connected** if the graph is directed) if there exists at least one path from every node to every other node in the graph. A graph that is not connected is called **disconnected** and consists of a set of **connected components**, which are connected subgraphs.

2.6 The shortest path distance

Considering a pair of nodes, it is really likely that there are several paths between them, and that these paths differ in **length** (number of hops or steps when following the path). The path with lower length is called the **shortest path** between those two nodes. In an unweighted graph, the shortest path distance between two nodes is defined as the (integer) length of minimum-length between them. For weighted graphs, the shortest path refers to the path for which the total cumulated cost (in terms of c_{ij}) along the path is minimal. The **shortest path** (or **lowest cost**) **distance** is then defined as this total. If there is no possible path between two nodes, then the distance between them is considered as infinite (or sometimes undefined). This happens when a graph is not connected, then the distance between at least one pair of nodes is infinite. In an undirected graph, a shortest path $i \rightsquigarrow j$ length is the same as for $j \rightsquigarrow i$.

Computing the shortest path distance in a weighted directed graph is of fundamental importance [90]. The objective here is either to compute the shortest path between two particular nodes or to compute its value between all pairs of nodes and record these as distances in a matrix $\mathbf{D}$ (not to be confused with the degree matrix, also called $\mathbf{D}$). There are many different techniques for computing such a distance matrix, depending on the problem (are there negative weights, compute the k shortest paths, …), the sparsity of the graph,... and the different constraints that have to be taken into account. Let us just mention some well-known algorithms such as Dijkstra's algorithm [204, 252, 4] (one shortest path pair only) and Floyd-Warshall's algorithm [110, 204] (all-pairs shortest path pairs).

The **diameter** of a connected graph is the length of the largest shortest path between any pair of nodes in graph G. In an unweighted graph, the diameter can range from a minimum of 0 (a single isolated node) to a maximum of $n-1$. If a graph is not connected, its diameter is infinite (or undefined) since the shortest path distance between one or more pairs of nodes in a disconnected graph is infinite. The diameter of a graph is important because it quantifies how far apart the farthest two nodes in the graph are.

2.7 Markov chains

We will now review some basic notions of finite Markov chains. The interested reader is invited to consult, among others, the following standard textbooks [53, 111, 141, 185, 205, 231, 235]. This section closely follows the introduction to Markov chains in [164] and [90], but is also largely inspired by [111, 141, 205].

2.7.1 The transition matrix

Consider a weighted directed graph or network G (possibly with loops), strongly connected with a set of n nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$. As already mentioned, the **adjacency matrix** of the graph (see Section 2.4), containing non-negative affinities between nodes, is denoted as $\mathbf{A}$, with elements $a_{ij} \geq 0$. A **Markov chain** is a simple discrete mathematical model representing a special class of dynamic systems (or processes) evolving probabilistically over time (a random process). Its simple matrix representation is easily interpretable in terms of a random walker (see Section 2.7.5) jumping around among a finite set of states (with probabilistic transitions).

Following [164], a n-states finite **Markov-chain process** s is determined by:

- a set of n **states** $\mathcal{S} = \{1, 2, \ldots, n\}$ (corresponding to the nodes of a graph if the Markov chain represents a random walk on a graph, see Section 2.7.5). The system can be in only one state at each time step.
- a set of **transition probabilities** $\{p_{ij}\}_{i,j=1}^{n}$. Each p_{ij} is the probability for the system to evolve from state i at time t to state j at time $t+1$.

In other words, if $s(t)$ is the random variable defining the state of the process at time step t, $p_{ij} = \mathrm{P}(s(t+1) = j|s(t) = i)$. Let us note by $\mathbf{P}$ the matrix containing all transition probabilities p_{ij}, called the **transition probabilities matrix** or simply the **transition matrix**. By construction, all entries of $\mathbf{P}$ are nonnegative and each of its rows sums to one: the probability that it goes somewhere during the next step must be one and the transition matrix is therefore **stochastic**.

- an **initial** (or **starting**) **state** $s(0)$ or an **initial probability distribution** on states $\mathrm{P}(s(0) = i), i = 1, \ldots, n$.

A Markov chain process therefore assumes that if the process is in state i, there is a fixed probability p_{ij} that it will next be in state j, independently of t, or

$$\mathrm{P}\left(s(t+1) = j|s(t) = i, s(t-1) = i_{t-1}, ..., s(1) = i_1, s(0) = i_0\right) = p_{ij} \tag{2.6}$$

for all states $i_0, i_1, ..., i_{t-1}, i, j$ and all $t \geq 0$. Equation (2.6) assumes that the conditional distribution of any future state $s(t+1)$ given the past states $s(0), s(1), ..., s(t-1)$ and the present state $s(t)$, is independent of the past states and thus depends only on the present state (this is the Markov property).

2.7.2 The multistep transition matrix

In addition to the (one-step) transition probability p_{ij}, the τ-step transition probability $p_{ij}^{(\tau)}$ can also be defined: the probability that a process in state i will be in state j after τ transitions (see [205, 164] for more details). That is,

$$p_{ij}^{(\tau)} = \mathrm{P}(s(t+\tau) = j|s(t) = i), \text{with } \tau \geq 1 \tag{2.7}$$

It can be shown [205, 164] that the transition probability of jumping from state i to state j in exactly τ time steps is equal to $[\mathbf{P}^{\tau}]_{ij}$. The τ-step transition matrix is therefore $\mathbf{P}^{\tau}$ (i.e. $\mathbf{P}$ to the power τ).

2.7.3 Some properties of states and Markov chains

The first property is closely related to the notion of connected components in a graph G. A Markov chain is called an **irreducible** chain if all states can be reached starting from every state (not necessarily in one move) [90]. A state j can be reached [164] starting from state i if, when starting in i, it is possible that the process will ever enter state j; i.e. if $p_{ij}^{(t)} > 0$ for some $t \geq 0$. This property is not symmetric because j could be reached starting from i while i could not be reached starting from j.

In the sequel, we will assume that Markov chains are irreducible, unless clearly stated.

If an irreducible Markov chain is **aperiodic** [90] then $p_{ij}^{(t)} > 0$, $\forall i, j$, for $t > T$: Every state can be reached from any other state in exactly t steps. In other words, if an irreducible Markov chain is aperiodic then all the elements of the t-step transition matrix (for $t > T$) are strictly positive. Such a Markov chain is also called **regular**.

Bipartite Markov chains and, more generally, **periodic chains** containing disjoint sets of states (with no transitions between states belonging to the same set) do not match this definition [90]. In periodic Markov chains, the process "oscillates" from one set of states to another and therefore does not converge to a stationary distribution.

A state α of a Markov chain is **absorbing** if it is impossible to leave it [111] (i.e. $p_{\alpha\alpha} = 1$ an the rest of the row are zeros). If the process has reached absorbing state α, it will never escape it.

Moreover, a Markov chain is **absorbing** if it contains at least one absorbing state and if it is possible to reach at least an absorbing state from every state of the Markov chain (not necessarily in one step). Since the process cannot escape from an absorbing state, it will eventually end up in one of the absorbing states. In an absorbing Markov chain, a state which is not absorbing is called **transient**. Absorbing Markov chains are a very useful concept and is investigated in Chapter 9.

2.7.4 Killed Markov chains.

Here, Contrary to what Section 2.7.1 stated, $\sum_{j=1}^{n} p_{ij}$ can be < 1 for some states. The underlying stochastic process is then called **killed** or evaporating.

In a killed random walk, the process has a non-zero probability of being killed in some states [220]. In this case, we have $\sum_{j=1}^{n} p_{ij} < 1$ for these states, which will be called **killing** (sometimes denoted as evaporating), and **absorbing** if $\sum_{j=1}^{n} p_{ij} = 0$ (the process stops when reaching i) [90]. This situation is similar to a standard finite Markov chain where the chain has a supplementary absorbing state in which the process can jump, with given probabilities, from the killing nodes. A killed random walk contains at least a **killing Markov state** [90]. The transition matrix of a killed random walk is **sub-stochastic** (all row sums are less than or equal to one and at least one is lesser than one). Thus, when visiting a killing state, the process has a non-zero probability of jumping to a "cemetery" state, in which case it can be considered as killed, or dead. A process containing killing states is called a **killed random walk**: the probability of finding the process in any state (aside from the fictive "cemetry" state) is decreasing over time (while for a standard Markov chain, it remains equal to one).

Moreover, An **absorbing, killing, Markov** state α verifies $p_{\alpha j} = 0, \forall j$ [90]. In this case, the process stops when α is reached.

2.7.5 Defining a random-walk model on a graph

Following [90], we now associate a state of the Markov chain to each node of the graph G. Remember that the random variable $s(t)$ is the state of the Markov model at time t, if the random walker is in node i on G at time t, then $s(t) = i$.

Then, the random walker will follow the (single-step) transition probabilities:

$$p_{ij} = \mathrm{P}(s(t+1) = j | s(t) = i) = \frac{a_{ij}}{a_{i\bullet}}, \text{ where } a_{i\bullet} = \sum_{j=1}^{n} a_{ij} \tag{2.8}$$

The Markov chain describing the sequence of nodes visited by a random walker on a graph G is called a **random walk** on G.

As before, the transition probabilities only depend on the current state and not on the past ones (memoryless first-order Markov chain). It means that the random walker chooses to follow the next edge at random, but favors edges with a high affinity a_{ij}. In matrix form, it reads:

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{A} \tag{2.9}$$

where $\mathbf{P}$ is the transition matrix and $\mathbf{D}$ is the diagonal degree matrix containing the degrees of the nodes on its diagonal.

Let us now compute the following probability for the random walk to be in the different states, according to $\mathbf{P}$ and a starting distribution. Let $x_i(t) = \mathrm{P}(s(t) = i)$ be the probability that the random walker is in state i at time t. The Markov chain process initial condition is given by:

$$x_i(0) = \mathrm{P}(s(0) = i) \tag{2.10}$$

It becomes, after one iteration:

$$\mathrm{P}(s(1) = i) = \sum_{j=1}^{n} p_{ji} \mathrm{P}(s(0) = j) \tag{2.11}$$

And, for the next steps:

$$\mathrm{P}(s(t+1) = i) = \sum_{j=1}^{n} p_{ji} \mathrm{P}(s(t) = j) \tag{2.12}$$

Or, in matrix form,

$$\mathbf{x}(t+1) = \mathbf{P}^{\mathrm{T}}\mathbf{x}(t) \tag{2.13}$$

where $\mathbf{x}(t)$ is a column vector containing the $\mathrm{P}(s(t) = i)$ for all states $i = [1, ..., n]$.

Recurrence Equation (2.13) can be solved easily. At first time step, we have $\mathbf{x}(1) = \mathbf{P}^{\mathrm{T}}\mathbf{x}_0$, then $\mathbf{x}(2) = \mathbf{P}^{\mathrm{T}}\mathbf{x}(1) = \mathbf{P}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{x}_0 = (\mathbf{P}^2)^{\mathrm{T}}\mathbf{x}_0$ and, more generally, at time step t,

$$\mathbf{x}(t) = (\mathbf{P}^{\mathrm{T}})^t\mathbf{x}_0 = (\mathbf{P}^t)^{\mathrm{T}}\mathbf{x}_0 \tag{2.14}$$

Therefore, for an initial distribution $\mathbf{x}_0$ at $t = 0$, the probability distribution of being in each state at time t can be computed for a given random walk on G.

In this thesis, we assume that graphs are strongly connected, and that Markov chains are irreducible (see Section 2.7.3), unless clearly stated. If this is not the case, each connected component can be studied one by one, so that each associated Markov chain is irreducible.

2.7.6 The stationary distribution of a regular Markov chain

For a **regular** (both irreducible and aperiodic) Markov chain, it is well-known [164, 185] that the limit

$$\pi_j = \lim_{t\to\infty} p_{ij}^{(t)} = \lim_{t\to\infty} [\mathbf{P}^t]_{ij} \tag{2.15}$$

exists and is independent of the initial state i. Thus, there is a limiting probability distribution that the process will be in a certain state after a large number of transitions, and this value is independent of the initial state. This limiting probability distribution π_j is also called **stationary distribution** or **equilibrium distribution** and is the probability of finding the process in each state j as time $t \to \infty$.

It can further be shown that π_j [164] is equal to the long-run proportion of visits to state j. The stationary distribution can therefore be interpreted as the probability of finding the process in state $s = j$ in the long-run behavior.

Moreover, Equation (2.13) can be rewritten assuming that the probability distribution converged to a stationary value (i.e. if the Markov chain is regular): $\lim_{t\to\infty} \mathbf{x}(t) = \boldsymbol{\pi}$. Then,

$$\boldsymbol{\pi} = \mathbf{P}^{\mathrm{T}}\boldsymbol{\pi} \tag{2.16}$$

The stationary distribution can therefore be obtained by computing the normalized eigenvector of the transition matrix $\mathbf{P}^{\mathrm{T}}$ associated to eigenvalue 1.

2.7.7 The fundamental matrix of a killed random walk

As introduced in Section 2.7.4, the transition matrix of a killed random walk is sub-stochastic. This defines a **killed Markov chain**.

An interesting related measure is the expected number of visits to each state before being killed when starting from a state i. This quantity leads to the fundamental matrix of the killed (or absorbing) Markov chain, and plays a fundamental role in finite Markov chain theory [53, 111, 141, 185, 205, 231, 235].

Assuming the initial state of the Markov process is node i: In this case, $\mathbf{x}_0 = \mathbf{e}_i$. Here, $\mathbf{e}_i$ is a column vector full of 0's except at position i where it is equal to 1. From Equation (2.14), the probability distribution after t step is $\mathbf{x}(t) = (\mathbf{P}^t)^\mathrm{T}\mathbf{e}_i$. Then, for computing the **expected number of visits** (starting from node i), $\mathbf{n}_i$, of each state before being killed, we sum the probability distribution (2.14) over all time step:

$$\begin{aligned}\mathbf{n}_i &= \sum_{t=0}^{\infty}\mathbf{x}(t) = \sum_{t=0}^{\infty}(\mathbf{P}^t)^\mathrm{T}\mathbf{e}_i \\ &= \Big(\sum_{t=0}^{\infty}\mathbf{P}^t\Big)^\mathrm{T}\mathbf{e}_i = \big((\mathbf{I}-\mathbf{P})^{-1}\big)^\mathrm{T}\mathbf{e}_i \\ &= \big(\mathbf{e}_i^\mathrm{T}(\mathbf{I}-\mathbf{P})^{-1}\big)^\mathrm{T} \qquad (2.17)\end{aligned}$$

If $\mathbf{P}$ is stochastic and that the associated Markov process is irreducible and aperiodic, this series does not converge as the spectral radius of $\mathbf{P}$ is 1. Now that the transition matrix is sub-stochastic (and non-negative), the spectral radius is less than 1 and the series converges [174]. Notice that when the matrix $\mathbf{P}$ is stochastic (as for standard Markov chain), $(\mathbf{I}-\mathbf{P})$ is not of full rank and the inverse of $(\mathbf{I}-\mathbf{P})$ is therefore not defined.

The full **fundamental matrix** of the killed random walk on G is defined as:

$$\mathbf{N} = \begin{bmatrix}\mathbf{n}_1^\mathrm{T}\\ \mathbf{n}_2^\mathrm{T}\\ \vdots \\ \mathbf{n}_n^\mathrm{T}\end{bmatrix} = (\mathbf{I}-\mathbf{P})^{-1} \qquad (2.18)$$

In this matrix, the expected number of visits to state j before being killed or absorbed, starting from state i can be found on entry i, j.

2.7.8 The commute-time and commute-cost distance

We now assume a random walk on a directed weighted graph, and that each node i is reachable starting from each node j. The **average commute-time** $n(i, j)$ between node i and node j is the number of steps needed to reach node j from i for the first time and then go back to starting node i [90]:

$$n(i, j) \triangleq m(i, j) + m(j, i) \tag{2.19}$$

where $m(i, j)$ is the expected number of step needed for visiting node j for the first time starting from a transient (non-absorbing) node i. Notice that, while $n(i, j)$ is symmetric by definition, $m(i, j)$ is not.

The average commute-time is a legit **distance measure** on graph G [108] and will therefore be referred to as the **commute-time distance**. Moreover, it has been shown that the average commute-time is proportional to the effective resistance between the two nodes [57, 90].

An interesting property is that the commute-time distance between two points is decreasing when the number of paths connecting the two points increases and when the length of one of these paths decreases [78, 90]. See Chapter 4 for a deeper discussion about this fact.

If costs are associated to edges, the **average commute-cost** can also be computed, considering the total cost of the paths instead of their lengths. However [144] showed that this quantity is proportional to the average commute-time for a given graph. It is therefore redundant with the average commute-time [90].

2.8 Graph generators

As the name states, graph generators are procedures used to generate (artificial) graphs. They can be used to study properties of typical graphs, or reproduce diverse types of complex networks behavior.

Two common graph generators are now introduced [24, 43], and will be used in Chapter 7.

2.8.1 Erdős-Rényi (ER) Graph Generator.

This model is also called the Poisson random graph generator because it generates a random graph with a **Poisson node degree distribution** [43]. This type of graph is often used to study theoretical properties and behavior of networks [183]. The model generates a random adjacency matrix $\mathbf{A}$ for the graph and only requires a probability parameter $p \in]0, 1]$. The implementation [106] first generates an upper triangular random matrix (zeros on diagonal, entries being $\in]0,1]$), then each entry of the matrix is replaced by a 0 if the entry is smaller than p, and 1 otherwise. This matrix $\mathbf{R}$ is then symmetrized using $\mathbf{A} = \mathbf{R} + \mathbf{R}^{\mathrm{T}}$. Of course, various variants of this procedure exist. For our experiments in this thesis, p was set to a random value for each graph, with $p \in]0, 1/2]$.

2.8.2 Albert-Barabási (AB) Graph Generator.

The model generates a random graph with a **power law degree distribution** [24, 25]. This kind of network is often observed in natural and human-generated systems, including the world wide web, citation networks, and social networks [183] (often called scale-free networks).

An integer parameter m is required for generating such a graph. The implementation [128] begins with an initial connected network of $m + 1$ nodes. Then, new nodes are added to the network, one at a time. Each new node is connected to m existing nodes with a probability that is proportional to the current degree of each node. The procedure stops when the desired number of nodes is reached.

Here again, various variants of this procedure exist. With this model, heavily linked nodes (named **hubs**) tend to quickly accumulate even more edges: the new nodes have a preference to attach themselves to these already heavily linked nodes (this is called a **preferential attachment**). For our experiments, p was set to a random value for each graph with $m \in \{1, 2, 3, 4, 5, 6\}$. Many natural networks in real life often behave like AB graphs (see for example [9] and citations inside).

Chapter 3

Semi-supervised Learning

This chapter briefly introduces **semi-supervised learning** and in particular graph-based semi-supervised learning. As the name suggests, the semi-supervised learning paradigm tries to combine the best of the **supervised** and the **unsupervised learning** worlds, those two fundamental tasks being very common in machine learning. This chapter is inspired by the first chapter of [268], which is a nice introductory reference, by [58] and by [90].

Historically, the oldest semi-supervised research work was related to the heuristics known as self-learning (or self-training, or self-labeling) [58] with for instance [213, 93, 5] in the mid-1960s. Self-learning repeatedly uses a supervised learning method: In each step a part of unlabeled data is labeled according to the current decision function, then the supervised method is retrained using its own prediction as additional labeled points [58].

Semi-supervised learning has become really popular in the early 2000s and its applications cover a large range of application such as social networks analysis, categorization of linked documents (e.g. patents or scientific papers), or protein function prediction, to name a few. For a review of recent work, see the survey article [266].

This chapter contains a reminder about supervised and unsupervised learning in Section 3.1, then we focus on semi-supervised learning in Section 3.2. The difference between transductive and inductive learning is also discussed, in Chapter 3.3. Finally, this learning scheme is extended to the case of graph-based classification in Section 3.4.

3.1 Supervised and Unsupervised Learning

Let $\mathbf{X}$ be the $n_s \times n_f$ training set data matrix containing n_s samples on its rows and n_f features on its columns. Typically it is assumed that samples are drawn independently and identically distributed (also known as i.i.d.) from a common distribution $\mathcal{X}$. The set of features is also called features (or input) space. Entries of $\mathbf{X}$ will be referenced by x_{ij}, the value of the jth feature for the

ith sample. Furthermore, $\mathbf{y}$ will be the observed column vector containing the target values, with its entries being noted y_i. For classification, those values are also called labels, are ,e.g., coded as integers between 1 and m (the number of classes), and represent the class membership of each observation. In this section, the entries of $\mathbf{y}$ can also be continuous in the case of regression. Notice that the ith row of $\mathbf{X}$ is associated to the ith sample and to the target value y_i.

3.1.1 Unsupervised Learning

In **unsupervised learning**, only $\mathbf{X}$ is used to fit the model. There is no supervision coming from $\mathbf{y}$, no indications about how samples should be grouped. Samples are said to be unlabeled. The goal of unsupervised learning is to find interesting structures in data $\mathbf{X}$ [58]. Clustering and outlier/anomaly detection are typical unsupervised learning tasks.

The goal of **clustering** [2, 58, 268] is to regroup a set of samples in a certain number of mutually exclusive subsets (or groups, clusters) such that samples in the same group are more similar to each other than to those in other groups, according to a given criterion.

Anomaly detection is the identification of samples which do not fit the expected pattern from a dataset. In other words, this task is about identifying the "normal" behavior of a dataset, and samples that do not exhibit this behavior.

3.1.2 Supervised Learning

Here, both $\mathbf{X}$ and $\mathbf{y}$ are used to train the model and are therefore called training data. The goal is to learn a mapping from $\mathbf{X}$ to $\mathbf{y}$ [58]. Training data are said to be labeled since all entries of $\mathbf{y}$ are (and must be) known for the training set. The model learns from those data and produces an inference function, which is then used to predict information (basically the class for classification) about new samples, also called **test data**. If the $\mathbf{y}_i$'s are discrete, those values are called classes and the supervised learning task is called **classification** [2, 58, 268]. If, on the contrary, continuous number of values is possible for the $\mathbf{y}_i$'s, the supervised learning task is called a **regression** [2, 58, 268]. In both cases, an error function can be defined to quantify the difference between the groundthrust target values and the predicted outputs of the supervised model.

In the case of classification, the model is often called a classifier. The goal of a classification task is to automatically assign data to predefined discrete classes (or categories). The most used performance metric for classification is the miss-classification rate. It is simply equal to the number of incorrectly classified samples in the test set divided by the number of samples in this set. This error rate should be minimum. For particular tasks, miss-classification rate must be put aside and more complex metrics should be used (as an illustrative example, see Chapter 8). As an example, consider two classifiers:

- Classifier 1 correctly predicts 5 samples of class 1 (out of 10) and 95 samples of class 2 (out of 990). The classification rate is therefore 10%.
- Classifier 2 correctly predicts 0 samples of class 1 (out of 10) and 990 samples of class 2 (out of 990). The classification rate is therefore 99%.

Classifier 2 actually always predicts class 2 and is useless to predict class 1, contrary to classifier 1, despite its higher classification rate.

3.2 Semi-supervised Learning

Now imagine that some of the $\mathbf{y}_i$'s are not known for all i but that we still want to use all data at our disposal. Reasons can be multiple, but intuitively, labeled samples are often more costly (in terms of cost, effort, or time) to gather than unlabeled samples:

- Protein labels can take months of laboratory work to be obtained.
- In spam filtering, only a few spams are reported by users.
- In image classification, millions of unlabeled pictures are available on internet, but a human annotator is required to label those pictures. Therefore, this task is time-consuming and can be highly subjective.

Semi-supervised learning therefore aims to understand how labeled and unlabeled data may change the learning behavior and to design algorithms that take advantage of such combination [218]. The goal is to provide a prediction based on both labeled and unlabeled samples.

In practice, nothing ensures that semi-supervised learning will always be superior to supervised or unsupervised learning. On the other hand, exploiting more information intuitively produces a better model (i.e. show better performance according to a given metric). By the way, this is not always the case, see as an example Chapter 6.

Semi-supervised learning tends to be most useful whenever there are more unlabeled data than labeled ones [2, 58, 268]: since unlabeled data carry less information than labeled ones, they are required in large number to increase prediction accuracy significantly [58]. It is also required that unlabeled samples bring useful information to the learning process. Actually, semi-supervised learning usually makes some assumptions about the learning process and works better when those assumptions are met. The most common assumptions are presented in Section 3.2.1.

We won't make a comprehensive description of semi-supervised learning algorithms in this chapter: many are presented in the Chapters 5 and 6. But the main trend is that most of semi-supervised learning models are either based on an extension of (fully) unsupervised learning (such as constrained clustering, described in Section 3.2.2) or supervised learning (such as semi-supervised classification and regression, described in Section 3.2.3).

3.2.1 Some common assumptions

This section presents three common assumptions often made in semi-supervised learning.

The smoothness/continuity/consistency assumption. This central hypothesis can be stated as "Samples which are close to each other are more likely to share the same value" (but then "close" still has to be defined...) [2, 58]. By using value, it means that it holds both for classification and regression. For classification, it means that close samples tend to belong to the same class (see for instance the case of nearest neighbors). This is also generally assumed in supervised learning to ensure geometrically simple decision boundaries but for semi-supervised learning, decision boundaries are favored to lie in low-density regions [58].

Figure 3.1 illustrates this assumption, showing how the consistency assumption can be used to tackle the lack of labels. This figure is called **two moons** [117, 107] and is a well known toy dataset generated according to a pattern of two intertwining moons. Every point should be similar to points in its local neighborhood, and furthermore, points in one moon should be more similar to each other than to points in the other moon [265]. Let us now imagine that we only know two labels, represented by the blue circle and the red circle. The classification result given by a fully supervised Support Vector Machine (SVM) is given in the central sub-figure. According to semi-supervised learning and the consistency assumption, however, the two moons should be classified as shown on the right.

The cluster assumption. This hypothesis states that samples sit in clusters (see clustering in Section 3.1.1) and that those clusters give information about the labels [2, 58]. In other words, samples from the same cluster are more likely to share the same label (or value for regression). The smoothness/continuity assumption and the cluster assumption are related but not interchangeable: actually, the cluster assumption can be seen as a special case of the smoothness assumption. Note that the cluster assumption does not imply that each class forms a single, compact cluster: it only means that, usually, we do not observe samples of two different classes in the same cluster [58].

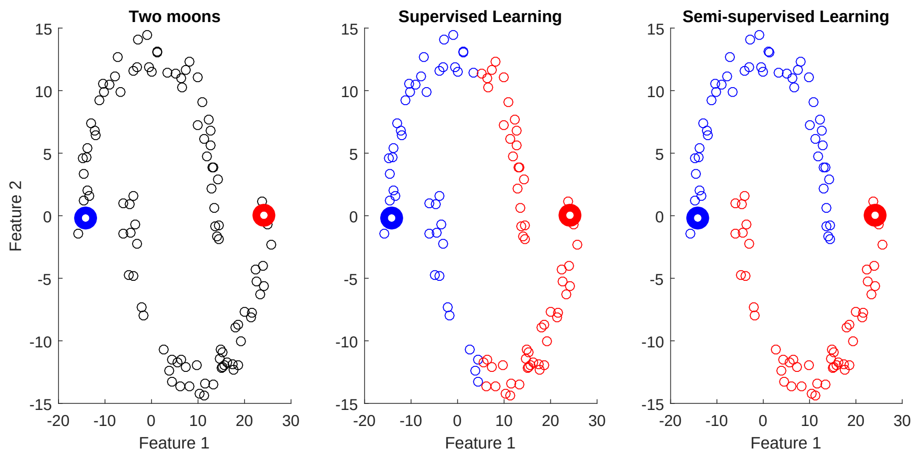


FIGURE 3.1: On the left sub-figure, we only know two labels, represented by the blue circle and the red circle. The center part is the classification resulting from a fully supervised SVM. Semi-supervised learning and the consistency assumption, however, classified the two moons as shown on the right part.

The manifold assumption. Another well known hypothesis is that the samples lie approximately on a manifold of (much) lower dimension than the input space. Then the learning process is about "learning this manifold", or distances and densities defined on this manifold [2, 58].

In a non-semi-supervised scheme, this is often used to avoid the well-known phenomenon called **curse of dimensionality** [39]. It is related to the fact that, in the input space, the volume grows exponentially with the number of dimensions and an exponential growing number of samples are required for statistical models such as reliable density estimation [58]. Another consequence is that pairwise distances between samples tend to become more similar (thus less meaningful) when the number of dimensions rises.

Computing a distance between nodes on a graph is closely related to the computation of a geodesic distance on a (low dimension) manifold [58]. Therefore, some methods consider the transformation of raw data **X** into a graph as a dimensionality reduction technique [76, 90]. If data lie on multiple manifolds however, this assumption is not met and disconnected graphs can help to correctly model the process [218]. This transformation, although interesting, is not discussed in this thesis.

3.2.2 Constrained clustering

Constrained clustering can be considered as an extension of unsupervised clustering. Constrained clustering [245, 268] learns from $\mathbf{X}$ (as regular, fully unsupervised clustering) and from some scarce supervised information about the clusters: usually it takes the form of a must-link and/or a cannot-link $n_s \times n_s$ matrix indicating on entry (i, j) if samples i and j must (or cannot, respectively) be present in the same cluster [268]. The goal of constrained clustering is to improve the quality of the partition compared to unsupervised clustering, which uses unlabeled data only.

3.2.3 Semi-supervised classification and regression

Here, the extension comes from the fact that the classifier uses both labeled and unlabeled data [2, 58, 268, 90]. There are often more unlabeled data than labeled one (since they are less costly). The goal is to improve (according to some metric) the classification performance in the semi-supervised case compared to the supervised one [268]. The same is valid for regression except that the target variable is continuous.

To be able to use unlabeled data in the learning of a model where all is about target $\mathbf{y}$, some hypothesis about the underlying distribution of data are often required [2, 58]. The most frequent were presented in Section 3.2.1. Notice that those hypotheses are not exclusive to semi-supervised learning, but at least one of them must be present to bridge the gap between unlabeled and labeled data. Once this gap is filled, another distinction can be made: will the model give a prediction for the unlabeled samples only ? Or will it be generalizable to new (unseen) samples ? This distinction is depicted in Section 3.3.

3.3 Transductive/Inductive Learning

Given data matrix $\mathbf{X}$ (and corresponding $\mathbf{y}$), let us now define $\mathbf{X}_l$ and $\mathbf{X}_u$ (and corresponding respectively to $\mathbf{y}_l$ and $\mathbf{y}_u$) the sets of labeled and unlabeled samples, respectively. A third set is also introduced: $\mathbf{X}_n$ and corresponding $\mathbf{y}_n$, another set of unlabeled samples we want to predict, but which are not involved in the model creation. This last set can be used to estimate the generalization power of the model, if $\mathbf{y}_n$ is known. Notice that $\mathbf{X}_l$, $\mathbf{X}_u$ and $\mathbf{X}_n$ are all generated by the same distribution $\mathcal{X}$. Using those notations, two learning schemes can be characterized:

- **Transductive scheme**: The model aims to predict only the target variable for the unlabeled samples $\mathbf{X_u}$ (which can be all or part of the test samples in this framework) i.e. the column vector $\mathbf{y_u}$ [268, 222]. This learning framework was formally introduced by Vapnik in the 1970s [242].

- **Inductive scheme**: The model aims to predict the target variable for future samples $\mathbf{X}_n$ (the test samples) which were not already present in the training data (neither from $\mathbf{X}_l$ nor from $\mathbf{X}_u$), or for any new samples further generated from $\mathcal{X}$. The inductive scheme is therefore a kind of generalization based on samples $\mathbf{X}_l$ (and $\mathbf{X}_u$ in the case of semi-supervised learning) [268, 222].

Notice that induction is somehow a more difficult problem than transduction. For a given test set, transduction is more direct as $\mathbf{X}_u$ is simply labeled by the model. On the other hand, induction requires to first build a general model for all the feature space, and then return the particular value for the $\mathbf{X}_u$. This discussion is a keystone of Vapnik's work on transductive learning [242]. In contrast to inductive learning, in transductive learning, no general decision rules are inferred, but only the labels of the unlabeled samples are predicted [58].

In practice, this does not ensure that one of the schemes will always be superior to the other for a particular dataset or application and both approaches must be considered case-by-case (see Chapter 6 as an example).

3.4 Graph-based semi-supervised classification

We will now discuss the case where data take the form of a graph (see Chapter 2 for a reminder about graphs). This section is inspired by the discussion of semi-supervised learning on graphs appearing in Amin Mantrach's PhD thesis [166].

Here, the feature space is replaced by an adjacency matrix $\mathbf{A}$ (see Chapters 5 for an application), or added to an adjacency matrix (see Chapter 6). Samples $\mathbf{X}_l$ and $\mathbf{X}_u$ will be replaced by (or correspond to) the sets of nodes $\mathcal{V}_l$ and $\mathcal{V}_u$, respectively the subset of labeled and unlabeled nodes. Then, the graph G (described by its adjacency matrix $\mathbf{A}$) is considered as the natural structure of the data (for example, facebook account linked by friendship links), or constructed from plain, raw, vector-based data using network formation strategies (see for example Chapter 4 of [218]).

Then the learning process consists in assigning a label for every unlabeled node forming a test set: Graph-based semi-supervised classification is sometimes called within-network classification and falls into the **semi-supervised classification** paradigm [2, 58, 266, 268] (see Section 3.2.3). The state of the art of this field can be found in Section 5.1. For a comprehensive survey of the topic see, e.g., [266, 268].

Graph-based semi-supervised algorithms often make the same assumption as regular, usually transductive, semi-supervised learning (see Section 3.2.1). The game is then to translate the different concepts from the vector-based world to the graph world. For example the Euclidean distance can become the shortest path distance for graphs. The most common assumption is that neighboring nodes are likely to belong to the same class and thus to share the same class label. In other words, we assume that the dependent variable (class label) is **structurally correlated** (autocorrelation, see Section 3.2.1). In particular, as in spatial statistics [40, 68, 113, 192], it is commonly assumed that the correlation between the values of the dependent variable on two nodes of the network depends on the distance between these two nodes (spatial correlation). The shortest path distance, or the minimal number of transitions between the two nodes, is often used as a substitute for the Euclidean distance.

This local consistency hypothesis (also called homophily, or structural autocorrelation [265]) can be tested by some standard spatial statistics tests [40, 68, 113, 192]. Two of these tests (Moran's I and Geary's c) are described in Section 6.3.1. This analysis is also used in Section 6.4.4.

Graph-based semi-supervised classification has received a growing focus in recent years [2, 58, 266, 268], see also Sections 5.1 and 6.2. This problem has numerous applications such as classification of individuals in social networks, categorization of linked documents (e.g. patents or scientific papers), or protein function prediction, to name a few. In this kind of application (as in many others), unlabeled data are usually available in large quantities and are easy to collect: friendship links can be recorded on Facebook, text documents can be crawled from the internet and DNA sequences of proteins are readily available from gene databases.

This chapter will now be concluded by some specific notation used when dealing with graph-based semi-supervised classification.

3.4.1 Class membership notations

When tackling the **graph-based semi-supervised classification** problem, a set of mutually exclusive classes is considered, $\{\mathcal{C}_k\}_{k=1}^{m}$, with the number of classes equal to m. Each node is assumed to belong to at most one class since the class label can also be unknown. To represent the class memberships, a $n \times m$-dimensional indicator matrix $\mathbf{Y}$, computed from $\mathbf{y}$, is used. On each of its rows, it contains, as entries, a 1 in column c when the corresponding node belongs to class c, and 0 otherwise (m zeros on line i if the node i is unlabeled). The cth column of $\mathbf{Y}$ will be denoted $\mathbf{y}^c$ and contains the binary memberships of the nodes to class c, and 0 if unlabeled.

Chapter 4

The bag-of-paths Framework

This section introduces the **bag-of-paths** (BoP) theoretical background and notations. The bag-of-paths model was recently introduced in [94] and three of its applications are discussed in this thesis: the semi-supervised classification through the bag-of-paths group betweenness (see Chapter 5), the bag-of-paths node criticality measure (see Chapter 7) and the constrained randomized shortest path problem (the RSP framework is really close to the bag-of-paths framework, see Chapter 9).

The motivation to develop this framework is the following: While relevant in many applications, simple distances as the shortest path distance (see Section 2.6) and the commute-time distance (see Section 2.7.8) cannot always be considered as a good candidate in network data analysis for multiple reasons [94]:

- The shortest path distance only depends on the shortest paths between those nodes and does not integrate the **degree of connectivity** between them. In many applications however, nodes connected by many indirect paths should be considered as "closer" than nodes connected by only a few paths.
- When computing the distance from or to a given node, the shortest path distance usually provides many ties, especially in unweighted, undirected, graphs. Considering the amount of connectivity (or other properties) can help breaking these ties.
- When the size of the graph increases, the resistance and the commute-time distances (both based on a random walker) converge to a useless value, only depending on the degrees of the two nodes. [244] depicts this phenomenon as the random walker getting "lost in space".
- Those two distances assume completely random movements or communication in the network, which is also unrealistic.

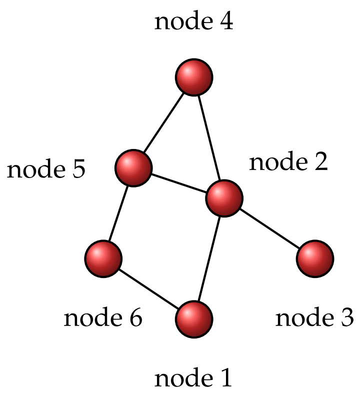


FIGURE 4.1: A small toy graph. According to the bag-of-path model, the probability to pick a path from node i to j is given by the entries of matrix $\mathbf{Z}$. Here, The most probable pair is node 2 to 2 (with probability 11.52%) and the lowest probable pair is node 6 to 3 (with probability 6.87%). For this example θ was set to 1.

To summarize, the shortest path distance fails to take the whole structure of the graph into account [94], whereas random walks quickly loose the notion of proximity to the initial node when the graph becomes larger [244].

The proposed bag-of-paths framework generalizes the shortest path and the commute-time distances by computing an intermediate distance, depending on a temperature parameter T. When T is close to zero, the distance reduces to the standard shortest path distance (emphasizing exploitation) while for $T \to \infty$, it reduces to the commute-cost distance (focusing on exploration) [94]. The resulting framework therefore defines a **family of distances** interpolating between the shortest path and the commute-time distance. Doing so, the framework take into account the advantages of both accessibility between nodes (or proximity) in the network and amount of connectivity.

As a consequence and in practice, the bag-of-paths framework is especially relevant when considering relatedness of nodes based on communication, movement, ..., in a network which does not always happen optimally, nor completely randomly [94]. Similar ideas appeared at the same time in [60], based on considering the co-occurrences of nodes in forests of a graph or walks on the graph, and in [8] and [119], based on a generalization of the effective resistance in electric circuits [94].

The main point of the bag-of-path framework is to compute the probability of picking a path starting in node i and ending in node j, taking into account the infinite set of path. These probabilities corresponds to the z_{ij} of the fundamental matrix **Z** (see Equation 4.9, appearing later). A toy example can be found on Figure 4.1.

The rest of this chapter is divided as follows: Section 4.1 summarizes bag-of-paths model. In addition, the bag-of-hitting-paths model is described in Section 4.2. Notice that those models are used in Chapter 5, Chapter 7, and Chapter 9.

4.1 The bag-of-paths framework

This framework was originally introduced in the context of distance computation on graphs [94], capturing its global structure with, as building block, network paths. This same idea was previously used in [167] for defining a covariance kernel on a graph. We will briefly review the whole framework in this section (see [94] for details). The bag-of-paths model can be considered as a motif-based model [175, 11] using, as building block, paths of the network. In the next section, hitting paths will be used instead, as motifs.

As you may remember from Section 2.5, a path $\wp$ (sometimes called a walk) is a sequence of transitions to adjacent nodes on G (loops are allowed), initiated from a starting node s, and stopping in an ending node e. If we want to emphasize on those starting and ending nodes, we will use $\wp_{se}$ instead of $\wp$.

The BoP framework is based on the probability of picking a path $i \rightsquigarrow j$ starting at a node i and ending in a node j from a virtual bag containing all possible paths in the network [94]. Let us define $\mathcal{P}_{ij}$ as the set of all possible paths connecting node i to node j, including loops. We further define the set of all paths through the graph as $\mathcal{P} = \bigcup_{i,j=1}^{n} \mathcal{P}_{ij}$. The **total cost** of a path $\wp$, $\tilde{c}(\wp)$, is defined as the sum of the individual transition costs c_{ij} along $\wp$.

The potentially infinite set of paths in the graph is enumerated and a probability distribution is assigned to the set of individual paths $\mathcal{P}$. This probability distribution P represents the probability of picking a path $\wp \in \mathcal{P}$ in the bag, and is defined as minimizing the total expected cost, $\mathbb{E}\left[\tilde{c}(\wp)\right]$, among all distributions with a fixed relative entropy J_0. The relative entropy is computed with respect to a reference distribution: we chose the natural random walk on the graph (see Equation (2.8)), as in [94].

This choice naturally defines a probability distribution on the set of paths such that long (high cost) paths occur with a low probability while short (low cost) paths occur with a high probability (see [94]).

In other words, we are seeking path probabilities, $\mathrm{P}(\wp), \wp \in \mathcal{P}$, minimizing the total expected cost subject to a constant relative entropy constraint:

$$\left|\begin{array}{ll} \underset{\{\mathrm{P}(\wp)\}}{\text{minimize}} & \sum\limits_{\wp \in \mathcal{P}} \mathrm{P}(\wp)\, \tilde{c}(\wp) \\ \text{subject to} & \sum_{\wp \in \mathcal{P}} \mathrm{P}(\wp) \ln(\mathrm{P}(\wp)/\tilde{\mathrm{P}}^{\mathrm{ref}}(\wp)) = J_0 \\ & \sum_{\wp \in \mathcal{P}} \mathrm{P}(\wp) = 1 \end{array}\right. \tag{4.1}$$

where $\tilde{\mathrm{P}}^{\mathrm{ref}}(\wp)$ represents the probability of following the path $\wp$ and $\tilde{\pi}^{\mathrm{ref}}(\wp)$ is the likelihood of this path when walking according to the natural random walk reference distribution (the product of the transition probabilities defined by Equation (2.8) along the path). More precisely, $\tilde{\mathrm{P}}^{\mathrm{ref}}(\wp) = \tilde{\pi}^{\mathrm{ref}}(\wp)/\sum_{\wp' \in \mathcal{P}} \tilde{\pi}^{\mathrm{ref}}(\wp')$ which ensures that the reference probability is properly normalized. The path likelihoods $\tilde{\pi}^{\mathrm{ref}}(\wp)$ are already properly normalized in the case of hitting paths (see Section 4.2): $\sum_{\wp \in \mathcal{P}} \tilde{\pi}^{\mathrm{ref}}(\wp) = 1$. On the contrary, for regular paths as considered in this section, $\tilde{\pi}^{\mathrm{ref}}(\wp)$ is not properly normalized and it can be shown that $\sum_{\wp \in \mathcal{P}} \tilde{\pi}^{\mathrm{ref}}(\wp) = (t+1)n$, where t is the maximum length of considered paths in $\mathcal{P}$ [94]. Interested readers are advised to read [94] for more details.

Here, $J_0 > 0$ (the relative entropy) is provided a priori by the user, according to the desired degree of randomness, or exploration, he is willing to concede. Note also that, normally, a non-negativity constraint on the paths probability should be added, but this is not necessary since the resulting probabilities are automatically non-negative.

As well-known (see [130, 139] and [94, 167, 208] for maximum entropy distributions over paths), this problem is similar to the standard maximum entropy problem and can be solved by introducing the following Lagrange function integrating equality constraints

$$\mathscr{L} = \sum_{\wp \in \mathcal{P}} \mathrm{P}(\wp)\tilde{c}(\wp) + \lambda \left[\sum_{\wp \in \mathcal{P}} \mathrm{P}(\wp) \ln\left(\frac{\mathrm{P}(\wp)}{\tilde{\mathrm{P}}^{\mathrm{ref}}(\wp)}\right) - J_0 \right] + \mu \left[\sum_{\wp \in \mathcal{P}} \mathrm{P}(\wp) - 1 \right]$$

and optimizing over the set of path probabilities $\{\mathrm{P}(\wp)\}_{\wp \in \mathcal{P}}$ (the partial derivatives are set to zero). The Lagrange parameters are then deduced after imposing the constraints.

The result of this minimization (see [94] for details) is a **Boltzmann probability distribution**:

$$\mathrm{P}(\wp) = \frac{\tilde{\pi}^{\mathrm{ref}}(\wp) \exp\left[-\theta \tilde{c}(\wp)\right]}{\sum\limits_{\wp' \in \mathcal{P}} \tilde{\pi}^{\mathrm{ref}}(\wp') \exp[-\theta \tilde{c}(\wp')]} \tag{4.2}$$

where $\theta = 1/T$ plays the role of an inverse temperature and replaces J_0 as the desired degree of randomness/exploration parameter. $\exp$ is the elementwise exponential and $\tilde{\pi}(\wp)$ is the likelihood of the path $\wp$, according to the natural random walk on G (the reference random walk) defined earlier in this section.

As expected, short paths $\wp$ (having a low $\tilde{c}(\wp)$) are favored in that they have a larger probability of being chosen. Moreover, from Equation (4.2), we clearly observe that when $\theta \to 0^+$, the paths probabilities reduce to the probabilities generated by the natural random walk on the graph. In this case, paths are chosen according to their likelihood in a natural random walk. On the other hand, when θ is large, the probability distribution defined by Equation (4.2) is biased towards short paths (shortest ones are more likely to be chosen). Notice that, in the sequel, it will be assumed that the user provides the value of the parameter θ instead of J_0, with $\theta > 0$.

The **bag-of-paths probability** [94] is now defined as the quantity $\text{P}(s = i, e = j)$ on the set of (starting, ending) nodes of the paths. It corresponds to the probability of drawing a path starting in node i and ending in node j from the virtual bag-of-paths:

$$\text{P}(s = i, e = j) = \frac{\sum\limits_{\wp \in \mathcal{P}_{ij}} \tilde{\pi}^{\text{ref}}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\sum\limits_{\wp' \in \mathcal{P}} \tilde{\pi}^{\text{ref}}(\wp') \exp[-\theta \tilde{c}(\wp')]} \tag{4.3}$$

where $\mathcal{P}_{ij}$ is the set of paths connecting the starting node i to the ending node j.

Let us derive the analytical closed form of this expression. To this end, we start from the cost matrix, $\mathbf{C}$ (see Section 2.4), from which we build a new matrix, $\mathbf{W}$, as

$$\mathbf{W} = \mathbf{P}^{\text{ref}} \circ \exp[-\theta \mathbf{C}] \tag{4.4}$$

where $\mathbf{P}^{\text{ref}}$ is the reference transition matrix containing the p_{ij}^{ref} (provided by Equation (2.8)), the exponential function is taken elementwise and $\circ$ is the elementwise multiplication (Hadamard product). The entries of $\mathbf{W}$ are therefore $w_{ij} = p_{ij}^{\text{ref}} \exp[-\theta c_{ij}]$.

It is shown in [94] that the numerator of Equation (4.3) is

$$\sum_{\wp \in \mathcal{P}_{ij}} \tilde{\pi}^{\text{ref}}(\wp) \exp\left[-\theta \tilde{c}(\wp)\right] = \sum_{t=0}^{\infty} \left[\mathbf{W}^t\right]_{ij} = \left[\sum_{t=0}^{\infty} \mathbf{W}^t\right]_{ij} \tag{4.5}$$

where, by convention, zero-length paths are taken into account and are associated to a zero cost. Computing the series of powers of $\mathbf{W}$ provides

$$\sum_{t=0}^{\infty} \mathbf{W}^t = (\mathbf{I} - \mathbf{W})^{-1} = \mathbf{Z} \tag{4.6}$$

which converges if the spectral radius of $\mathbf{W}$ is less than 1, $\rho(\mathbf{W}) < 1$. Since the matrix $\mathbf{W}$ only contains non-negative elements, a sufficient condition for $\rho(\mathbf{W}) < 1$ is that the matrix is sub-stochastic and G is strongly connected, which is always achieved for $\theta > 0$ and at least one $c_{ij} > 0$ when $a_{ij} > 0$ (see Equation (4.4)), which is assumed for now. The matrix $\mathbf{Z}$ is called the **fundamental matrix** and z_{ij} is the element i, j of $\mathbf{Z}$.

Hence, following Equations (4.5-4.6), we finally obtain, for the numerator of Equation (4.3),

$$\sum_{\wp \in \mathcal{P}_{ij}} \tilde{\pi}^{\text{ref}}(\wp) \exp\left[-\theta \tilde{c}(\wp)\right] = z_{ij} \tag{4.7}$$

On the other hand, for the denominator of Equation (4.3), we have

$$\sum_{i,j=1}^{n} \sum_{\wp \in \mathcal{P}_{ij}} \tilde{\pi}^{\text{ref}}(\wp) \exp\left[-\theta \tilde{c}(\wp)\right] = \sum_{i,j=1}^{n} z_{ij} \triangleq \mathcal{Z}, \tag{4.8}$$

where $\mathcal{Z}$ is called the **partition function**.

Therefore, from Equation (4.3), the probability of picking a path starting in i and ending in j in our bag-of-paths model is simply

$$\mathrm{P}(s = i, e = j) = \frac{z_{ij}}{\mathcal{Z}}, \text{ with } \mathbf{Z} = (\mathbf{I} - \mathbf{W})^{-1} \tag{4.9}$$

Notice that $\mathrm{P}(s = i, e = j)$ is not symmetric[1]. These probabilities quantify the **relative accessibility** between the nodes and it was shown that minus their logarithm, $-\log \mathrm{P}(s = i, e = j)$, defines a useful distance measure between nodes [94]. By construction, this probability is high when the two nodes i and j are highly connected (there are many terms in the numerator of Equation (4.7)) by low-cost paths (each term of the numerator is large). In other words, it accurately captures the intuitive notion of relative accessibility. These BoP probabilities are used to define the BoP criticality in Chapter 7.

Note that the BoP probabilities can also be used to define some betweenness measures [145] which are related to well-known centrality/betweenness measures in some sense: if $\theta \to \infty$ the betweenness tends to be highly correlated with Freeman's betweenness [96] (only shortest paths are considered),

[1]For a symmetric variant in the case of undirected graphs [94], we can consider the probability of picking either $i \rightsquigarrow j$ or $j \rightsquigarrow i$, which is $\mathrm{P}(s = i, e = j) + \mathrm{P}(s = j, e = i)$.

while if $\theta \to 0^+$, the betweenness tends to be highly correlated with Newman's random walk betweenness [181]. More details can be found in Chapter 5.

We now turn to a variant of the bag-of-paths, the **bag-of-hitting-paths**.

4.2 The bag-of-hitting-paths framework

The idea behind the bag-of-hitting-paths model is the same as the bag-of-paths model but the set of paths is now restricted to paths in which the ending node does not appear more than once. In other words, no intermediate node on the path is allowed to be the ending node j (node j is made **absorbing**) and the motifs are now the hitting paths. Hitting paths plays an important role in the derivation of the BoP betweenness in Chapter 5.
In fact, each non-hitting path $\wp_{ij} \in \mathcal{P}_{ij}$ can be split uniquely into two sub-paths, before hitting node j for the first time, $\wp^{\mathrm{h}}_{ij} \in \mathcal{P}^{\mathrm{h}}_{ij}$ (the set of all hitting paths), and after hitting node j, $\wp_{jj} \in \mathcal{P}_{jj}$ (see [94] for details). Notice the usage of the superscript h to refer to hitting paths. The composition of the two sub-paths is a valid path, where $\wp^{\mathrm{h}}_{ij} \circ \wp_{jj} \in \mathcal{P}_{ij}$ is the concatenation of the two paths.

In the case of a bag containing hitting paths, the probability of picking a path $i \rightsquigarrow j$ is defined in a similar way as for non-hitting paths (Equation (4.3)),

$$\mathrm{P}_{\mathrm{h}}(s=i, e=j) = \frac{\displaystyle\sum_{\wp \in \mathcal{P}^{\mathrm{h}}_{ij}} \tilde{\pi}^{\mathrm{ref}}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\displaystyle\sum_{\wp' \in \mathcal{P}^{\mathrm{h}}} \tilde{\pi}^{\mathrm{ref}}(\wp') \exp[-\theta \tilde{c}(\wp')]} \tag{4.10}$$

and is called the **bag-of-hitting-paths probability** distribution.

Now, since $\tilde{c}(\wp_{ij}) = \tilde{c}(\wp^{\mathrm{h}}_{ij}) + \tilde{c}(\wp_{jj})$ and $\tilde{\pi}^{\mathrm{ref}}(\wp_{ij}) = \tilde{\pi}^{\mathrm{ref}}(\wp^{\mathrm{h}}_{ij})\tilde{\pi}^{\mathrm{ref}}(\wp_{jj})$ for any $\wp_{ij} = \wp^{\mathrm{h}}_{ij} \circ \wp_{jj}$, we easily obtain

$$\begin{aligned} z_{ij} &= \sum_{\wp_{ij} \in \mathcal{P}_{ij}} \tilde{\pi}^{\mathrm{ref}}(\wp_{ij}) \exp[-\theta \tilde{c}(\wp_{ij})] \\ &= \sum_{\wp^{\mathrm{h}}_{ij} \in \mathcal{P}^{\mathrm{h}}_{ij}} \sum_{\wp_{jj} \in \mathcal{P}_{jj}} \tilde{\pi}^{\mathrm{ref}}(\wp^{\mathrm{h}}_{ij}) \exp[-\theta \tilde{c}(\wp^{\mathrm{h}}_{ij})] \\ &\qquad \times \tilde{\pi}^{\mathrm{ref}}(\wp_{jj}) \exp[-\theta \tilde{c}(\wp_{jj})] \\ &= z^{\mathrm{h}}_{ij} z_{jj}. \end{aligned} \tag{4.11}$$

From which we deduce $z_{ij}^{\mathrm{h}} = \sum_{\wp \in \mathcal{P}_{ij}^{\mathrm{h}}} \tilde{\pi}^{\mathrm{ref}}(\wp) \exp[-\theta \tilde{c}(\wp)] = z_{ij}/z_{jj}$. Now, by analogy with Equation (4.9), but for hitting paths (Equation (4.10)),

$$\mathrm{P}_{\mathrm{h}}(s = i, e = j) = \frac{z_{ij}^{\mathrm{h}}}{\sum\limits_{i'=1}^{n} \sum\limits_{j'=1}^{n} z_{i'j'}^{\mathrm{h}}} = \frac{z_{ij}^{\mathrm{h}}}{\mathcal{Z}_{\mathrm{h}}} \tag{4.12}$$

and the partition function for the bag-of-hitting-paths is therefore

$$\mathcal{Z}_{\mathrm{h}} = \sum_{i,j=1}^{n} z_{ij}^{\mathrm{h}} = \sum_{i,j=1}^{n} \frac{z_{ij}}{z_{jj}} \tag{4.13}$$

Let us finally mention that another derivation is available in [94], where it is further shown that z_{ij}^{h} can be interpreted as either

- The expected reward endorsed by an agent (the reward along a path $\wp$ being defined as $\exp[-\theta \tilde{c}(\wp)]$) when traveling from i to j along all possible hitting paths $\wp \in \mathcal{P}_{ij}^{h}$ with probability $\tilde{\pi}^{\mathrm{ref}}(\wp)$.
- The probability of surviving during the killed random walk from i to j with transition probabilities w_{ij} and node j made killing and absorbing. In other words, it corresponds to the probability of reaching absorbing node j without being killed during the walk (i.e. the probability of surviving).

Chapter 5

Semi-supervised classification through the bag-of-paths group betweenness

This chapter tackles the **graph-based semi-supervised classification** problem (see Section 3.4) within the **bag-of-paths** (BoP) **framework** introduced in Chapter 4. More precisely, we assume a weighted directed graph or network G where a transition cost is associated to each arc, as described in Chapter 2. We further consider, as detailed in Chapter 4, a bag containing all the possible paths (or walks) between pairs of nodes in G. Then, a Boltzmann distribution, depending on a temperature parameter T, is defined on the set of paths such that long (high-cost) paths have a low probability of being picked from the bag, while short (low-cost) paths have a high probability of being picked.

In this framework, the **BoP probabilities**, $\mathrm{P}(s = i, e = j)$, of sampling a path starting in node i and ending in node j can easily be computed in closed form by a simple $n \times n$ matrix inversion, where n is the number of nodes (for more details, see Chapter 4 and in particular Equation 4.3).

Within this context, a betweenness measure quantifying to what extent a node j is in-between two nodes i and k is defined. More precisely, the **BoP betweenness**, $\mathrm{bet}_j = \sum_{i=1}^{n} \sum_{k=1}^{n} \mathrm{P}(int = j | s = i, e = k)$, of a node j of interest is computed quite naturally as the sum of the a posteriori probabilities that node j (the intermediate node) lies on a path between the two nodes i and k sampled from the graph bag-of-paths according to a Boltzmann distribution. Intuitively, a node receives a high betweenness if it has a large probability of appearing on paths connecting two arbitrary nodes of the network.

For the **group betweenness**, the paths are constrained to start and end in classes, therefore defining a group betweenness between classes,

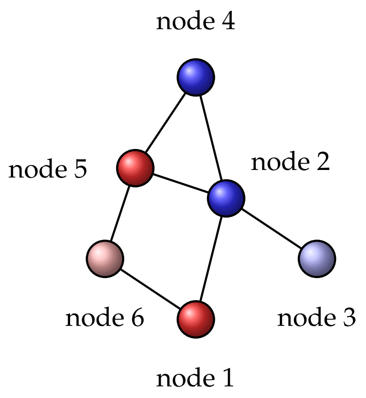


FIGURE 5.1: A small toy graph. According to the bag-of-path classifier, node 3 belongs to the blue class and node 6 belongs to the red class. Dark red and blue nodes are training samples. For this example θ was set to 1.

$\text{gbet}_j(\mathcal{C}_i, \mathcal{C}_k) = \text{P}(int = j | s \in \mathcal{C}_i, e \in \mathcal{C}_k)$. Unlabeled nodes are then classified according to the class showing the highest group betweenness when starting and ending within the same class. More details are given in Section 5.2.

The **bag-of-paths classifier** (just called BoP) will refer to the semi-supervised classification algorithm based on the bag-of-paths group betweenness and is further developed in Section 5.3. A toy example using the bag-of-paths classifier can be found on Figure 5.1

This chapter (and the related paper, see Chapter 1) has three main contributions:

- It develops both a betweenness measure and a group betweenness measure from a well-founded theoretical framework, the bag-of-paths framework. These two measures can be easily computed in closed form.
- This group betweenness measure provides a new algorithm for graph-based semi-supervised classification.
- It assesses the accuracy of the proposed algorithm on thirteen standard data sets and compares it to state-of-the-art techniques. The obtained performances are competitive with the other graph-based semi-supervised techniques.

The main drawback of the proposed method is that it requires a matrix inversion and therefore does not scale to large graphs.

This chapter is organized as follows: related work in semi-supervised classification is discussed in Section 5.1. The bag-of-paths betweenness and group betweenness centralities are introduced in Section 5.2.2. This enables us to derive the BoP classifier in Section 5.3. Then experiments involving the BoP classifier and baseline classifiers discussed in the related work section are described in Section 5.4. Results and discussions of those experiments can be found in Section 5.4.3. Finally, Section 5.5 concludes this chapter and opens a reflection for further work.

5.1 Related work

Graph-based semi-supervised classification has been the subject of intensive research in recent years and a wide range of approaches has been developed in order to tackle the problem [2, 268, 58]: Random-walk-based methods [263, 226], spectral methods [59, 136], regularization frameworks [262, 27, 248, 265], transductive and spectral SVM [132]. Many other approaches have been proposed in different fields for classification in the presence of structural correlations, such as kriging in spatial statistics [40, 68, 113, 192], spatial autoregressive models in spatial econometrics [10, 157], various Laplacian-based regularization frameworks for semi-supervised classification [219, 264], random-walk based or label propagation models in machine learning [30].

We compared the BoP classifier to some of those techniques, namely,

1. A simple alignment with the regularized Laplacian kernel (RL) based on a sum of similarities, $\mathbf{K}_{\mathrm{RL}}\mathbf{y}^c$, where $\mathbf{K}_{\mathrm{RL}} = (\mathbf{I} + \lambda\mathbf{L})^{-1}$, $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the Laplacian matrix, $\mathbf{I}$ is the identity matrix, $\mathbf{D}$ is the generalized outdegree matrix, and $\mathbf{A}$ is the adjacency matrix of G [27, 140]. The similarity is computed for each class c in turn. Then, each node is assigned to the class showing the largest sum of similarities. The (scalar) parameter $\lambda > 0$ is the regularization parameter [91, 168].

2. A simple alignment with the regularized normalized Laplacian kernel (RNL) based on a sum of similarities, $\mathbf{K}_{\mathrm{RNL}}\mathbf{y}^c$, where $\mathbf{K}_{\mathrm{RNL}} = (\mathbf{I}+\lambda\widetilde{\mathbf{L}})^{-1}$, and $\widetilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ is the normalized Laplacian matrix [265, 264]. The assignment to the classes is the same as for the previous method. The regularized normalized Laplacian approach seems less sensitive to the priors of the different classes than the un-normalized regularized Laplacian approach (RL) [264].

3. A simple alignment with the regularized commute time kernel (RCT) based on a sum of similarities, $\mathbf{K}_{\mathrm{RCT}}\mathbf{y}^c$, with $\mathbf{K}_{\mathrm{RCT}} = (\mathbf{D} - \alpha\mathbf{A})^{-1}$ [265, 91]. The assignment to the classes is the same as for previous methods.

Element i, j of this kernel can be interpreted as the discounted cumulated probability of visiting node j when starting from node i. The (scalar) parameter $\alpha \in]0, 1]$ corresponds to an evaporating or killed random walk where the random walker has a $(1-\alpha)$ probability of disappearing at each step. This method provided the best results in a previous comparative study on semi-supervised classification [91].

4. The harmonic function (HF) approach [267, 2], is closely related to the regularization framework of RL and RNL. Furthermore, it is equivalent and provides the same results as the label propagation algorithm [58] and the wvRN (or pRN) classifier used by the Netkit software as a baseline [165], but Netkit is a more general-purpose toolbox able to tackle more complex situations [104]. As those three algorithms give the same results, we only report HF which appeared first in the litterature and is fastest. It is based on a structural contiguity measure that smoothes the predicted values and leads to a model having interesting interpretations in terms of electrical potential and absorbing probabilities in a Markov chain.

5. The random walk with restart (RWWR) classifier [188, 238] relies on random walks performed on the weighted graph seen as a Markov chain. More precisely, a group betweenness measure is derived for each class, based on the stationary distribution of a random walk restarting from the labeled nodes belonging to a class of interest. Each unlabeled node is then assigned to the class showing maximal betweenness. In this version [91], the random walker has a probability $(1 - \alpha)$ to be teleported – with a uniform probability – to a node belonging to the class of interest c.

6. The discriminative random walks approach ($\mathcal{D}$-walk or DW1; see [54]) also relies on random walks performed on the weighted graph. As for the RWWR, a group betweenness measure, based on passage times during random walks, is derived for each class. More precisely, a $\mathcal{D}$-walk is a random walk starting in a labeled node and ending when any node having the same label (possibly the starting node itself) is reached for the first time. During this random walk, the number of visits to any unlabeled node is recorded and corresponds to a group betweenness measure. As for the previous method, each unlabeled node is then assigned to the class showing maximal betweenness.

7. A modified version of the $\mathcal{D}$-walk (or DW2). The only difference is that all elements of the transition matrix $\mathbf{P}^{\text{ref}}$ (since the random walks is seen as a Markov chain) are multiplied by $\alpha \in]0, 1]$ so that α can be seen as a probability of continuing the random walk at each time step (and so

$(1-\alpha) \in [0, 1[$ is the probability of stopping the random walk at each step. This defines a killed random walk since $\alpha\mathbf{P}^{\text{ref}}$ is now sub-stochastic.

Notice that the random walker of the random-walk-based methods usually takes too long – and thus irrelevant – paths into account: popular entries are therefore intrinsically favored [159, 48]. The bag-of-paths approach tackles this issue by putting a negative exponential term in Equation (4.4) and part of its success can be imputed to this fact.

Some authors also considered bounded (or truncated) walks [168, 210, 55] and obtained promising results on large graphs.

Tong et al. suggested a method avoiding to take the inverse of an $n \times n$ matrix for computing the random walk with restart measure [238]. They reduce the computing time by partitioning the input graph into smaller communities. Then, a sparse approximate of the random walk with restart is obtained by applying a low rank approximation. This approach suffers from the fact that it adds a hyperparameter k (the number of communities) that depends on the network and is still untractable for large graphs with millions of nodes. On the other hand, the computing time is reduced by this same factor k. This is another track to investigate in further work.

Herbster et al. [120] proposed a technique for fast label prediction on graphs through the approximation of the graph with either a minimum spanning tree or a shortest path tree. Once the tree has been extracted, the pseudo inverse of the Laplacian matrix has to be computed. The fast computation of the pseudo inverse enables to address prediction problems on large graphs.

Finally, Tang and Liu have investigated relational learning via latent social dimensions [228, 229, 230]. They proposed to extract latent social dimensions based on network information (such as Facebook, Twitter,...) first, then they used these as features for discriminative learning (via a SVM, for example [228]). Their approach tackles very large networks and provides promising results, especially when only a few labeled data are available.

A lot of research has also been done on collective classification of nodes in networks (see [105] for an introduction). Collective classification also uses the graph topology and a proportion of labeled nodes to classify unlabeled nodes using the same assumption as our proposed technique (i.e. local consistency or homophily).

5.2 The bag-of-paths betweennesses

In order to define the BoP classifier, we need to introduce the BoP group betweenness centrality. This concept is itself an extension of the BoP betweenness centrality, which is developed first. The BoP betweenness is related to well-known betweenness measures in some sense: if $\theta \to \infty$ the BoP betweenness tends to be correlated with Freeman's betweenness [96] (only shortest paths are considered), while if $\theta \to 0^+$, the BoP betweenness tends to be correlated with Newman's betweenness [181] (based on a natural random walk).

This section starts with the presentation of the BoP betweenness centrality measure in Section 5.2.1. Then, its extension, the BoP group betweenness centrality, is described in Section 5.2.2.

5.2.1 The bag-of-paths betweenness centrality

The BoP betweenness measure measures to what extent a node j is likely to lie in-between other pairs of nodes i, k, and therefore is an important intermediary between nodes. In short, the **bag-of-paths betweenness measure** is defined as

$$\text{bet}_j = \sum_{i=1}^{n} \sum_{k=1}^{n} \text{P}(int = j | s = i, e = k; i \neq j \neq k \neq i) \tag{5.1}$$

which corresponds to the a posteriori probability of finding intermediate node $int = j$ on a path $i \rightsquigarrow k$ drawn from the bag of paths, cumulated over all source-destination pairs i, k (with $i \neq k$).

For computing this quantity from the bag-of-paths framework, we first have to calculate the probability $\text{P}(s = i, int = j, e = k; i \neq j \neq k \neq i)$ – or P_{ijk} in short – that such paths visit an intermediate node $int = j$ with $i \neq j \neq k \neq i$. Indeed, by using Equations (4.2) and (4.3),

$$\begin{aligned}
\text{P}_{ijk} &= \sum_{\wp \in \mathcal{P}_{ik}} \delta(j \in \wp)\, \text{P}(\wp) \\
&= \frac{\displaystyle\sum_{\wp \in \mathcal{P}_{ik}} \delta(j \in \wp)\, \tilde{\pi}^{\text{ref}}(\wp) \exp\left[-\theta \tilde{c}(\wp)\right]}{\displaystyle\sum_{\wp' \in \mathcal{P}} \tilde{\pi}^{\text{ref}}(\wp') \exp\left[-\theta \tilde{c}(\wp')\right]} \\
&= \frac{1}{\mathcal{Z}} \sum_{\wp \in \mathcal{P}_{ik}} \delta(j \in \wp)\, \tilde{\pi}^{\text{ref}}(\wp) \exp\left[-\theta \tilde{c}(\wp)\right]
\end{aligned} \tag{5.2}$$

where $\delta(j \in \wp) = 1$ when node j is visited on path $\wp$, and 0 otherwise.

Each path $\wp_{ik}$ between i and k passing through j can be decomposed uniquely into a hitting sub-path $\wp_{ij}$ from i to j and a regular sub-path $\wp^{\text{h}}_{jk}$ from j to k (see Section 4.2). The sub-path $\wp^{\text{h}}_{ij}$ is found by following path $\wp_{ik}$

until reaching j for the first time. Therefore, for $i \neq j \neq k \neq i$,

$$\begin{aligned} \mathrm{P}_{ijk} = \frac{1}{\mathcal{Z}} \sum_{\wp^{\mathrm{h}}_{ij} \in \mathcal{P}^{\mathrm{h}}_{ij}} \sum_{\wp_{jk} \in \mathcal{P}_{jk}} & \tilde{\pi}^{\mathrm{ref}}(\wp^{\mathrm{h}}_{ij}) \tilde{\pi}^{\mathrm{ref}}(\wp_{jk}) \\ & \times \exp\left[-\theta \tilde{c}(\wp^{\mathrm{h}}_{ij})\right] \exp\left[-\theta \tilde{c}(\wp_{jk})\right] \end{aligned} \tag{5.3}$$

This equation can be reordered to get, for $i \neq j \neq k \neq i$:

$$\begin{aligned} \mathrm{P}_{ijk} = \frac{1}{\mathcal{Z}} & \left[\sum_{\wp^{\mathrm{h}}_{ij} \in \mathcal{P}^{\mathrm{h}}_{ij}} \tilde{\pi}^{\mathrm{ref}}(\wp^{\mathrm{h}}_{ij}) \exp\left[-\theta \tilde{c}(\wp^{\mathrm{h}}_{ij})\right] \right] \\ \times & \left[\sum_{\wp_{jk} \in \mathcal{P}_{jk}} \tilde{\pi}^{\mathrm{ref}}(\wp_{jk}) \exp\left[-\theta \tilde{c}(\wp_{jk})\right] \right] \end{aligned} \tag{5.4}$$

Then, after multiplying by $\mathcal{Z}_{\mathrm{h}}/\mathcal{Z}_{\mathrm{h}}$, we obtain

$$\begin{aligned} \mathrm{P}_{ijk} &= \frac{1}{\mathcal{Z}} z^{\mathrm{h}}_{ij} z_{jk} = \mathcal{Z}_{\mathrm{h}} \frac{z^{\mathrm{h}}_{ij}}{\mathcal{Z}_{\mathrm{h}}} \frac{z_{jk}}{\mathcal{Z}} \\ &= \mathcal{Z}_{\mathrm{h}}\, \mathrm{P}_{\mathrm{h}}(s=i, e=j)\, \mathrm{P}(s=j, e=k) \end{aligned} \tag{5.5}$$

with $i \neq j \neq k \neq i$ and where we used Equations (4.11) and (4.12).
Finally, recalling Equations (4.9), (4.12),

$$\begin{aligned} \mathrm{P}_{ijk} &= \frac{\left(\dfrac{z_{ij}}{z_{jj}}\right)(z_{jk})}{\mathcal{Z}} \\ &= \frac{1}{\mathcal{Z}} \frac{z_{ij} z_{jk}}{z_{jj}}, \text{ with } i \neq j \neq k \neq i \\ &= \frac{1}{\mathcal{Z}} \frac{z_{ij} z_{jk}}{z_{jj}}\, \delta(i \neq j \neq k \neq i) \end{aligned} \tag{5.6}$$

Now, using the Bayes' rule, the a posteriori probabilitiy of finding intermediate node j on a path starting in i and ending in k is

$$\begin{aligned} & \mathrm{P}(int = j | s = i, e = k; i \neq j \neq k \neq i) \\ & \quad = \frac{\mathrm{P}(s = i, int = j, e = k; i \neq j \neq k \neq i)}{\sum_{j'=1}^{n} \mathrm{P}(s = i, int = j', e = k; i \neq j' \neq k \neq i)} \end{aligned} \tag{5.7}$$

Using Equation (5.6), if we assume that node k can be reached from node i, this leads to

$$\begin{aligned}&\mathrm{P}(int = j|s = i, e = k; i \neq j \neq k \neq i) \\ &= \frac{\left(\dfrac{z_{ij}z_{jk}}{z_{jj}}\right)}{\displaystyle\sum_{\substack{j'=1 \\ j'\neq\{i,k\}}}^{n} \left(\dfrac{z_{ij'}z_{j'k}}{z_{j'j'}}\right)} \delta(i \neq j \neq k \neq i) \end{aligned} \tag{5.8}$$

Based on this a posteriori probability distribution, the **bag-of-paths betweenness** of node j is defined as the sum of the a posteriori probabilities of visiting j for all possible starting-ending pairs i, k:

$$\mathrm{bet}_j = \sum_{i=1}^{n}\sum_{k=1}^{n} \mathrm{P}(int = j|s = i, e = k; i \neq j \neq k \neq i) \tag{5.9}$$

$$= \frac{1}{z_{jj}} \sum_{\substack{i=1 \\ i\neq j}}^{n} \sum_{\substack{k=1 \\ k\neq\{i,j\}}}^{n} \frac{z_{ij}z_{jk}}{\displaystyle\sum_{\substack{j'=1 \\ j'\neq\{i,k\}}}^{n} \left(\dfrac{z_{ij'}z_{j'k}}{z_{j'j'}}\right)} \tag{5.10}$$

which allows to compute the betweenness from the fundamental matrix $\mathbf{Z}$ (Equation (4.6)).

Let us now derive the matrix formula providing the betweenness vector $\mathbf{bet}$, containing the betweennesses for each node. First of all, the normalization factor appearing in the denominator of Equation (5.10), denoted here by n_{ik}, is computed,

$$n_{ik} = \sum_{j'=1}^{n} (1 - \delta_{ij'})(1 - \delta_{j'k})\ (z_{ij'}z_{j'k})/z_{j'j'} \tag{5.11}$$

which can be re-written as

$$n_{ik} = \sum_{j'=1}^{n} \{(1 - \delta_{ij'})z_{ij'}\}\{1/z_{j'j'}\}\{(1 - \delta_{j'k})z_{j'k}\} \tag{5.12}$$

Therefore, the matrix containing the normalization factors n_{ik} is

$$\mathbf{N} = (\mathbf{Z} - \mathbf{Diag}(\mathbf{Z}))\,(\mathbf{Diag}(\mathbf{Z}))^{-1}(\mathbf{Z} - \mathbf{Diag}(\mathbf{Z})) \tag{5.13}$$

where $\mathbf{Diag}(\mathbf{M})$ is a diagonal matrix containing the diagonal of matrix $\mathbf{M}$.

Moreover, the inner term appearing in Equation (5.10) can be rewritten as

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{k=1}^{n}\delta(i \neq j \neq k \neq i) z_{ij}(1/n_{ik}) z_{jk} \\
&= \sum_{i=1}^{n}\sum_{k=1}^{n}\{(1-\delta_{ji}) z^{\mathrm{t}}_{ji}\}\{(1-\delta_{ik})(1/n_{ik})\}\{(1-\delta_{kj}) z^{\mathrm{t}}_{kj}\}
\end{aligned} \tag{5.14}$$

where z^{t}_{ij} is the element i, j of matrix $\mathbf{Z}^{\mathrm{T}}$ (transpose of $\mathbf{Z}$). In matrix form, $\mathbf{bet}$ (see Equation (5.8)) is therefore equal to

$$\begin{aligned}
\mathbf{bet} =& (\mathbf{Diag}(\mathbf{Z}))^{-1}\mathbf{diag}\big[(\mathbf{Z}^{\mathrm{T}} - \mathbf{Diag}(\mathbf{Z})) \\
&\times (\mathbf{N}^{\div} - \mathbf{Diag}(\mathbf{N}^{\div}))(\mathbf{Z}^{\mathrm{T}} - \mathbf{Diag}(\mathbf{Z}))\big]
\end{aligned} \tag{5.15}$$

with matrix $\mathbf{N}^{\div}$ containing elements $n^{\div}_{ik} = 1/n_{ik}$ (elementwise reciprocal).

5.2.2 The bag-of-paths group betweenness centrality

Let us now generalize the bag-of-paths betweenness to a group betweenness measure. The **bag-of-paths group betweenness** of node j will be defined as

$$\mathrm{gbet}_j(\mathcal{C}_i, \mathcal{C}_k) = \mathrm{P}(int = j | s \in \mathcal{C}_i, e \in \mathcal{C}_k; s \neq int \neq e \neq s) \tag{5.16}$$

and can be interpreted as the extent to which the node j lies in-between the two subsets of nodes $\mathcal{C}_i$ and $\mathcal{C}_k$. It is assumed that the sets $\{\mathcal{C}_i\}_{i=1}^{m}$ are disjoint. Using Bayes' law provides

$$\begin{aligned}
&\mathrm{P}(int = j | s \in \mathcal{C}_i, e \in \mathcal{C}_k; s \neq int \neq e \neq s) \\
&= \frac{\mathrm{P}(s \in \mathcal{C}_i, int = j, e \in \mathcal{C}_k; s \neq int \neq e \neq s)}{\mathrm{P}(s \in \mathcal{C}_i, e \in \mathcal{C}_k; s \neq int \neq e \neq s)} \\
&= \frac{\displaystyle\sum_{i' \in \mathcal{C}_i}\sum_{k' \in \mathcal{C}_k} \mathrm{P}(s = i', int = j, e = k'; s \neq int \neq e \neq s)}{\displaystyle\sum_{j'=1}^{n}\sum_{i' \in \mathcal{C}_i}\sum_{k' \in \mathcal{C}_k} \mathrm{P}(s = i', int = j', e = k'; s \neq int \neq e \neq s)}
\end{aligned} \tag{5.17}$$

Substituting (5.6) for the joint probabilities in Equation (5.17) allows to compute the group betweenness measure in terms of the elements of the fundamental matrix $\mathbf{Z}$:

$$\mathrm{gbet}_j(\mathcal{C}_i, \mathcal{C}_k) = \frac{\displaystyle\sum_{i' \in \mathcal{C}_i} \sum_{k' \in \mathcal{C}_k} \delta(i' \neq j \neq k' \neq i') \frac{z_{i'j} z_{jk'}}{z_{jj}}}{\displaystyle\sum_{j'=1}^{n} \sum_{i' \in \mathcal{C}_i} \sum_{k' \in \mathcal{C}_k} \delta(i' \neq j' \neq k' \neq i') \frac{z_{i'j'} z_{j'k'}}{z_{j'j'}}} \tag{5.18}$$

where the denominator is simply a normalization factor ensuring that the probability distribution sums to one. It is therefore sufficient to compute the numerator only and then normalize the resulting quantity.

Let us put this expression in matrix form. As before, we denote element i, j of matrix $\mathbf{Z}^{\mathrm{T}}$ as z^{t}_{ij}. It is also assumed that nodes i' and k' belong to different groups, $\mathcal{C}_i \neq \mathcal{C}_k$, so that i and k are necessarily different (moreover, classes are disjoint). The numerator in Equation (5.18) is

$$\begin{aligned} &\mathrm{num}(\mathrm{gbet}_j(\mathcal{C}_i, \mathcal{C}_k)) \\ &\quad = \frac{1}{z_{jj}} \sum_{i' \in \mathcal{C}_i} \sum_{k' \in \mathcal{C}_k} (1 - \delta_{ji'})(1 - \delta_{jk'})\, z_{i'j} z_{jk'} \\ &\quad = \frac{1}{z_{jj}} \left(\sum_{i' \in \mathcal{C}_i} (1 - \delta_{ji'}) z^{\mathrm{t}}_{ji'} \right) \left(\sum_{k' \in \mathcal{C}_k} (1 - \delta_{jk'}) z_{jk'} \right) \end{aligned} \tag{5.19}$$

If y^c_i is a binary indicator indicating if node i belongs to the class c (as described in the introduction of Chapter 4), the numerator can be rewritten as

$$\begin{aligned} &\mathrm{num}(\mathrm{gbet}_j(\mathcal{C}_i, \mathcal{C}_k)) \\ &\quad = \frac{1}{z_{jj}} \left(\sum_{i'=1}^{n} (1 - \delta_{ji'}) z^{\mathrm{t}}_{ji'} y^i_{i'} \right) \left(\sum_{k'=1}^{n} (1 - \delta_{jk'}) z_{jk'} y^k_{k'} \right) \end{aligned} \tag{5.20}$$

Consequently, in matrix form, the group betweenness vector reads

$$\begin{cases} \mathbf{gbet}(\mathcal{C}_i, \mathcal{C}_k) \leftarrow (\mathbf{Diag}(\mathbf{Z}))^{-1} \left((\mathbf{Z}_0^{\mathrm{T}} \mathbf{y}^i) \circ (\mathbf{Z}_0 \mathbf{y}^k) \right) \\ \qquad\qquad\qquad\quad \text{with } \mathbf{Z}_0 = \mathbf{Z} - \mathbf{Diag}(\mathbf{Z}), \\ \mathbf{gbet}(\mathcal{C}_i, \mathcal{C}_k) \leftarrow \dfrac{\mathbf{gbet}(\mathcal{C}_i, \mathcal{C}_k)}{\|\mathbf{gbet}(\mathcal{C}_i, \mathcal{C}_k)\|_1} \text{ (normalization)} \end{cases} \tag{5.21}$$

where we assume $i \neq k$. In this equation, the vector $\mathbf{gbet}(\mathcal{C}_i, \mathcal{C}_k)$ must be normalized by dividing it by its L_1 norm. Notice that $\mathbf{Z}_0 = \mathbf{Z} - \mathbf{Diag}(\mathbf{Z})$ is simply the fundamental matrix whose diagonal is set to zero.

5.3 Semi-supervised classification through the bag-of-paths group betweenness

In this section, the bag-of-paths framework (more precisely the bag-of-paths group betweenness measure) is used for classification purposes. Notice, however, that in the derivation of the group betweenness measure (see Equation (5.21)), it was assumed that the starting and ending classes are different ($\mathcal{C}_i \neq \mathcal{C}_k$). We now recompute this quantity when starting and ending in the same class c, i.e. calculating $\text{gbet}_j(\mathcal{C}_c, \mathcal{C}_c)$, which provides a **within-class betweenness**. Indeed, this quantity measures to what extent the nodes of G are in-between – and therefore in the neighborhood of – the nodes of class c.

A within-class betweenness is thus computed for each class c and each node is assigned to the class showing the highest betweenness. This is our simple classification rule based on the within-class betweenness. The main hypothesis underlying this classification technique is that a node is likely to belong to the same class as its neighboring nodes. This is usually called the local consistency assumption (also called smoothness, homophily or cluster assumption [267, 58, 149], see Section 3.2).

The same reasoning as for deriving Equation (5.21) is applied in order to compute the numerator of (5.18) in this new case. We start with Equation (5.18), considering now the same starting and ending class c but multiplying the term inside the double sum by $(1 - \delta_{i'k'})$. This new term ensures that the starting node is different from the ending node (this was always the case with different starting and ending classes, but now this must be forced). From Equation (5.19), this can be rewritten as

$$\begin{aligned} \text{num}(\text{gbet}_j(\mathcal{C}_c, \mathcal{C}_c)) = \frac{1}{z_{jj}} \sum_{i',k' \in \mathcal{C}_c} & (1 - \delta_{ji'})(1 - \delta_{i'k'}) \\ & \times (1 - \delta_{jk'})\, z_{i'j} z_{jk'} \end{aligned} \tag{5.22}$$

$\text{num}(\text{gbet}_j(\mathcal{C}_c, \mathcal{C}_c))$ is thus the same as $\text{num}(\text{gbet}_j(\mathcal{C}_i, \mathcal{C}_k))$ with $\mathcal{C}_i = \mathcal{C}_c$ of Equation (5.19) and $\mathcal{C}_k = \mathcal{C}_c$, plus an extra term:

$$\begin{aligned} & \text{num}(\text{gbet}_j(\mathcal{C}_c, \mathcal{C}_c)) = \\ & \quad \frac{1}{z_{jj}} \sum_{i' \in \mathcal{C}_c} \sum_{k' \in \mathcal{C}_c} (1 - \delta_{ji'})(1 - \delta_{jk'})\, z_{i'j} z_{jk'} \\ & \quad - \frac{1}{z_{jj}} \sum_{i' \in \mathcal{C}_c} \sum_{k' \in \mathcal{C}_c} (1 - \delta_{ji'}) \delta_{i'k'} (1 - \delta_{jk'})\, z_{i'j} z_{jk'} \end{aligned} \tag{5.23}$$

TABLE 5.1: The eight classifiers, the value range tested for tuning their parameters and the most selected values. Mode1 is the most selected value, Mode2 is the second most selected value, and Mode3 is the third most selected value. Notice that DW2 with $\alpha = 1.0$ is the same as DW1.

Classifier name	**Acronym**	**Parameter**	**Tested values**	**Mode1**	**Mode2**	**Mode3**
Regularized Laplacian kernel	RL	$\lambda > 0$	$10^{-6}, 10^{-5}, ..., 10^{6}$	10^{-6} 12.3%	10^{-1} 11.7%	10^{-2} 11.5%
Regularized normalised Laplacian kernel	RNL	$\lambda > 0$	$10^{-6}, 10^{-5}, ..., 10^{6}$	10^{-1} 42.1%	10^{-2} 13.9%	10^{-3} 09.3%
Regularized commute-time kernel	RCT	$\alpha \in]0, 1]$	$0.1, 0.2, ..., 1$	0.9 28.0%	0.8 16.2%	0.7 12.2%
Harmonic function	HF	none	–	–	–	–
Random walk with restart	RWWR	$\alpha \in]0, 1]$	$0.1, 0.2, ..., 1$	0.9 45.8%	0.8 16.8%	0.7 10.1%
Discriminative random walks	DW1	none	–	–	–	–
Killed discriminative random walks	DW2	$\alpha \in]0, 1]$	$0.1, 0.2, ..., 1$	1.0 19.5%	0.1 11.8%	0.9 11.2%
BoP classifier	BoP	$\theta > 0$	$10^{-6}, 10^{-5}, ..., 10^{2}$	10^{-4} 28.3%	10^{-3} 25.9%	10^{-2} 12.3%

It is easy to show that with this additional term, the matrix equation for $\text{num}(\mathbf{gbet}(\mathcal{C}_c, \mathcal{C}_c))$ (Equation (5.21)) becomes [156]

$$\begin{aligned} &\text{num}(\mathbf{gbet}(\mathcal{C}_c, \mathcal{C}_c)) \\ &\quad = (\mathbf{Diag}(\mathbf{Z}))^{-1} \left[(\mathbf{Z}_0^{\text{T}} \mathbf{y}^c) \circ (\mathbf{Z}_0 \mathbf{y}^c) - (\mathbf{Z}_0^{\text{T}} \circ \mathbf{Z}_0) \mathbf{y}^c \right] \end{aligned} \tag{5.24}$$

Once again, this is the same result as in Equation (5.21) with one more term that ensures that the starting node is different from the ending node. After having computed this equation, the numerator must be normalized in order to obtain $\mathbf{gbet}(\mathcal{C}_c, \mathcal{C}_c)$ (as for Equation (5.21)).

Finally, if we want to classify a node, $\mathbf{gbet}(\mathcal{C}_c, \mathcal{C}_c)$ is computed for each class c in turn and then, for each node, the class label showing the maximal betweenness is chosen,

$$\begin{aligned} &\hat{\mathbf{y}} = \arg\max_{c \in \mathcal{L}} \{\mathbf{gbet}(\mathcal{C}_c, \mathcal{C}_c)\}, \text{ with} \\ &\begin{cases} \mathbf{D}_{\text{z}} = \mathbf{Diag}(\mathbf{Z}); \mathbf{Z}_0 = \mathbf{Z} - \mathbf{D}_{\text{z}} \text{ (set diagonal to 0)} \\ \mathbf{gbet}(\mathcal{C}_c, \mathcal{C}_c) \leftarrow \mathbf{D}_{\text{z}}^{-1} \left[(\mathbf{Z}_0^{\text{T}} \mathbf{y}^c) \circ (\mathbf{Z}_0 \mathbf{y}^c) - (\mathbf{Z}_0^{\text{T}} \circ \mathbf{Z}_0) \mathbf{y}^c \right] \\ \mathbf{gbet}(\mathcal{C}_c, \mathcal{C}_c) \leftarrow \dfrac{\mathbf{gbet}(\mathcal{C}_c, \mathcal{C}_c)}{\|\mathbf{gbet}(\mathcal{C}_c, \mathcal{C}_c)\|_1} \text{ (normalization)} \end{cases} \end{aligned} \tag{5.25}$$

where $\mathcal{L}$ is the set of class labels. The pseudo-code for the BoP classifier can be found in Algorithm 1. Of course, once computed, the group betweenness is only used to classify the unlabeled nodes.

Algorithm 1 Classification through the bag-of-paths group betweenness algorithm.

```
Input:
    – A weighted directed graph $G$ containing $n$ nodes, represented by its $n \times n$ adjacency matrix $\mathbf{A}$, containing affinities.
    – The $n \times n$ transition cost matrix $\mathbf{C}$ associated to $G$.
    – $m$ binary indicator vectors $\mathbf{y}^c$ containing as entries 1 for nodes belonging to the class $\mathcal{C}_c$, and 0 otherwise. Classes are mutually exclusive.
    – The inverse temperature parameter $\theta$.
Output:
    – The $n \times 1$ vector $\hat{\mathbf{y}}$ containing the predicted class labels of each node.
 1: $\mathbf{D} \leftarrow \mathbf{Diag}(\mathbf{Ae})$    ▷ the row-normalization matrix
 2: $\mathbf{P}^{\text{ref}} \leftarrow \mathbf{D}^{-1}\mathbf{A}$    ▷ the reference transition probabilities matrix
 3: $\mathbf{W} \leftarrow \mathbf{P}^{\text{ref}} \circ \exp[-\theta\mathbf{C}]$    ▷ elementwise exponential and multiplication $\circ$
 4: $\mathbf{Z} \leftarrow (\mathbf{I} - \mathbf{W})^{-1}$    ▷ the fundamental matrix
 5: $\mathbf{Z}_0 \leftarrow \mathbf{Z} - \mathbf{Diag}(\mathbf{Z})$    ▷ set diagonal to zero
 6: $\mathbf{D}_z \leftarrow \mathbf{Diag}(\mathbf{Z})$
 7: for $c = 1$ to $m$ do
 8:     $\mathbf{gbet}_c \leftarrow \mathbf{D}_z^{-1}\left[(\mathbf{Z}_0^{\text{T}}\mathbf{y}^c) \circ (\mathbf{Z}^0\mathbf{y}^c) - (\mathbf{Z}_0^{\text{T}} \circ \mathbf{Z}_0)\mathbf{y}^c\right]$    ▷ compute the group betweenness for class $c$; $\circ$ is the elementwise multiplication (Hadamard product)
 9:     $\mathbf{gbet}_c \leftarrow \dfrac{\mathbf{gbet}_c}{\|\mathbf{gbet}_c\|_1}$    ▷ normalize the betweenness scores
10: end for
11: $\hat{\mathbf{y}} \leftarrow \arg\max_{c \in \mathcal{L}}(\mathbf{gbet}_c)$    ▷ each node is assigned to the class showing the largest class betweenness
12: return $\hat{\mathbf{y}}$
```

5.4 Experimental comparisons

In this section, the bag-of-paths group betweenness approach for semi-supervised classification (referred to as the BoP classifier for simplicity) is compared to other semi-supervised classification techniques on multiple datasets. The different classifiers to which the BoP classifier is compared were already introduced in Section 5.1 and are recalled in Table 5.1.

The goal of the experiments of this section is to classify unlabeled nodes in partially labeled graphs and to compare the different methods in terms of classification accuracy. This comparison is performed on medium-size networks only since kernel approaches are difficult to compute on large networks. The computational tractability of the methods used in this experimental section is also analyzed.

This section is organized as follows: First, datasets are described in Section 5.4.1. Second, the experimental methodology is detailed in Section 5.4.2.

TABLE 5.2: Class distribution of the nine **Newsgroups** datasets. NG 1-3 contain two classes, NG 4-6 contain three classes, and NG 7-9 contain five classes.

Class	**NG1**	**NG2**	**NG3**	**NG4**	**NG5**	**NG6**	**NG7**	**NG8**	**NG9**
1	200	198	200	200	200	197	200	200	200
2	200	200	199	200	198	200	200	200	200
3				200	200	198	200	198	197
4							200	200	200
5							198	200	200
Total	400	398	399	600	598	595	998	998	997

TABLE 5.3: Class distribution of the **IMDb-proco** dataset.

Class	**IMDb**
High-revenue	572
Low-revenue	597
Total	1169

Third, the results are discussed in Section 5.4.3. Fourth, the computation time is investigated in Section 5.4.4. Finally, some extreme cases are studied in Section 5.4.5.

5.4.1 Datasets

The different classifiers are compared on 14 datasets that have been used previously for semi-supervised classification: nine **Newsgroups** datasets [153], the four universities **WebKB cocite** datasets [165, 262], and the **IMDb prodco** dataset [165]. The different datasets used for these comparisons are described in Section 5.4.1. Implementations and datasets are available at `http://www.isys.ucl.ac.be/staff/lebichot/research.htm` and `https://b-lebichot.github.io/`.

Newsgroups: The Newsgroups dataset is composed of about 20,000 unstructured documents, taken from 20 discussion groups (newsgroups) of the Usenet diffusion list. 20 Classes (or topics) were originally present in the dataset. For our experiments, nine subsets related to different topics are extracted from the original dataset, resulting in a total of nine different datasets. The datasets were built by sampling about 200 documents at random in each topic (three samples of two, three, and five classes, thus nine samples in total).

TABLE 5.4: Class distribution of the four **WebKB cocite** datasets.

Class	**Cornell**	**Texas**	**Washington**	**Wisconsin**
Course	54	51	170	83
Department	25	36	20	37
Faculty	62	50	44	37
Project	54	28	39	25
Staff	6	6	10	11
Student	145	163	151	155
Total	346	334	434	348
Majority class (%)	41.9	48.8	39.2	44.5

The repartition is listed in Table 5.2. The extraction process and the procedure used for building the graph are detailed in [258].

WebKB cocite: These datasets consist of sets of web pages gathered from four computer science departments (four datasets, one for each university), with each page manually labeled into one of six categories: course, department, faculty, project, staff, and student [165]. The pages are linked by co-citation (if x links to z and y links to z, then x and y are co-citing z), resulting in an undirected graph. The composition of the datasets is shown in Table 5.4.

IMDb-prodco: The collaborative Internet Movie Database (IMDb, [165]) has several applications such as making movie recommendations or movie category classification. The classification problem focuses on the prediction of whether the movie is a box-office hit or not. It contains a graph of movies linked together whenever they share the same production company and weight of an edge in the graph is the number of production companies that two movies have in common. The class distribution is shown in Table 5.3.

5.4.2 Experimental methodology

The classification accuracy is reported for several labeling rates (10%, 30%, 50%, 70%, 90%), i.e. proportions of nodes for which the label is known. The labels of remaining nodes are deleted during the modeling phase and are used as test data during the assessment phase. For each considered labeling rate, 5 random node label deletions were performed (5 runs). For each unlabeled node, the various classifiers predict the most suitable category. For each run, a 10-fold nested cross-validation is performed for tuning the parameters of the models. The external folds are obtained by 10 successive rotations of the nodes and the performance of one specific run is the average over these 10 folds.

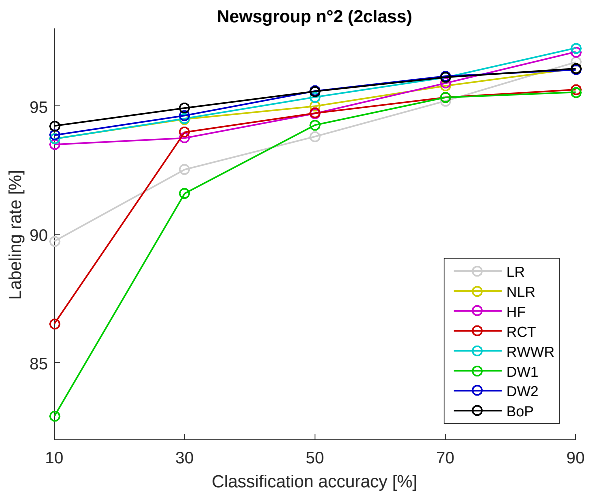


FIGURE 5.2: Classification accuracies in percents, averaged over 5 runs, obtained on partially labeled graphs. Results are reported for the eight methods (RL, RNL, RCT, HF, RWWR, DW1, DW2, BoP) and for five labeling rates (10%, 30%, 50%, 70%, 90%). This graphs shows the results obtained on the **NewsGroups** (NG2) dataset.

Moreover, for each fold of the external cross-validation, a 10-fold internal cross-validation is performed on the remaining labeled nodes in order to tune the hyperparameters of the classifiers (i.e. parameters α, λ and θ (see Table 5.1) – methods HF and DW1 do not have any hyperparameter). Thus, for each method and each labeling rate, the mean classification accuracy averaged on the 5 runs are reported.

5.4.3 Results & discussion

Comparative results for each method on the fourteen datasets are reported as follows: the results on the nine **NewsGroups** datasets, on the four **WebKB Cocite** datasets and on the **IMBd-prodco** dataset are shown in Table 5.5. The results of the second **NewsGroups** dataset are also reported as a plot on Figure 5.2 to visualize the typical relation between classification accuracy and labeling rate. For convenience, the acronyms of the tested methods are summarized in Table 5.1.

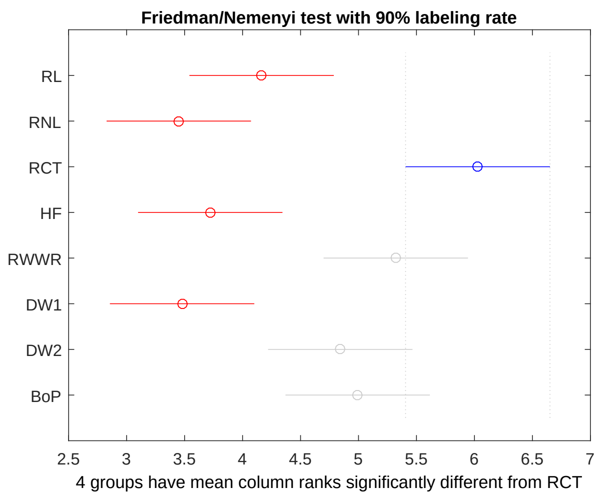


FIGURE 5.3: Mean rank (circles) and critical difference (plain line) of the Friedman/Nemenyi test. Results are reported for the eight methods (RL, RNL, RCT, HF, RWWR, DW1, DW2, BoP) and for a 90% labeling rate. The critical difference is 1.2546.

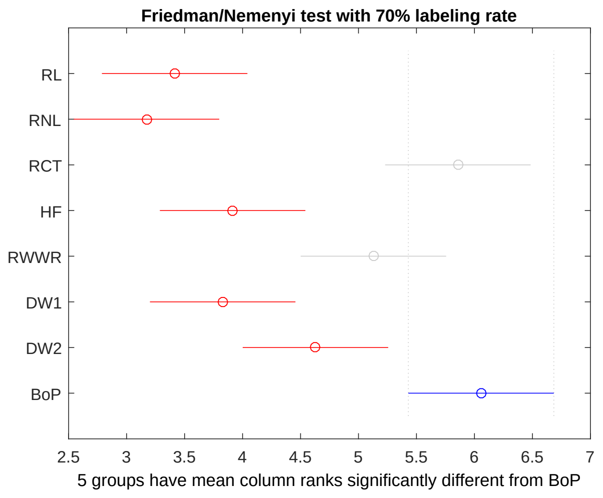


FIGURE 5.4: Friedman/Nemenyi test for a 70% labeling rate. See Figure 5.3 for details.

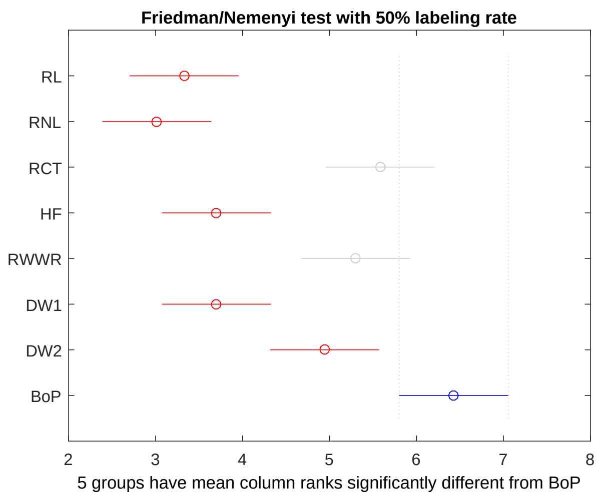


FIGURE 5.5: Friedman/Nemenyi test for a 50% labeling rate. See Figure 5.3 for details.

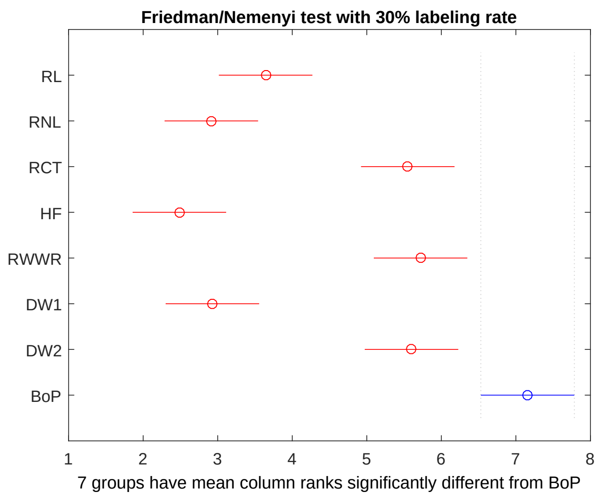


FIGURE 5.6: Friedman/Nemenyi test for a 30% labeling rate. See Figure 5.3 for details.

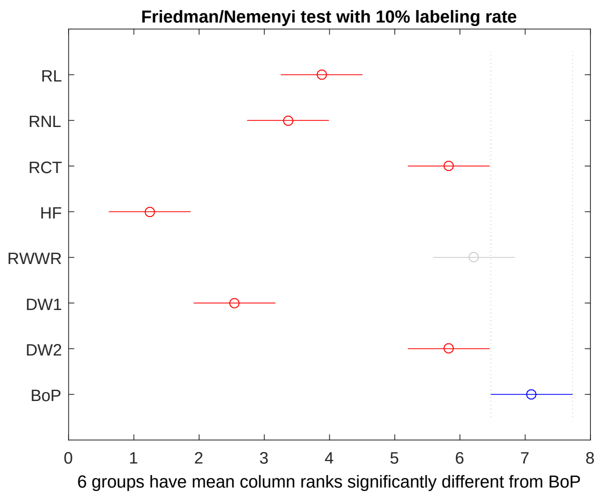


FIGURE 5.7: Friedman/Nemenyi test for a 10% labeling rate. See Figure 5.3 for details.

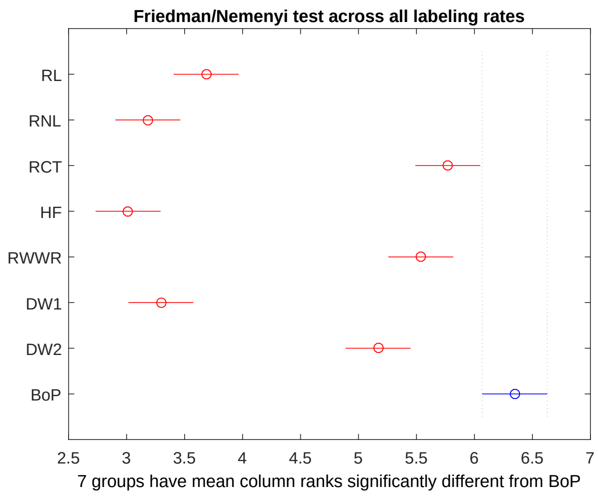


FIGURE 5.8: Friedman/Nemenyi test. Results are reported for the eight methods (RL, RNL, RCT, HF, RWWR, DW1, DW2, BoP) and for the five labeling rates together (10%, 30%, 50%, 70%, 90%). The Critical Difference is 0.5603.

TABLE 5.5: Classification accuracies in percents ± the standard deviation, averaged over the 5 runs, obtained on partially labeled graphs. Results are reported for the eight method, for the five labeling rates and all the tested datasets.

	l	**RL**	**RNL**	**RCT**	**HF**	**RWWR**	**DW1**	**DW2**	**BoP**
NG1	**90%**	98.90±0.22	**98.95±0.11**	97.90±0.22	97.40±0.38	97.90±0.22	96.95±0.21	98.05±0.33	98.35±0.22
	70%	98.02±0.37	**98.16±0.19**	97.09±0.13	97.12±0.28	97.07±0.08	96.30±0.74	97.78±0.12	97.63±0.04
	50%	97.75±0.39	**97.79±0.15**	97.23±0.15	96.60±0.65	96.93±0.10	95.46±0.76	97.47±0.07	97.54±0.22
	30%	95.19±4.10	97.50±0.22	97.03±0.04	95.94±0.61	97.02±0.08	94.27±0.55	**97.62±0.02**	97.48±0.15
	10%	95.90±1.96	96.83±0.06	96.95±0.11	87.43±1.03	**97.24±0.06**	88.26±0.29	97.20±0.00	97.11±0.16
NG2	**90%**	96.65±0.14	96.35±0.22	**97.30±0.33**	95.78±0.21	**97.30±0.21**	96.08±0.42	96.45±0.11	96.20±0.21
	70%	95.17±2.04	95.53±0.43	95.97±0.27	95.48±0.33	96.14±0.17	95.32±0.42	**96.17±0.11**	**96.17±0.05**
	50%	93.02±3.44	94.94±0.33	94.89±0.04	94.68±0.45	95.38±0.08	94.07±0.71	**95.56±0.03**	95.37±0.33
	30%	94.68±0.56	93.50±2.15	93.88±0.15	93.90±0.57	94.49±0.05	91.60±0.80	94.60±0.08	**94.92±0.15**
	10%	93.87±1.20	93.77±0.16	93.64±0.22	87.65±1.76	93.79±0.05	83.03±0.83	93.84±0.04	**94.03±0.43**
NG3	**90%**	96.15±0.14	95.55±0.11	97.45±0.27	96.79±0.21	97.50±0.00	96.94±0.57	95.90±0.22	**97.65±0.29**
	70%	96.03±0.04	96.14±0.16	97.06±0.18	96.67±0.44	97.07±0.08	96.89±0.44	95.99±0.04	**97.54±0.17**
	50%	96.03±0.03	96.17±0.46	96.86±0.10	96.57±0.24	96.76±0.12	96.66±0.42	95.98±0.00	**97.08±0.40**
	30%	95.86±0.10	95.71±0.11	96.44±0.17	95.83±0.10	**96.76±0.09**	95.65±0.17	95.94±0.02	96.70±0.31
	10%	90.93±2.09	94.68±0.11	96.12±0.09	83.52±1.31	**96.58±0.06**	91.96±0.45	95.82±0.00	96.26±0.28
NG4	**90%**	95.17±0.00	**95.97±0.07**	95.43±0.30	94.87±0.22	95.43±0.30	94.87±0.25	94.80±0.14	95.27±0.19
	70%	91.94±3.55	94.71±0.19	94.22±0.09	94.88±0.20	94.17±0.07	**95.04±0.40**	94.25±0.08	94.36±0.25
	50%	91.49±2.99	93.56±0.15	93.24±0.08	**94.42±0.27**	93.22±0.08	94.24±0.58	93.86±0.02	93.89±0.17
	30%	90.68±2.90	93.31±0.13	92.86±0.05	92.90±0.32	93.14±0.09	92.57±0.42	93.50±0.05	**93.52±0.12**
	10%	90.46±2.74	92.15±0.29	92.50±0.21	75.03±1.28	92.65±0.20	85.71±0.35	92.44±0.02	**92.97±0.29**
NG5	**90%**	95.47±0.07	95.67±0.00	95.87±0.22	95.06±0.26	**96.00±0.20**	94.86±0.28	95.83±0.12	95.50±0.35
	70%	94.70±0.20	94.43±0.00	94.81±0.16	94.78±0.26	**94.97±0.11**	94.53±0.34	94.27±0.13	94.54±0.22
	50%	91.91±2.71	92.60±0.00	93.42±0.24	93.90±0.19	94.42±0.12	93.70±0.21	93.74±0.05	**94.55±0.18**
	30%	90.08±2.68	90.39±0.07	92.62±0.36	91.43±0.25	93.61±0.21	91.71±0.19	92.96±0.02	**94.16±0.40**
	10%	87.10±5.04	85.05±0.58	91.21±0.33	77.50±0.43	92.52±0.23	84.11±0.30	92.54±0.01	**92.55±0.54**
NG6	**90%**	94.00±0.00	92.50±0.00	**96.37±0.14**	94.31±0.22	96.33±0.17	93.94±0.52	95.23±0.09	96.13±0.18
	70%	92.49±0.03	91.07±0.04	95.99±0.13	93.32±0.17	**96.13±0.14**	94.03±0.22	94.00±0.12	95.92±0.20
	50%	91.44±0.09	90.36±0.08	94.92±0.17	91.52±0.23	**95.35±0.05**	93.06±0.42	93.45±0.09	95.24±0.26
	30%	89.86±0.05	88.48±0.20	93.59±0.29	88.30±0.37	**94.50±0.12**	90.75±0.32	92.40±0.05	94.47±0.32
	10%	87.12±2.41	86.44±0.23	92.67±0.27	73.73±1.08	**93.58±0.17**	83.28±0.40	91.66±0.04	92.76±0.38
NG7	**90%**	93.10±0.10	91.96±0.23	**93.30±0.07**	92.28±0.29	93.26±0.05	92.34±0.46	93.28±0.19	93.22±0.04
	70%	92.29±0.08	91.18±0.34	92.37±0.10	91.30±0.25	92.35±0.09	91.66±0.29	**92.66±0.02**	92.59±0.11
	50%	90.58±1.65	90.35±0.01	91.85±0.09	89.90±0.17	91.86±0.12	90.46±0.30	**92.18±0.05**	91.70±0.23
	30%	89.19±3.13	89.52±0.46	90.75±0.17	87.22±0.22	**91.38±0.08**	88.43±0.18	91.28±0.04	91.12±0.16
	10%	88.10±1.17	88.72±0.15	89.95±0.21	69.55±0.78	90.76±0.07	80.10±0.23	90.03±0.03	**90.88±0.16**
NG8	**90%**	88.60±0.12	89.30±0.00	89.72±0.08	88.65±0.48	**89.72±0.13**	88.37±0.33	88.96±0.09	88.72±0.32
	70%	89.10±0.10	89.18±0.24	89.33±0.05	87.92±0.36	89.41±0.09	87.92±0.47	**89.52±0.07**	89.29±0.10
	50%	87.89±0.38	86.78±0.09	88.02±0.05	86.52±0.36	88.34±0.10	86.42±0.37	88.08±0.10	**88.50±0.13**
	30%	86.92±0.08	83.41±0.72	86.60±0.35	83.16±0.27	87.69±0.06	83.30±0.32	87.08±0.04	**88.05±0.11**
	10%	84.61±2.16	83.88±0.87	85.70±0.16	62.80±1.11	86.65±0.11	73.46±0.36	85.82±0.03	**87.33±0.26**
NG9	**90%**	87.52±0.15	87.38±0.04	87.26±0.15	**88.72±0.15**	87.40±0.12	87.88±0.18	87.70±0.07	87.64±0.15
	70%	86.44±0.05	87.11±0.00	87.30±0.11	87.19±0.26	87.39±0.13	87.01±0.32	86.47±0.06	**87.58±0.17**
	50%	85.40±0.13	85.99±0.00	86.85±0.15	85.82±0.27	**87.27±0.08**	85.80±0.11	85.85±0.08	87.01±0.19
	30%	84.05±0.44	83.20±0.15	86.16±0.14	82.21±0.50	86.60±0.03	82.91±0.26	85.01±0.02	**86.65±0.22**
	10%	81.64±0.71	79.87±0.58	84.81±0.41	69.06±0.68	**86.36±0.08**	72.82±0.33	84.07±0.10	85.66±0.41
Cornell	**90%**	59.42±1.33	52.95±0.86	**65.32±1.42**	63.18±0.27	56.42±0.14	60.74±0.51	60.21±0.57	60.11±1.11
	70%	59.04±0.76	52.17±1.31	**63.97±0.52**	60.58±1.00	57.84±0.24	58.19±0.67	59.14±0.33	61.50±2.31
	50%	55.03±1.35	46.80±1.52	**61.54±1.64**	56.88±0.75	56.20±0.12	54.64±1.03	56.88±0.24	61.15±2.20
	30%	50.09±2.97	44.27±0.16	**60.64±1.33**	50.31±0.45	54.80±0.29	51.31±0.42	55.93±0.40	60.14±2.99
	10%	47.98±0.55	43.12±0.25	56.98±1.12	42.81±0.26	57.17±0.21	48.02±1.35	58.31±0.07	**59.01±3.26**
Texas	**90%**	72.17±0.12	68.83±0.66	78.44±0.58	74.38±0.73	79.22±0.36	82.06±0.46	81.78±0.89	**82.22±0.39**
	70%	71.24±0.21	68.26±0.30	78.45±0.39	72.54±1.11	78.57±0.20	80.62±0.35	79.58±0.45	**80.88±0.69**
	50%	68.99±0.18	65.45±0.14	77.50±0.51	69.29±0.69	76.58±0.48	79.20±0.79	77.90±0.20	**79.64±0.76**
	30%	67.00±0.05	61.73±0.71	76.45±0.78	65.66±0.75	73.18±0.42	76.89±0.45	76.66±0.13	**78.06±0.23**
	10%	64.56±0.92	58.58±0.51	72.99±2.01	51.25±0.30	70.55±0.56	69.68±1.05	74.24±0.10	**75.40±1.93**
Washington	**90%**	68.53±0.20	63.56±0.00	**70.40±0.51**	66.99±0.72	61.96±0.10	61.50±0.64	58.31±0.12	64.49±0.19
	70%	62.39±1.83	63.65±0.00	**69.40±0.52**	65.76±0.62	59.74±0.37	59.87±0.69	55.46±0.09	64.31±0.67
	50%	62.11±1.21	64.38±0.00	**67.86±0.54**	63.84±0.62	58.76±0.62	56.36±1.79	55.43±0.12	60.82±0.87
	30%	60.44±1.29	64.74±0.33	**66.18±0.87**	59.97±1.01	57.72±0.53	53.40±1.65	55.67±0.15	59.86±1.85
	10%	51.62±2.84	59.87±3.48	**65.36±0.69**	42.41±0.52	54.30±1.22	45.85±1.16	53.00±0.31	61.66±0.61
Wisconsin	**90%**	71.42±0.14	67.84±0.34	74.21±0.19	75.08±0.43	73.42±0.00	**80.34±0.80**	77.26±0.35	73.53±0.14
	70%	70.45±0.17	67.33±0.09	75.21±0.16	73.61±0.22	71.79±0.09	**79.59±1.05**	75.48±0.26	74.35±0.08
	50%	69.25±0.06	66.93±0.11	75.62±0.27	72.00±0.18	69.09±0.20	**77.86±1.27**	75.32±0.30	75.12±0.67
	30%	68.12±0.00	66.31±0.06	**75.73±0.37**	68.83±0.45	69.40±0.36	73.93±1.07	74.49±0.60	75.44±0.26
	10%	67.29±0.00	63.70±1.22	**75.45±1.00**	53.85±0.97	63.48±2.21	65.53±0.39	72.65±0.61	73.89±1.29
IMDb	**90%**	83.37±0.15	83.10±1.32	83.51±0.20	76.89±2.65	83.64±0.20	50.85±0.13	**84.02±0.24**	82.81±0.20
	70%	74.28±2.90	66.59±3.55	**82.73±0.25**	55.49±2.73	82.44±0.21	50.58±0.26	81.71±0.10	81.80±0.26
	50%	64.48±4.58	59.26±1.71	81.72±0.17	51.04±0.05	**81.71±0.19**	52.48±4.20	81.10±0.03	81.18±0.25
	30%	53.71±3.09	50.66±0.03	80.78±0.18	50.99±0.04	80.75±0.30	57.95±5.33	80.12±1.31	**80.97±0.31**
	10%	50.60±0.00	50.61±0.01	79.64±0.31	50.98±0.02	79.58±0.14	73.93±2.52	80.03±0.03	**80.56±0.34**

The most selected values of the parameters for each method are also reported in Table 5.1. With each method, all the tested values were selected at least once. RNL, RCT, RWWR, and BoP selected a short range of parameters. Conversely, some methods, such as RL and DW2, selected a very wide range of parameters, and the different most represented values are not grouped.

The different classifiers have been compared across datasets through a Friedman test and a Nemenyi post-hoc test [74] (see Appendix A). The Friedman test is a non-parametric equivalent of the repeated-measures ANOVA. It ranks the methods for each dataset separately, the best algorithm getting the rank m, the second best rank $m-1, \dots$ Once the null hypothesis (the mean ranking of all methods is equal, meaning all classifiers are equivalent) is rejected with p-value < 0.05, the (non parametric) posthoc Nemenyi test is then computed. Notice that all Friedman tests were found to be positive. The Nemenyi test determines whether or not each method is significantly better (p-value less than 0.05 based on the 5 runs) to another.

Friedman/Nemenyi tests are reported for each labeling rate on Figure 5.3 to Figure 5.7. Furthermore, a Friedman/Nemenyi test across all labeling rate is presented on Figure 5.8.

We observe that the BoP classifier always achieved competitive results since it ranges among the top methods on all datasets. The BoP classifier actually tends to be the best algorithm for all labeling rates except for the 90% labeling rate, where it cannot be significantly be discriminated from most of other methods from Figure 5.3. Notice that the BoP classifier is significantly better than all other methods on Figures 5.6 (labeling rate equal to 30%) and 5.8 (all labeling rates). It is also almost the case for Figures 5.7 (labeling rate equal to 10%).

The RCT method is often second and is the best of the kernel-based classifiers (as suggested in [91]). The RWWR kernel also achieves good performance and is often close to RCT. This last method is also the best algorithm when the labeling rate is very high (90%).

Notice that RWWR, RCT, and DW2 largely outperform the other algorithms (besides BoP). However, it is difficult to figure out which of those three methods is the best, after BoP. It can be noticed that the DW2 version of the $\mathcal{D}$-walk is more competitive when the labeling rate is low and that it performs significantly better than the DW1 version, for almost all labeling rates.

From the fifth to the eighth position, the ranking is less clear since none of the methods is really better than the other. However, most of these methods (NR, RNL, HF, and DW1) are significantly worse than BoP, RCT, RWWR, and DW2. Notice also that the performance of DW1, and HF drops significantly when labeling rate decreases.

TABLE 5.6: Overview of cpu time in seconds needed to classify all the unlabeled nodes. Results are averaged on 20 runs. The CPU used was an Intel(R)Core(TM)i3 at 2.13 Ghz with 3072 Ko of cache size and 6 GB of RAM and the programming language is Matlab.

	RL	**RNL**	**RCT**	**HF**	**RWWR**	**DW1**	**DW2**	**BoP**
Exp1: 1000 nodes, 2 classes	0.0872	0.4937	0.0751	0.1680	0.3480	0.6826	0.6772	0.4997
Exp2: 2000 nodes, 2 classes	0.4616	3.4961	0.4225	0.9618	2.0441	4.5561	4.5858	3.0574
Exp3: 4000 nodes, 2 classes	2.8274	27.0695	2.5949	7.1481	14.1116	35.7161	36.0207	22.393
Ratio Exp2/Exp1	5.2935	7.0814	5.6258	5.7250	5.8739	6.6746	6.7717	6.1185
Ratio Exp3/Exp2	6.1252	7.7428	6.1418	7.4320	6.9036	7.8392	7.8548	7.3242
Exp2: 2000 nodes, 2 classes	0.4616	3.4961	0.4225	0.9618	2.0441	4.5561	4.5858	3.0574
Exp4: 2000 nodes, 4 classes	0.5011	3.4563	0.4036	1.2064	3.4249	8.5048	8.4003	3.2535
Exp5: 2000 nodes, 8 classes	0.4813	3.8449	0.4482	1.5748	6.0697	16.0031	16.3956	3.5868
Ratio Exp4/Exp2	1.0856	0.9886	0.9553	1.2543	1.6755	1.8667	1.8318	1.0641
Ratio Exp5/Exp4	0.9605	1.1124	1.1105	1.3054	1.7722	1.8817	1.9518	1.1024

5.4.4 Computation time

The computational tractability of a method is an important consideration to take into account. Table 5.6 provides a comparison of the running time of all methods. To explore computation time with respect to the number of nodes and the number of classes, artificial graphs with a certain number of classes have been created. For each method, 20 runs on each of the datasets are performed and the running time is recorded for each run. The 20 running times are averaged and results are reported in Table 5.6.

We observe that HF is one of the quickest methods, but sadly it is not competitive in terms of accuracy, as reported in Section (5.4.3). Notice that the two kernel methods, RL and RCT, have more or less the same computation time since the alignment is done once for all the classes. RNL, the last kernel method, is slower than RL, HF, and RCT (because of the time-consuming normalisation). After the HF and the kernel methods, the BoP classifier achieves competitive results compared with the remaining classifiers. The time augmentation when the graph size increases is similar for all methods (except for RL and RCT for which the augmentation is smaller). The cause is that all those methods require a matrix inversion: the complexity of such an operation is $\mathcal{O}(n^3)$ (where n is the number of nodes) and this is what can be observed from Table 5.6 (when the number of nodes doubles, the time is more or less multiplied by eight). But the BoP classifier has the same advantage as the kernel methods: its computation time does not increase strongly when the number of classes increases. This comes from the algorithm structure: Unlike RWWR, DW1, and DW2, the BoP classifier does not require a matrix inversion for each class. Furthermore, the matrix inversions (or linear systems of equations to solve) required for the BoP can be computed as far as the graph (through its adjacency matrix) is known,

TABLE 5.7: Classification accuracies in percents, averaged over 20 runs, obtained on partially labeled artificial graphs. Results are reported for the eight methods (RL, RNL, RCT, HF, RWWR, DW1, DW2, BoP). Only two labels is the case where only two labeled nodes per class are known. Imbalanced is the case where one of the classes is much more represented than the other (labeling rate is 50%): Major stands for the majority class and Minor stands for the minority class.

	RL	**RNL**	**RCT**	**HF**	**RWWR**	**DW1**	**DW2**	**BoP**
Only two labels	51.9 ± 1.4	52.6 ± 2.6	84.0 ± 2.5	50.0 ± 0.05	84.0 ± 2.5	51.9 ± 0.5	67.3 ± 2.8	83.0 ± 3.1
Imbalanced: Major	98.2 ± 0.04	98.2 ± 0.04	98.1 ± 0.08	99.9 ± 0.04	95.5 ± 0.03	88.7 ± 7.0	93.8 ± 2.5	96.8 ± 0.4
Imbalanced: Minor	42.4 ± 4.2	43.8 ± 3.3	32.2 ± 2.0	1.9 ± 2.4	11.6 ± 10.5	38.5 ± 0.0	17.2 ± 2.7	11.9 ± 2.5

which is not the case with kernel methods. This is a good property for BoP, since it means that rows 1 to 6 of Algorithm 1 can be pre-computed once for all folds in the cross-validation. Finally, the space complexity is $\mathcal{O}(n^2)$ for all the methods.

5.4.5 Extreme cases

In this section, two extreme classification cases are considered. First, what happens if only one or two labeled data points are available? As described in Section 5.3, the BoP classifier requires at least two nodes for computing the BoP group betweenness. We performed a small experiment on the first **News-Groups** NG1 dataset. The parameters were tuned by a 10-fold cross-validation and 20 runs were averaged. The classification accuracies are reported in Table 5.7. Only the RCT, RWWR, and BoP classifiers remain competitive for this first extreme case.

Secondly, let us consider the case where the classes are imbalanced. The following experiment was designed to study this other extreme case. The classes of the well-known **industry-yh** dataset [87] were merged to get two classes: the majority class with 1768 nodes and the minority class with only 30 nodes (this represents 1.67% for the minority class). Once again, the parameters were tuned by a 10-fold cross-validation and 20 runs were averaged. The classification accuracies for the two classes are reported in Table 5.7. The best methods to identify the minority class are RL and RNL, followed by RWWR and RCT. In this particular case, the BoP classifier is outperforms by all methods except HF and RWWR. Therefore, it seems to achieve poor results in the case of highly unbalanced classes.

5.5 Conclusion

This chapter investigates an application of the bag-of-paths framework viewing the graph as a virtual bag from which paths are drawn according to a Boltzmann sampling distribution.

In particular, it introduces a novel algorithm for graph-based semi-supervised classification through the bag-of-paths group betweenness, or BoP for short (described in Section 5.3). The algorithm sums the a posteriori probabilities of drawing a path visiting a given node of interest according to a biased sampling distribution, and this sum defines our BoP betweenness measure. The Gibbs-Boltzmann sampling distribution depends on a parameter, θ, gradually biasing the distribution towards shorter paths: when θ is large, only little exploration is performed and only the shortest paths are considered, while when θ is small (close to 0^+), longer paths are considered and are sampled according to the product of the transition probabilities p_{ij}^{ref} along the path (a natural random walk).

Experiments on real-world datasets show that the BoP method outperforms the other considered approaches when only a few labeled nodes are available. When more nodes are labeled, the BoP method is still competitive. Its computation time is comparable in most of the cases.

The biggest drawback of the BoP classifier is that it is not applicable as-is on large graphs. A key question is therefore: Is it possible to enhance the classifier to be computationally more tracktable on large graphs? Investigations have been carried in that direction but no convincing solution has been found so far.

Another interesting issue is how to combine the information provided by the graph and the features on the nodes in a clever, preferably optimal, way. The interest of including node features can be assessed experimentally. This question is investigated in Chapter 6.

Chapter 6

Graph-based semi-supervised classification with additional information on nodes

Nowadays, with the increasing volume of generated data, for instance by internet and social networks, there is a need for efficient ways to infer useful information from those network-based data. Moreover, these data can take several different forms and, in that case, it would be useful to use these alternative views in the prediction model.

In this chapter, we focus our attention on semi-supervised classification (see Section 3.2) using **both regular tabular data** defined on nodes and **structural information coming from graphs** or networks. We investigate several ways of combining the structural information provided by a network of interactions between objects (for instance interactions between members of an online social network) and information associated to these different objects (for instance the gender of the person, her age,...) for solving objects (nodes) classification problems. Another possibility is to use semi-supervised learning based on a network structure only (discarding the features), as studied in Chapter 5 (see for instance [2, 58, 90, 122, 149, 165, 218, 222, 266, 268]).

Of course, as discussed in [89] (see, e.g., [151] for a survey), many different approaches have been developed for information fusion in machine learning, pattern recognition, and applied statistics. This includes [89] simple weighted averages (see, e.g., [65, 127]), Bayesian fusion (see, e.g., [65, 127]), majority vote (see, e.g., [61, 143, 152]), models coming from uncertainty reasoning [148] (see, e.g., [80]), standard multivariate statistical analysis techniques such as correspondence analysis [173], maximum entropy modeling (see, e.g., [158, 180, 89]) . . . This is also an emerging field of machine learning called multi-view learning [223, 261].

This problem has numerous applications such as classification of individuals in social networks, categorization of linked documents (e.g. patents or scientific papers), or protein function prediction, to name a few. In this kind of application (as in many others), unlabeled data are usually available in large quantities and are easy to collect: friendship links can be recorded on Facebook, text documents can be crawled from the internet and DNA sequences of proteins are readily available from gene databases.

In this chapter, we investigate experimentally various models combining information on the nodes of the graph and the graph structure. Indeed, it has been shown that network information can improve significantly prediction accuracy in a number of contexts [121, 165]. A total of 16 classification algorithms using various combinations of data sources, mainly described in [90], are compared. The different considered algorithms are detailed in Section 6.3.

A standard support vector machine (SVM) classifier is used as a baseline algorithm, but we also investigated the ridge logistic regression classifier. The results and conclusions obtained with this second classification model were similar to the SVM and are therefore not reported in this work.

In short, the main questions investigated in this work are:

- ▶ Does the combination of features on nodes and network structure works better than using the features only?
- ▶ Does the combination of features on nodes and network structure works better than using the graph structure only?
- ▶ Which classifier performs best on network structure alone, without considering features on nodes?
- ▶ Which classifier performs best when combining information, that is, using network structure with features on nodes?

Finally, this comparison leads to some general conclusions and advices when tackling classification problems on network data with node features.

In summary, this chapter (and the related paper, see Chapter 1) has four main contributions:

- ▶ The chapter reviews different algorithms used for learning from both a graph structure and node features, mainly following [90]. All considered algorithms are transductive, except the SVM baseline.
- ▶ An empirical comparison of those algorithms is performed on ten real-world datasets.
- ▶ We investigate the effect of extracting features from the graph structure (and some well-known indicators in spatial statistics) in a classification context.

- ▶ Finally, this comparison leads to some general conclusions and advices to tackle graph-based classification tasks.

The remaining of this chapter is organized as follows. Section 6.1 provides some background and notation. Section 6.2 investigates related work. Section 6.3 introduces the investigated classification methods. Then, Section 6.4 presents the experimental methodology and the results. Finally, Section 6.5 concludes the chapter.

6.1 Background and notation

This section aims to recall the theoretical background and notation used in this chapter. Most of them has already been presented in Chapter 2.

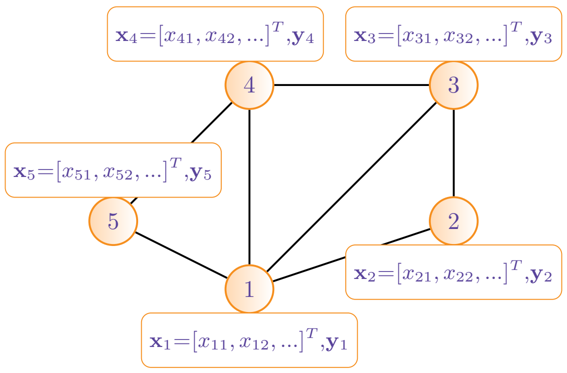
In this chapter, we consider a weighted, undirected, strongly connected, graph or network G (with no self-loop) containing a set $\mathcal{V}$ of n vertices. Let us just recall the $n \times n$ **adjacency matrix A** of the graph (containing non-negative affinities between nodes), the **Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$** and the **random walk transition probabilities matrix P** with elements $p_{ij} = a_{ij} / \sum_{j'=1}^{n} a_{ij'}$ (see Equation (2.8)): the random walker chooses to follow an edge with a likelihood proportional to the affinity, therefore favoring edges with a large affinity).

Moreover, we consider that each of the nodes of G has the same set of m features, or attributes, with no missing values. The column vector $\mathbf{x}_i$ contains the values of the m features of node i and x_{ij} states for the value of feature j taken by node i. Moreover, $\mathbf{X}_{\text{features}}$, or simply $\mathbf{X}$, refer to the $n \times m$ data matrix containing the elements x_{ij}.

As a last notation we define $\mathbf{y}$, the column vector containing the class labels of the nodes. More precisely, $\mathbf{y}^c$ is a binary vector indicating whether or not each node belongs to class number c. That is, $\mathbf{y}_i^c$ is equal to one if node i belongs to class c, and zero otherwise.

Recall that the purpose of the classification task is to predict the class of the unlabeled data (in a transductive setting), or to predict new test data (in an inductive setting), while knowing the structure of the graph G, the values of the features $\mathbf{X}$ for all the nodes of G and the class labels $\mathbf{y}^c$ on the labeled nodes only for each c. As already mentioned, our baseline classifier based on features is a linear support vector machines (SVM).

As stated in Section 3.3, semi-supervised classification comes in two different settings: inductive and transductive [268]. The goal of the former setting is to predict the labels of future test data, unknown when fitting the model, while the second is to classify (only) the unlabeled instances of the training sample. All classifiers in this chapter are transductive, except the SVM baseline.



$\mathbf{x}_4=[x_{41},x_{42},...]^T$,$\mathbf{y}_4$
$\mathbf{x}_3=[x_{31},x_{32},...]^T$,$\mathbf{y}_3$
4
3
$\mathbf{x}_5=[x_{51},x_{52},...]^T$,$\mathbf{y}_5$
5
2
$\mathbf{x}_2=[x_{21},x_{22},...]^T$,$\mathbf{y}_2$
1
$\mathbf{x}_1=[x_{11},x_{12},...]^T$,$\mathbf{y}_1$

FIGURE 6.1: Example of graph with additional node information. Each node is characterized by a feature vector and a class label.

Finally, the structure of the data should also have been of a different type. We focus on a particular data structure: we assume that our dataset takes the form of a network with features associated to the nodes. Nodes are the samples of our dataset and edges between these nodes represent a given type of interaction or relation between these samples (like a friendship relation on Facebook). For each node, a number of features or attributes characterizing it is also available (see Figure 6.1 for an example). Other data structures exist but are not studied in this thesis; for instance:

- Different types of nodes can be present, with different types of features sets describing them.
- Different types of relations can link the different nodes.

6.2 Related work

The 16 investigated models are presented in the next Section 6.3. In addition to those models, other approaches exist.

For example [28, 118] use a standard ridge regression model complemented by a Laplacian regularization term, which has been called the Laplacian regularized least squares. This option was investigated but provided poor results compared to reported models (therefore not reported).

Note that using a logistic ridge regression as the base classifier (instead of a support vector machine) was also investigated in this work but results are not reported here for conciseness as it provided performances similar to the SVM.

Laplacian support vector machines (LapSVMs) extend the SVM classifier in order to take the structure of the network into account. They exploit both the information on the nodes and the graph structure in order to categorize the nodes with the use of its Laplacian matrix (see Section 6.1). To this end, [28] proposed to add a graph Laplacian regularization term to the traditional SVM cost function in order to obtain a semi-supervised version of this model. A Matlab toolbox for this model is available but provided poor results in terms of performance and tractability. This model was therefore not included in our comparisons.

Chakrabarti et al. [56] developed, in the context of patents classification, a naive Bayes model in the presence of structural autocorrelation. The main idea is to use a naive Bayes classifier combining both feature information on the nodes and structural information by making some independence assumptions. However, we found that this procedure is very time consuming, even for small-size networks, and decided to not include it in the present work as results were impossible to obtain on the larger datasets.

Also note that various semi-supervised classifiers based on network data only (features on nodes are not available) were also developed [165, 266]. The interested reader is invited to consult, e.g., [156, 2, 58, 122, 218, 222, 266, 268] (and included references), focused on this topic, for comprehensive reviews. Finally, an interesting survey and a comparative experiment of related methods, but more focused on relational learning, can be found in [165].

Finally, our problem is a particular case of the more general multi-view learning framework: Multi-view learning is an emerging direction in machine learning which considers learning with multiple views to improve the generalization performance [223, 261]. Also known as data fusion, it learns from multiple feature sets. In our particular case, we study the problem of learning from two information sources: on the one hand features and on the other hand a single graph.

Multi-view learning methods are divided into three major categories : co-training style algorithms, co-regularization style algorithms and margin-consistency style algorithms [223, 261]. Some of these multi-view algorithms are investigated.

- Co-training is historically the first family of multi-view algorithms: the classifier is trained alternately on two distinct views with confident labels for the unlabeled data (see [41] for details). Examples are co-EM, co-testing, and robust co-training [261].

- Co-regularization algorithms make sure that data from multiple views are consistent by adding a regularization term in the classifier objective function: the disagreement between the discriminant functions of the two views. Examples are sparse multi-view SVMs, multi-view TSVMs, multi-view Laplacian SVMs, and multi-view Laplacian TSVMs [261].
- Margin-consistency style algorithms ensure the latent consistency of classification results from multiple views by using the framework of maximize entropy discrimination (MED, see [223, 261] for details).

6.3 Description of relevant classification methods

The different classification methods compared in this work are briefly presented in this section, which is largely inspired by [90]. For a more thorough presentation, see the provided references to the original works or [90]. The classification models are sorted into different families: graph embedding-based classifiers, extensions of feature-based classifiers, graph-based classifiers, and multi-view learning.

All these methods rely on a standard, strong, assumption about the distribution of the labels in the graph: it is assumed that **neighboring nodes are likely to belong to the same class** and thus are likely to share the same class label (see, e.g., [149, 90, 218]). This assumption, called **homophily**, **associativity**, **local consistency**, or **structural autocorrelation** is discussed in Section 3.2. This hypothesis can be tested by using some autocorrelation measures widely used in spatial statistics (see Section 6.4.4). Moreover, the experiments show that if the assumption is not verified, the methods exploiting the graph structure do not bring any useful information to classify the nodes.

6.3.1 Graph embedding-based classifiers

A first interesting way to combine information from the features on the nodes and from the graph structure is to perform a **graph embedding** projecting the nodes of the graph into a low-dimensional space (an embedding space) preserving as much as possible its structural information, and then use the coordinates of the projected nodes as **additional features** in a standard classification model, such as a logistic regression or a support vector machine.

This procedure has been proposed, e.g., in the field of spatial statistics for ecological modeling [45, 79, 172], but also more recently in data mining [228, 229, 230, 260]. While many graph embedding techniques could be used, [79] suggests to exploit Moran's or Geary's index of spatial autocorrelation in order to compute the embedding.

Let us briefly develop their approach by closely following [90] (see this reference for more information). Moran's I and Geary's c (see, e.g., [113, 192, 246, 247]) are two coefficients commonly used in spatial statistics in order to test the hypothesis of spatial autocorrelation of a numerical quantity defined on the nodes. This interesting property is investigated on the datasets used in the experimental section, in the context of semisupervised classification (see, e.g., Table 6.8). Four different possibilities are considered to extract features from the graph structure: maximizing Moran's I, minimizing Geary's c, local principal component analysis and maximizing the bag-of-paths (BoP) modularity. Experimentally, we observed that some datasets are more "graph-driven": the network structure conveys important information for predicting the class labels. As we will see later, the four values of this section can also be used to assert this (see Table 6.8).

Maximizing Moran's index

Moran's I [176, 177] is given by

$$I(\mathbf{x}) = \frac{n}{a_{\bullet\bullet}} \frac{\sum_{i,j=1}^{n} a_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_{i'=1}^{n}(x_{i'} - \bar{x})^2} \tag{6.1}$$

where x_i and x_j are the values observed on nodes i and j respectively, for a considered quantity defined on the nodes (for instance the age of the person in a social network). The column vector $\mathbf{x}$ is the vector containing the values x_i on the nodes and $\bar{x}$ is the average value of $\mathbf{x}$. Then, $a_{\bullet\bullet}$ is simply the sum of all entries of $\mathbf{A}$ – the volume of the graph.

$I(\mathbf{x})$ can be interpreted as a correlation coefficient similar to the Pearson correlation coefficient [113, 192, 246, 247]. The numerator is a measure of covariance among the neighboring x_i in G, while the denominator is a measure of variance. It is a common misconception that I is in the interval $[-1, +1]$. Instead, the upper and lower bound depends on n, $a_{\bullet\bullet}$, but also on the maximum and minimum eigenvalues of $\mathbf{A}$ (see [134] for details). A value close to $I_0 = -1/(n-1) \approx 0$ [134] indicates no evidence of autocorrelation, a larger value indicates positive autocorrelation and a smaller value indicates negative autocorrelation (autocorrelation means that neighboring nodes tend to take similar values).

In matrix form, Equation (6.1) can be rewritten as

$$I(\mathbf{x}) = \frac{n}{a_{\bullet\bullet}} \frac{\mathbf{x}^{\mathrm{T}}\mathbf{HAHx}}{\mathbf{x}^{\mathrm{T}}\mathbf{Hx}} \tag{6.2}$$

where $\mathbf{H} = (\mathbf{I} - \mathbf{E}/n)$ is the centering matrix [169] and $\mathbf{E}$ is a matrix full of ones. Note that the centering matrix is idempotent, $\mathbf{HH} = \mathbf{H}$.

The objective is now to find the scores $\mathbf{x}$ that achieve the largest autocorrelation, as defined by Moran's index. This corresponds to the values that most explain the structure of G. It can be obtained by setting the gradient equal to zero; we then obtain the following generalized eigensystem:

$$\mathbf{HAHx}' = \lambda \mathbf{x}', \text{ and then } \mathbf{x} = \mathbf{Hx}' \tag{6.3}$$

The idea is thus to extract the first eigenvector $\mathbf{x}_1$ of the centered adjacency matrix (6.3) corresponding to the largest eigenvalue λ_1 and then to compute the second-largest eigenvector, $\mathbf{x}_2$, orthogonal to $\mathbf{x}_1, \ldots$ The eigenvalues λ_i are proportional to the corresponding explained Moran's $I(\mathbf{x})$.

The p largest centered eigenvectors of (6.3) are thus extracted and then used as additional p features for a supervised classification model (here a SVM). In other words, $\mathbf{X}_{\text{Moran}} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p]^T$ is a new data matrix, capturing the structural information of G, that can be concatenated to the feature-based data matrix $\mathbf{X}_{\text{features}}$, forming the extended data matrix $[\mathbf{X}_{\text{features}}, \mathbf{X}_{\text{Moran}}]$.

Minimizing Geary's constant

On the other hand, Geary's c [103] is another estimate of autocorrelation given by

$$c(\mathbf{x}) = \frac{(n-1)}{2\,a_{\bullet\bullet}} \frac{\sum_{i,j=1}^{n} a_{ij}(x_i - x_j)^2}{\sum_{i'=1}^{n} (x_{i'} - \bar{x})^2} \tag{6.4}$$

and is related to Moran's I. However, while Moran's I considers a covariance between neighboring nodes, Geary's c considers distances between values on pairs of neighboring nodes. Once again, lower and upper bounds are often assumed to be respectively 0 and 2 with 0 indicating perfect positive autocorrelation and 2 indicating perfect negative autocorrelation [172, 192, 247]. However, as for Moran's I, [134] shows that the bounds are more complex and actually depend on n, $a_{\bullet\bullet}$, and the maximum and minimum eigenvalues of $\mathbf{A}$. $c = 1$ indicates no evidence of autocorrelation.

In matrix form, Geary's c can be rewritten as

$$c(\mathbf{x}) = \frac{(n-1)}{2a_{\bullet\bullet}} \frac{\mathbf{x}^{\mathrm{T}}\mathbf{Lx}}{\mathbf{x}^{\mathrm{T}}\mathbf{Hx}}. \tag{6.5}$$

This time, the objective is to find the score vector minimizing Geary's c. By proceeding as for Moran's I, we find that minimizing $c(\mathbf{x})$ aims to compute the p lowest non-trivial eigenvectors of the Laplacian matrix:

$$\mathbf{Lx} = \lambda \mathbf{Hx} \tag{6.6}$$

and then use these eigenvectors as additional p features in a classification model. We therefore end up with the problem of computing the lowest eigenvectors of the Laplacian matrix, which also appears in spectral clustering (ratio cut, see, e.g., [243, 90, 183]).

Geary's c has a computational advantage over Moran's I: the Laplacian matrix is usually sparse, which is not the case for Moran's I. Moreover, note that since the Laplacian matrix $\mathbf{L}$ is centered, any non-trivial solution of $\mathbf{Lx} = \lambda\mathbf{x}$ is also a solution of Equation (6.6).

Local principal component analysis

In [29, 154], the authors propose to use a measure of local, structural, association between nodes, the **contiguity ratio** defined as

$$cr(\mathbf{x}) = \frac{\sum_{i=1}^{n}(x_i - m_i)^2}{\sum_{i'=1}^{n}(x_{i'} - \bar{x})^2}, \text{ with } m_i = \sum_{j \in \mathcal{N}(i)} p_{ij} x_j \tag{6.7}$$

and m_i is the average value observed on the neighbors of i, $\mathcal{N}(i)$. As for Geary's index, the value is close to zero when there is a strong structural association. However, there are no clear bounds indicating no structural association or negative correlation [154].

The numerator of Equation (6.7) is the mean squared difference between the value on a node and the average of its neighboring values; it is called the local variance in [154]. The denominator is the standard sample variance. In matrix form,

$$= \frac{\mathbf{x}^{\mathrm{T}}(\mathbf{I} - \mathbf{P})^{\mathrm{T}}(\mathbf{I} - \mathbf{P})\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}} \tag{6.8}$$

Proceeding as for Geary and Moran's indexes, minimizing $cr(\mathbf{x})$ aims to solve

$$(\mathbf{I} - \mathbf{P})^{\mathrm{T}}(\mathbf{I} - \mathbf{P})\mathbf{x} = \lambda\mathbf{H}\mathbf{x} \tag{6.9}$$

Here again, eigenvectors corresponding to the smallest non-trivial eigenvalues of the eigensystem (6.9) are extracted. This procedure is also referred to as **local principal component** analysis in [154].

Bag-of-paths modularity

For this algorithm, we also compute a number of structural features, but now derived from the modularity measure (which was introduced by Newman and co-workers in [184, 182, 183]) redefined in the bag-of-paths (BoP) framework [75], and concatenate them to the node features, $[\mathbf{X}_{\text{features}}, \mathbf{X}_{\text{BoPMod}}]$. Again, a SVM is then used to classify all unlabeled nodes. Indeed, it has been shown that using the dominant eigenvectors of the BoP modularity matrix provides better performances than using the eigenvectors of the standard modularity matrix [75]. The results for the standard modularity matrix are therefore not reported in this work.

It can be shown (see [75] for details) that the BoP modularity matrix is equal to

$$\mathbf{Q}_{\text{BoP}} = \mathbf{Z} - \frac{(\mathbf{Z}\mathbf{e})(\mathbf{e}^T\mathbf{Z})}{\mathbf{e}^T\mathbf{Z}\mathbf{e}} \tag{6.10}$$

where $\mathbf{Z}$ is the fundamental bag-of-paths $n \times n$ matrix and $\mathbf{e}$ is a length n column vector full of ones. Then as for Moran's I and Geary's c, an eigensystem

$$\mathbf{Q}_{\text{BoP}}\mathbf{x} = \lambda\mathbf{x} \tag{6.11}$$

must be solved and the largest eigenvectors are used as new, additional, structural, features.

6.3.2 Extensions of standard feature-based classifiers

These techniques rely on extensions of standard feature-based classifiers (for instance a logistic regression model or a support vector machine). The extension is defined in order to take the network structure into account. As before, the discussion is based on [90].

The AutoSVM: taking autocovariates into account

This model is also known as the **autologistic** or **autologit** model [36, 179, 16, 163], and is frequently used in the spatial statistics and biostatistics fields.

Note that, as a SVM is used as base classifier in this work, we adapted this model (instead of the logistic regression in [16]) in order to take the graph structure into account.

The method is based on the quantity $ac_i^c = \sum_{j \in \mathcal{N}(i)} p_{ij} \hat{y}_j^c$, where $\hat{y}_j^c$ is the predicted membership of node j, called the **autocovariate** in [16] (other forms are possible, see [179, 16]). It corresponds to the weighted averaged membership to class c within the neighborhood of i: it indicates to what extent neighbors of i belong to class c. The assumption is that node i has a higher chance to belong to class c if its neighbors also belong to that class.

However, since the predicted value $\hat{y}_j^c$ depends on the occurrence of the predicted value on other nodes, fitting the model is not straightforward. For the autologistic model, it goes through the maximization of the (pseudo-) likelihood (see for example [189, 36]), but we will consider a simpler alternative [16] which uses a kind of expectation-maximization-like heuristics (EM, see, e.g. [73, 171]), and is easy to adapt to our SVM classifier.

Following [90], here is a summary of the estimation procedure proposed in [16]:

1. At $t = 0$, initialize the predicted class memberships $\hat{y}_i^c(t = 0)$ of the unlabeled nodes by a standard SVM depending on the feature vectors only, from which we disregard the structural information (the information about neighbors' labels). For the labeled nodes, the membership values are of course not modified.
2. Compute the current values of the autocovariates, $ac_i^c = \sum_{j \in \mathcal{N}(i)} p_{ij} \hat{y}_j^c(t)$, for all nodes.
3. Train a so-called **autoSVM** model based on these current autocovariate values as well as the features on nodes, providing parameter estimates $\hat{\mathbf{w}}^c$.
4. Compute the new predicted class memberships $\hat{y}_i^c(t+1)$ of the set of unlabeled nodes from the fitted autoSVM model. After having considered all the unlabeled nodes, we have the new predicted values $\hat{y}_i^c(t+1)$.
5. Steps 2 to 4 are iterated until convergence of the predicted membership values $\hat{y}_i^c(t)$.

Double kernel SVM

Here, we describe another simple way of combining the information coming from features on nodes and graph structure. The basic idea ([206, 90]) is to

1. Compute a $n \times n$ kernel matrix based on node features [211, 217], for instance a linear kernel or a gaussian kernel.
2. Compute a $n \times n$ kernel matrix on the graph [91, 90, 102, 217], for instance the regularized commute-time kernel (see Section 6.4.2).
3. Fit a SVM based on these two combined kernels.

Then, by using the kernel trick, everything happens as if the new data matrix is

$$\mathbf{X}_{\text{new}} = [\mathbf{K}_{\mathbf{A}}, \mathbf{K}_{\mathbf{X}}] \tag{6.12}$$

where $\mathbf{K}_{\mathbf{A}}$ is a kernel on a graph and $\mathbf{K}_{\mathbf{X}} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$ is the kernel matrix associated to the features on the nodes (see [90] for details). Then, we can fit a SVM classifier based on this new data matrix and the labeled nodes.

A spatial autoregressive model

This model is a spatial extension of a standard regression model [157] and is well known in spatial econometrics. This extended model assumes that the predicted vector of class memberships $\hat{\mathbf{y}}^c$ is generated in each class c according to

$$\hat{\mathbf{y}}^c = \rho \mathbf{P} \hat{\mathbf{y}}^c + \mathbf{X}\mathbf{w}^c + \boldsymbol{\epsilon} \tag{6.13}$$

where $\mathbf{w}^c$ is the usual parameter vector, ρ is a scalar parameter introduced to account for the structural dependency through $\mathbf{P}$ and $\boldsymbol{\epsilon}$ is an error term. This model is somehow related to the previously introduced AutoSVM model. Obviously if ρ is equal to zero, there is no structural dependency and the model reduces to a standard linear regression model. Lesage's Econometrics Matlab toolbox was used for the implementation of this model [157]; see this reference for more information.

6.3.3 Graph-based classifiers

We also investigate some semi-supervised methods based on the graph structure only (no node feature exists or features are simply not taken into account). We selected the techniques performing best in a series of previous experimental comparisons [91, 156, 168]. As already discussed, they rely on some strong assumptions about the distribution of labels: that neighboring nodes are likely to share the same class label [58].

The bag-of-paths group betweenness classifier

This model (see Chapter 5 and [156]), inspired by [94, 208], considers a bag containing all the possible paths between pairs of nodes in G. Then, a Boltzmann distribution, depending on a temperature parameter T, is defined on the set of paths such that long (high-cost) paths have a low probability of being drawn from the bag, while short (low-cost) paths have a high probability of being drawn. This model is described in Section 5.3.

A sum-of-similarities based on the regularized commute time kernel

We also investigate a classification procedure based on a simple alignment with the regularized commute time kernel (RCT) $\mathbf{K}_{\text{RCT}}$, a **sum-of-similarities** (or **kernel alignment**) defined by $\mathbf{K}_{\text{RCT}}\,\mathbf{y}^c$, with $\mathbf{K}_{\text{RCT}} = (\mathbf{D} - \alpha\mathbf{A})^{-1}$ [265, 91, 90]. This expression quantifies to what extent each node is close (in terms of the similarity provided by the regularized commute time kernel) to class c. This similarity is computed for each class c in turn. Then, each node is assigned to the class showing the largest sum of similarities. It corresponds to a variant of the k nearest neighbors classifier when dealing with a similarity matrix instead of distances.

Element i, j of this kernel can be interpreted as the discounted cumulated probability of visiting node j when starting from node i. The (scalar) parameter $\alpha \in [0, 1]$ corresponds to a killed random walk where the random walker has a $(1 - \alpha)$ probability of disappearing at each step. Other graph kernels could be used in a sum-of-similarities setting [90] but this one consistently provided good results in comparative studies of graph-based semi-supervised classification techniques [91, 168, 94].

6.3.4 Multi-view learning

Finally, Multi-view learning is also considered. The three classes of Multi-view learning were recalled in Section 6.2. The original co-training algorithms [41] was re-implemented based on SVMs.

Co-training

Given a set $\mathcal{L}$ of labeled samples/nodes and a set $\mathcal{U}$ of unlabeled samples/nodes, the algorithm iterates the following two-steps procedure. First, use $\mathcal{L}$ to train two distinct classifiers: here one is based on the features only and the second is based on a kernel extracted from the graph only. Second, allow each of the classifier to label a small subset of $\mathcal{U}$ with the highest posteriors provided by both views (the most "certain" nodes; here, distances to the hyperplane were used instead), then update $\mathcal{L}$ and $\mathcal{U}$. When all unlabeled nodes have been labeled, the procedure stops. See [41] for more details.

Kernel canonical correlation analysis

Kernel canonical correlation analysis [116] is an kernel extension of standard canonical correlation analysis. The idea is to project the data in a new space, and constrain the multiple transformed feature sets to be as close as possible, while regularizing the self covariance of each transformed feature sets to be small enough. The goal is to find projection vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ such that

$$\frac{\mathrm{cov}(\mathbf{X}\mathbf{w}_1, \mathbf{Y}\mathbf{w}_2)}{\sqrt{\mathrm{var}(\mathbf{X}\mathbf{w}_1)\mathrm{var}(\mathbf{Y}\mathbf{w}_2)}} \tag{6.14}$$

is maximal. $\mathrm{var}()$ and $\mathrm{cov}()$ are respectively the variance and covariance measures and $\mathbf{X}$ and $\mathbf{Y}$ are two kernel-based views. In our case, $\mathbf{X}$ corresponds to the regular features and $\mathbf{Y}$ corresponds to a kernel built from the graph (here, we chose the RCT kernel with $\alpha = 0.85$). See [261] or [116] for more details. Notice that this method is related to the double kernel SVM of Section 6.3.2.

TABLE 6.1: Class distribution of the four **WebKB** datasets.

Class	**Cornell** (DB1)	**Texas** (DB2)	**Washington** (DB3)	**Wisconsin** (DB4)
Course	42	33	59	70
Faculty	32	30	25	32
Student	83	101	103	118
Project + staff	38	19	28	31
Total	195	183	230	251
Majority class (%)	42.6	55.2	44.8	47.0
Number of features	1704	1704	1704	1704

TABLE 6.2: Class distribution of the three **Ego facebook** datasets.

Class	**FB 107** (DB5)	**FB 1684** (DB6)	**FB 1912** (DB7)
Main group	524	568	737
Other groups	232	225	308
Total	756	793	1045
Majority class (%)	69.3	71.2	70.5
Number of features	480	319	576

6.4 Experiments

In this section, the different classification methods will be compared on semi-supervised classification tasks and several datasets. The goal is to classify unlabeled nodes and to compare the results obtained by the different methods in terms of classification accuracy.

This section is organized as follows. First, the datasets used for semi-supervised classification are described in Section 6.4.1. Then, the compared methods are recalled in Section 6.4.2. The experimental methodology is explained in Section 6.4.3. Finally, results are presented and discussed in Section 6.4.4.

6.4.1 Datasets

All datasets are described by (i) an adjacency matrix $\mathbf{A}$ of the underlying graph, (ii) class vectors $\mathbf{y}^c$ (to predict), and (iii) a number of features on nodes gathered in the data matrix $\mathbf{X}_{\text{features}}$. A chi-square test was used to keep only the 100

TABLE 6.3: Class distribution of the **Citeseer, Cora** and **Wikipedia** datasets.

Class	**Citeseer** (DB8)	**Cora** (DB9)	**Wikipedia** (DB10)
Class 1	269	285	248
Class 2	455	406	509
Class 3	300	726	194
Class 4	75	379	99
Class 5	78	214	152
Class 6	188	131	409
Class 7		344	181
Class 8			128
Class 9			364
Class 10			351
Class 11			194
Class 12			81
Class 13			233
Class 14			111
Total	1392	2708	3271
Majority class (%)	32.7	26.8	15.6
Number of features	3703	1434	4973

most significant features for each dataset. The datasets are available at `http://www.isys.ucl.ac.be/staff/lebichot/research.htm` and `https://b-lebichot.github.io/`.

For each of these dataset, if more than one connected component is present, we only use the largest connected component, deleting all the others nodes, features and target classes. The reason is that some of the considered classifiers require a connected graph to work. Also, we choose to work with undirected graphs for all datasets: if a graph is directed, we used $\mathbf{A} = (\mathbf{A}^{\mathrm{T}} + \mathbf{A})/2$ to introduce reciprocal edges.

- ▶ The four **WebKB** datasets (**DB1-DB4**) [196] consist of web pages gathered from computer science departments from four universities (there are four datasets, one for each university), with each page manually labeled into one of four categories: course, faculty, student and project [165]. The pages are linked by citations (if x links to y then it means that y is cited by x, not to be confused with the four co-citation datasets). Each web page in the dataset is also characterized by a binary word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1703 unique words (words appearing less than 10 times were ignored). Originally, a fifth category, Staff, was present but since it contained only very few instances, it was merged with the Project class. Details on these datasets are shown in Table 6.1.

- The three **Ego Facebook** datasets (**DB5-DB7**) [170] consist of "circles" (or friends communities) from Facebook. Facebook data were collected from survey participants using a Facebook application. The original dataset includes node features (profiles), circles, and ego networks for 10 networks. Those data are anonymized and the exact signification of the circles is unknown [170]. We use only the three first networks and the classification task is to predict the affiliation to a circle. Details on these datasets are shown in Table 6.2. Each dataset has two classes.

- The **CiteSeer** dataset (**DB8**) [196] consists of 3312 scientific publications classified into six classes. The pages are linked by citation. Each publication in the dataset is described by a binary word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3703 unique words (words appearing less than 10 times were ignored). The target variable contains the topic of the publications (six topics). Details on this dataset are shown in Table 6.3.

- The **Cora** dataset (**DB9**) [196] consists of 2708 scientific publications classified into one of seven classes denoting topics as for previous dataset. Pages are linked by citations. Each publication is also described by a binary word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1434 unique words or features (words appearing less than 10 times were ignored). The target variable represents the topic of the publications. Details on this dataset are shown in Table 6.3.

- The **Wikipedia** dataset (**DB10**) [196] consists of 3271 Wikipedia articles that appeared in the featured list in the period Oct. 7-21, 2009. Each document belongs to one of 14 distinct broad categories, which were obtained by using the category under which each article is listed. After stemming and stop-word removal, the content of each document is represented by a tf/idf-weighted feature vector, for a total of 4973 words. Pages are linked by citation. The target variable represents the articles field (14 different topics). Details on this dataset are shown in Table 6.3.

Moreover, in order to study the impact of the relative information provided by the **graph structure** and the **features on nodes**, we created new derived datasets by **weakening gradually** the information provided by the node features. More precisely, for each dataset, the features available on the nodes have been ranked by decreasing association (using a chi-square statistics) with the target classes to be predicted. Then, datasets with **subsets** of the features containing respectively the 5 (**5F**), 10 (**10F**), 25 (**25F**), 50 (**50F**), and 100 (**100F**) most informative features were created (5 sets of features). These datasets are weakened versions of the original datasets, allowing to investigate the respective impact of features on nodes and graph structure. We also investigated sets with more features (200 and 400), but conclusions were the same, so that they are not reported here for conciseness.

6.4.2 Compared classification models

In this work, a transductive scheme is used, as we need to know the whole graph structure to label unlabeled nodes. The 16 different algorithms described before will be compared and can be sorted in three categories, according to the information they use. Some algorithms use only features to build the model (denoted as **X – data matrix with features only**), others use only the graph structure (denoted as **A – adjacency matrix of the graph only**), and the third category uses both the structure of the graph and the features of the nodes (denoted as **AX – combined information**).

Using features on nodes only

This reduces to a standard classification problem and we use a linear Support Vector Machine (SVM) based on the features of the nodes to label these nodes (**SVM-X**). Here, we consider SVMs in the binary classification setting (i.e. $y_i \in \{-1, +1\}$). For multiclass problems, we used a one-vs-rest strategy [125]. This classifier is considered as a baseline. In practical terms, we use the well-known Liblinear library [85]. Notice that SVM follows an inductive scheme, unlike all other methods. Transductive SVMs [100] were also considered, but their available Matlab implementation was too slow to be included in the present analysis.

Using graph structure only

Three different families of methods using graph structure only are investigated.

For the **bag-of-paths classifier** based on the bag-of-paths group betweenness (**BoP-A**), the betweenness is computed for each class in turn. Then, each unlabeled node is assigned to the class showing the largest value (see Section 6.3.3 for more details).

Then, for the **sum-of-similarities method** based on the **regularized commute time kernel** (**CTK-A**), the classification procedure is the same as BoP-A: the class similarity is computed for each class in turn and each unlabeled node is assigned to the class showing the largest similarity (see Section 6.3.3).

The four **graph embedding techniques** discussed in Section 6.3.1, used together with a SVM without considering any node feature, are also considered. The SVM is trained using a given number of extracted dominant eigenvectors derived from each measure (this number is a parameter to tune). The SVM model is then used to classify the unlabeled nodes. SVMs using Moran's I, Geary's c, local principal component analysis and the bag-of-paths modularity (see Section 6.3.1) are denoted as **SVM-M-A**, **SVM-G-A**, **SVM-L-A**, and **SVM-BoPM-A**, respectively.

Using both information (features on nodes and graph structure)

Here, we investigate the following models.

In the **double kernel SVM** (**DK-SVM-AX**), two kernels are computed, one defined on the graph and the second from the node features $\mathbf{X}_{\text{new}} = [\mathbf{K}_{\text{A}}, \mathbf{K}_{\text{X}}]$ (see Section 6.3.2). A SVM is then used to classify the unlabeled nodes.

Similarly, the **support vector machine using autocovariates** (**ASVM-AX**), autocovariates are added to the node features $\mathbf{X}_{\text{new}} = [\mathbf{X}_{\text{features}}, \mathbf{Ac}]$ (see Section 6.3.2).

On the other hand, the **spatial autoregressive model** (**SAR-AX**) is a spatial extension of the standard regression model (see Section 6.3.2), used to classify the unlabeled nodes.

Moreover, the dominant eigenvectors (this number is a parameter to tune) provided by the four **graph embedding techniques** (Section 6.3.1) are combined with the node features and then injected into a linear SVM classifier. The new set of feature is therefore $\mathbf{X}_{\text{new}} = [\mathbf{X}_{\text{features}}, \mathbf{X}_{\text{embedding}}]$, where $\mathbf{X}_{\text{embedding}}$ can be obtained using Moran's I, Geary's c, local principal component analysis and the bag-of-paths modularity (see Section 6.3.1). Those four variants are named **SVM-M-AX**, **SVM-G-AX**, **SVM-L-AX**, and **SVM-BoPM-AX**, respectively.

Furthermore, **co-training based on two SVMs** (**SVM-COT-AX**) uses two SVM classifiers, one based on a kernel computed from the features $\mathbf{X}$ and the second based on a graph kernel computed from $\mathbf{A}$ (see Section 6.3.4). A two-steps procedure is then used to classify the unlabeled nodes.

Finally, the **SVM based on kernel canonical correlation analysis** (**SVM-KCA-AX**) first aligns two kernels, one based on $\mathbf{X}$ and one based on $\mathbf{A}$. Then the two aligned kernels are used together as for DK-SVM-AX.

The considered classifiers, together with their parameters to be tuned, are listed in Table 6.4.

6.4.3 Experimental methodology

The classification accuracy will be reported for a 20% labeling rate, i.e. proportion of nodes for which labels are known. Labels of remaining nodes are

TABLE 6.4: The 16 classifiers, the value range tested for tuning their parameters and the most frequently selected values: Mode is the most selected value across all datasets. Note that p, the number of extracted eigenvector, is given in %: this is the relative number of kept features with respect to the number of node of the graph (different for each dataset).

Classification model	Use A	Use X	Acronym	Param.	Tested values	Mode
Bag-of-paths betweenness (6.3.3)	yes	no	BoP-A	$\theta > 0$	$10^{[-9,-6,-3,0]}$	$10^{-6}(40.2\%)$
Sum of similarities with the RCT kernel (6.3.3)	yes	no	CTK-A	$\lambda > 0$	$0.2, 0.4, 0.6, 0.8, 1$	$0.8(39.4\%)$
SVM on Moran's extracted features only (6.3.1)	yes	no	SVM-M-A	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	$10^{-2}(63.0\%)$
				$p > 0$	$[5, 10, 20, 35, 50\%]$	$5\%(74.0\%)$
SVM on Geary's extracted features only (6.3.1)	yes	no	SVM-G-A	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	$10^{-2}(34.8\%)$
				$p > 0$	$[5, 10, 20, 35, 50\%]$	$5\%(39.6\%)$
SVM on LPCA's extracted features only (6.3.1)	yes	no	SVM-L-A	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	$10^{2}(47.3\%)$
				$p > 0$	$[5, 10, 20, 35, 50\%]$	$5\%(69.5\%)$
SVM on BoP modularity extracted features (6.3.1)	yes	no	SVM-BoPM-A	$\theta > 0$	$10^{[-9,-6,-3,0]}$	$10^{0}(35.2\%)$
				$C > 0$	$10^{[-6,-3,0,3,6]}$	$10^{3}(44.4\%)$
				$p > 0$	$[5, 10, 20, 35, 50\%]$	$5\%(72.0\%)$
SVM on node features only (**baseline**)	no	yes	SVM-X	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	$10^{-2}(27.2\%)$
Spatial autoregressive model (6.3.2)	yes	yes	SAR-AX	*none*	–	-
SVM on Moran and nodes features (6.3.1)	yes	yes	SVM-M-AX	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	$10^{2}(26.9\%)$
				$p > 0$	$[5, 10, 20, 35, 50\%]$	$5\%(33.3\%)$
SVM on Geary and nodes features (6.3.1)	yes	yes	SVM-G-AX	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	$10^{2}(21.2\%)$
				$p > 0$	$[5, 10, 20, 35, 50\%]$	$5\%(31.8\%)$
SVM on LPCA and nodes features (6.3.1)	yes	yes	SVM-L-AX	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	$10^{2}(28.4\%)$
				$p > 0$	$[5, 10, 20, 35, 50\%]$	$5\%(41.8\%)$
SVM on BoP modularity and nodes features (6.3.1)	yes	yes	SVM-BoPM-AX	$\theta > 0$	$10^{[-9,-6,-3,0]}$	$10^{0}(27.4\%)$
				$C > 0$	$10^{[-6,-3,0,3,6]}$	$10^{3}(41.0\%)$
				$p > 0$	$[5, 10, 20, 35, 50\%]$	$5\%(48.9\%)$
SVM on autocovatiates and nodes features (6.3.2)	yes	yes	ASVM-AX	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	$10^{0}(28.1\%)$
SVM on a double kernel (6.3.2)	yes	yes	SVM-DK-AX	$C > 0$	$10^{[-6,-4,-2,0,2,4,6]}$	$10^{-4}(31.7\%)$
Co-training based on two SVMs (6.3.4)	yes	yes	SVM-COT-AX	$C > 0$	$10^{[-6,-3,0,3,6]}$	$10^{-6}(34.8\%)$
SVM based on kernel canonical correlation (6.3.4)	yes	yes	SVM-KCA-AX	$C > 0$	$10^{[-6,-3,0,3,6]}$	$10^{-6}(44.2\%)$

deleted during model fitting phase and are used as test data during the assessment phase, where the various classification models predict the most suitable category of each unlabeled node in the test set.

A standard 5-fold nested cross-validation is used for assessing the investigated methods. For each dataset and for each considered feature set, samples (nodes) are randomly assigned into 5 external folds, which defines one **run** of the experimental comparison. Moreover, for each external fold, a 5-fold internal, nested, cross-validation is performed to tune the parameters of the models (see Table 6.4). The results for one specific run are then computed by taking the average over the 5 external folds. The whole procedure is repeated 5 times to mitigate the effect of lucky/unlucky samples-to-fold assignation, so that 5 runs of the experimental comparison for each dataset and feature set are performed, with different fold assignments.

6.4.4 Results and discussion

First of all, most frequently selected parameter values are reported on Table 6.4. We observe that the most selected value for p (the number of eigenvectors

TABLE 6.5: Time analysis of the 16 classifiers. **Time 251** is the time in seconds required to label DB4, with two classes and 251 nodes (Student vs other, cfr Table 6.1). **Time 756** is the time in seconds required to label DB5, with two classes and 756 nodes. **Ratio** is computed as **Time 756** divided by **Time 251** (from DB4 to DB5, the number of node is multiplied by 3.012). Computation times are averaged on 10 runs. Param is a reminder about the parameters to be tuned. The fastest methods are indicated in bold.

Acronym	**Param**	**Implementation**	**Time 251**	**Time 756**	**Ratio**
BoP-A	θ	Matlab (not sparse)	0.69s	25.99s	37.47
CTK-A	α	Matlab (sparse)	0.15s	0.59s	**4.06**
SVM-M-A	C,p	Matlab with MEX(C)	2.03s	28.44s	14.01
SVM-G-A	C,p	Matlab with MEX(C)	0.44s	4.66s	10.57
SVM-L-A	C,p	Matlab with MEX(C)	0.53s	6.13s	11.55
SVM-BoPM-A	θ,C,p	Matlab with MEX(C)	2.16s	58.46s	27.03
SVM-X	none	Matlab with MEX(C)	0.10s	0.22s	**2.08**
SAR-AX	none	Lesage's Matlab toolbox	2.96s	43.80s	14.79
SVM-M-AX	C,p	Matlab with MEX(C)	2.00s	29.76s	14.89
SVM-G-AX	C,p	Matlab with MEX(C)	0.47s	5.19s	11.00
SVM-L-AX	C,p	Matlab with MEX(C)	0.53s	6.88s	13.00
SVM-BoPM-AX	θ,C,p	Matlab with MEX(C)	2.18s	60.41s	27.73
ASVM-AX	C	Matlab with MEX(C)	1.50s	7.78s	**5.18**
SVM-DK-AX	C	Matlab with MEX(C)	1.26s	17.00s	13.48
SVM-COT-AX	C	Matlab with MEX(C)	2.43s	34.92s	14.34
SVM-KCA-AX	C	Matlab	4.37s	82.76s	18.95

extracted for representing the graph structure; see Section 6.3.1) is actually low. This is good news since efficient eigensystem solvers can be used to compute sequentially the first eigenvectors corresponding to the largest (or smallest) eigenvalues.

The classification accuracy and standard deviation, averaged on the 5 runs, are reported on Tables 6.6 (for the methods based on both features and the graph structure) and 6.7 (for the methods based on the graph structure only), for the 10 different datasets and the 5 sets of features. Bold values indicate the best performance on each row. Recall that the BoP-A, CTK-A, SVM-M-A, SVM-G-A, and SVM-L-A methods do not depend on the node features as they are based on the graph structure only.

Moreover, the different classifiers are compared across datasets through a Friedman test and a Nemenyi post-hoc test [74]. The Friedman/Nemenyi test is a non-parametric equivalent of the repeated-measures ANOVA (see Appendix A. The post-hoc Nemenyi test determines whether or not each method is significantly better than another.

This is reported, for each feature set in turn (5F, 10F, . . . , 100F), and thus increasing information available on the nodes, in Figures 6.2 to 6.6, while the result of an overall test based on all the features sets and datasets is shown in

TABLE 6.6: Classification accuracy in percent ± standard deviation, obtained on the 5 runs, the **AX** combined methods (as well as the baseline) and the 10 datasets. Results are reported for the five different feature sets (100F stands for the set of 100 features, and so on). The standard deviation is computed on the 5 folds of the external cross-validation and the 5 independent runs. Best results for each dataset and feature set are highlighted in bold.

		SAR	SVM-G	SVM-M	ASVM	SVM-DK	SVM-BoPM	SVM-L	SVM-COT	SVM-KCA	SVM
		AX	AX	AX	AX	AX	AX	AX	AX	AX	X
	100F	53.2±7.5	84.0±1.4	83.7±1.4	79.4±0.8	**84.9±1.1**	84.1±1.0	83.7±1.3	66.7±1.5	84.1±0.7	83.7±1.5
	50F	66.9±0.9	79.5±1.8	79.5±2.0	79.2±1.9	**80.9±1.6**	79.0±3.2	79.5±2.0	64.4±2.0	78.3±0.9	80.6±0.7
DB1	25F	65.4±3.2	69.8±4.0	68.1±4.4	**74.6±1.3**	73.6±1.5	73.9±2.0	68.6±4.4	57.5±2.0	69.3±1.6	74.6±1.6
	10F	64.4±3.3	63.0±3.8	62.2±4.8	**71.4±3.5**	69.7±3.2	69.8±2.6	62.7±3.9	56.5±1.5	70.3±3.1	70.9±2.1
	5F	62.0±4.2	57.3±5.4	58.1±3.3	**65.8±4.1**	65.2±1.7	60.4±0.9	57.9±2.6	51.7±3.1	65.0±1.2	65.5±0.8
	100F	62.9±1.6	**81.0±0.6**	80.6±0.4	75.1±1.5	79.9±1.4	80.4±1.1	80.4±1.0	60.1±1.2	78.8±1.4	80.8±0.4
	50F	66.5±4.2	76.6±1.4	76.4±2.0	74.4±2.2	76.8±0.9	76.6±1.5	76.7±1.4	58.6±2.5	75.9±0.8	**76.9±0.8**
DB2	25F	66.9±3.6	70.6±2.3	71.0±2.3	73.3±1.1	74.2±2.0	73.4±1.9	71.0±2.5	57.1±1.8	73.0±1.2	**75.5±1.5**
	10F	64.8±5.7	58.9±10.6	57.3±8.2	72.3±2.3	72.4±1.3	72.1±2.3	58.6±9.2	56.6±2.3	72.9±1.5	**74.3±2.1**
	5F	55.7±8.6	56.8±8.1	56.4±7.9	**70.9±1.5**	68.1±2.7	65.7±1.7	57.3±6.8	52.6±1.4	67.6±2.2	66.2±2.0
	100F	64.2±4.4	80.8±1.3	80.7±1.5	80.1±0.7	**81.3±0.7**	81.2±0.9	80.8±1.7	59.0±1.0	80.9±0.6	80.9±1.7
	50F	65.9±3.2	77.2±0.7	77.3±0.8	77.7±1.6	77.7±2.0	78.3±1.1	77.3±0.7	55.5±2.3	77.7±1.0	**78.3±1.3**
DB3	25F	69.1±2.1	73.6±3.2	73.1±2.9	75.7±0.9	76.9±1.2	**78.2±0.7**	73.5±3.2	55.1±1.6	76.8±0.8	77.5±1.0
	10F	63.6±4.4	64.0±8.0	63.7±6.9	**75.7±1.3**	74.9±0.7	74.9±0.8	64.6±7.9	53.8±2.2	74.8±2.1	75.6±1.9
	5F	60.5±6.4	64.3±8.2	65.3±7.4	69.4±1.9	**71.1±0.6**	68.5±3.3	64.1±8.5	51.4±0.4	68.9±0.9	71.0±1.6
	100F	70.5±3.8	83.2±0.7	83.2±0.6	81.1±1.8	**84.3±0.9**	83.4±0.6	83.2±0.9	59.7±1.7	83.3±1.5	83.1±0.7
	50F	72.0±3.8	78.5±1.6	78.5±1.7	79.4±1.0	80.0±2.0	**81.6±1.6**	78.8±1.3	60.7±1.7	79.5±1.2	81.1±2.3
DB4	25F	68.1±3.4	74.3±4.2	74.4±4.1	**79.8±1.9**	78.6±0.9	79.2±0.6	74.1±4.7	60.7±2.1	78.6±1.1	79.4±1.2
	10F	67.7±4.8	64.8±7.7	64.2±7.5	**76.2±1.8**	72.2±0.8	70.2±1.8	65.7±7.0	57.7±0.9	71.4±1.2	71.7±0.7
	5F	61.6±8.8	59.6±7.4	59.4±7.8	**75.5±1.7**	74.4±1.7	73.7±1.7	58.8±7.7	58.2±1.2	73.8±1.1	75.0±0.6
	100F	50.7±12.3	87.8±0.8	87.7±0.7	**92.1±0.4**	88.8±1.0	88.1±0.5	87.9±0.8	91.2±1.2	88.4±0.8	88.5±0.5
	50F	66.2±17.2	87.6±2.2	89.9±1.9	**92.3±0.3**	88.9±0.5	88.9±0.6	88.8±2.1	91.5±1.2	88.8±0.3	89.1±0.7
DB5	25F	80.3±16.6	90.4±1.4	93.7±1.6	**95.3±0.2**	89.1±1.4	91.1±0.9	90.8±1.5	92.1±0.7	89.6±0.6	89.5±0.5
	10F	76.1±9.8	93.1±1.2	94.9±0.7	**95.5±0.4**	89.4±1.2	92.9±0.9	93.7±1.3	91.9±0.7	89.4±1.3	89.5±0.5
	5F	74.7±8.3	93.2±1.1	94.1±1.4	**95.6±1.8**	87.2±0.1	90.1±4.4	94.2±0.8	89.9±1.0	87.3±0.0	87.2±0.4
	100F	72.7±8.1	91.6± 0.4	**93.2±1.1**	92.6±0.4	92.1±0.4	92.5±0.5	91.8±0.9	93.0±0.5	92.3±0.3	91.4±0.2
	50F	78.9±14.4	89.6±2.1	**95.0±1.0**	94.8±1.1	91.5±0.3	92.9±0.8	93.1±0.8	92.2±0.5	91.4±0.5	91.2±0.3
DB6	25F	86.3±8.8	91.7± 1.2	**96.6±1.0**	96.0±1.0	92.0±0.2	94.7±1.0	95.2±2.2	92.5±0.4	91.8±0.4	91.8±0.5
	10F	78.3±9.9	93.5± 0.7	**97.6±0.4**	96.6±1.2	89.7±3.1	95.6±1.4	96.6±0.6	94.0±0.5	90.9±2.5	92.0±0.1
	5F	78.1±10.0	93.3±0.8	**97.6±0.4**	96.6±1.3	88.5±7.6	95.0±1.4	96.5±1.1	92.3±0.5	88.7±7.5	92.2±0.1
	100F	56.4±11.2	74.9±0.8	74.9±1.2	**79.8±0.8**	76.1±0.7	74.1±1.2	75.3±1.1	79.8±1.0	75.4±0.7	74.7±1.1
	50F	61.5±11.3	77.5±1.5	76.0±1.7	**80.5±0.9**	79.6±0.6	77.7±1.0	78.4±1.2	81.1±0.5	78.9±1.1	78.0±1.1
DB7	25F	69.7±8.0	79.3±0.5	79.9±0.9	**81.6±0.4**	80.3±0.4	79.1±0.4	80.0±0.6	81.3±0.4	80.8±0.3	78.9±2.4
	10F	66.0±10.6	78.7±2.1	80.7±1.2	**81.2±0.1**	80.2±0.8	79.3±1.3	**81.2±0.9**	80.2±0.9	80.6±0.3	80.5±0.3
	5F	66.2±10.3	79.9±0.8	79.8±1.1	**80.9±0.6**	80.5±0.4	77.1±2.9	**80.9±0.8**	77.9±0.4	80.4±0.2	78.5±3.9
	100F	61.2±4.9	70.5±0.6	70.5±0.6	66.1±0.6	70.4±0.3	**70.6±0.4**	70.5±0.6	68.1±0.3	70.4±0.3	70.5±0.6
	50F	64.5±1.6	62.9±4.2	64.1±3.1	66.1±0.7	68.5±0.3	68.7±0.5	62.8±4.3	68.6±0.8	68.6±0.2	**68.8±0.5**
DB8	25F	56.0±7.9	59.9±3.1	65.4±0.8	66.0±1.2	**70.2±0.3**	66.0±0.4	62.3±2.0	70.0±0.8	68.1±0.6	66.7±0.4
	10F	44.2±11.2	57.1±3.1	66.3±1.9	62.9±0.8	**72.7±0.4**	63.6±1.0	61.4±2.7	65.6±1.5	65.7±1.3	59.4±0.4
	5F	42.7±12.0	57.1±2.5	67.5±0.6	61.9±0.9	**72.0±1.4**	63.0±2.1	61.9±1.2	65.8±4.4	68.0±0.9	53.9±0.6
	100F	**77.5±1.0**	71.4±0.4	71.3±0.4	70.6±0.9	71.3±0.7	71.7±0.7	71.2±0.5	75.7±0.6	67.5±0.9	71.4±0.4
	50F	64.3±5.7	66.0±2.5	72.1±1.2	**76.3±0.3**	69.4±0.2	71.9±1.5	73.0±1.8	76.1±0.6	68.4±0.5	68.6±0.1
DB9	25F	53.6±10.1	67.5±3.9	73.8±2.5	**77.0±0.3**	74.0±0.3	76.2±0.3	73.9±2.8	74.5±0.4	70.1±0.1	64.2±0.1
	10F	42.9±9.8	72.1±3.1	76.4±1.8	74.9±0.5	76.6±0.3	**77.3±0.9**	76.8±1.9	69.7±0.6	67.1±0.1	56.3±0.2
	5F	37.1±6.8	73.2±2.5	76.1±1.3	71.8±0.9	78.0±0.3	**80.3±1.2**	76.3±1.6	65.6±1.1	67.0±0.3	42.3±1.0
	100F	32.1±5.2	54.6±0.5	54.4±0.5	44.5±1.2	54.2±0.6	**56.2±0.5**	54.9±0.2	49.6±0.2	52.1±0.3	54.6±0.5
	50F	35.6±7.7	45.9±1.7	46.7±0.9	37.1±0.4	41.9±0.3	45.0±0.6	**48.6±0.7**	40.4±0.5	40.8±0.5	40.2±0.4
DB10	25F	35.4±6.1	43.4±2.0	45.5±2.4	32.7±0.9	36.7±0.6	41.6±1.4	**46.2±2.4**	34.7±0.5	35.5±0.2	34.2±0.2
	10F	29.0±2.5	39.2±2.1	41.9±1.9	28.2±0.9	31.8±0.3	42.2±0.4	**42.3±2.0**	32.9±0.4	31.1±0.1	30.8±0.1
	5F	21.2±1.8	35.9±2.0	38.7±2.6	25.1±0.8	25.8±0.2	**42.0±0.8**	39.6±2.5	31.6±0.9	25.8±0.2	25.2±0.3

TABLE 6.7: Classification accuracy in percent ± standard deviation, obtained on the 5 runs, the 6 **A** methods (and baseline) and the 10 datasets. Baseline results are reported for 100F feature sets. The standard deviation is computed on the 5 folds of the external cross-validation and the 5 independent runs. Best results for each dataset and feature set are highlighted in bold.

	BoP	CTK	SVM-M	SVM-G	SVM-L	SVM-BoPM	SVM (100F)
	A	A	A	A	A	A	X
DB1	54.5±1.7	54.2±0.9	46.3±5.1	43.4±2.7	40.8±2.3	43.5±0.8	**83.7±1.5**
DB2	41.8±4.2	42.4±1.1	33.1±1.7	33.3±2.2	33.1±3.5	32.3±3.4	**80.8±0.4**
DB3	48.4±0.6	46.7±1.3	47.0±4.9	44.1±2.8	40.4±1.5	39.3±2.3	**80.9±1.7**
DB4	45.7±1.5	42.9±3.2	40.0±1.5	40.0±2.3	42.0±1.6	42.6±3.3	**83.1±0.7**
DB5	**96.8±0.1**	96.3±0.1	95.4±0.5	89.8±4.1	90.9±0.8	91.7±0.6	88.5±0.5
DB6	**98.6±0.1**	98.4±0.1	95.1±0.3	93.8±0.4	95.9±1.0	94.4±1.1	91.4±0.2
DB7	82.5±0.2	**82.9±0.5**	81.6±0.8	78.5±0.5	79.1±1.4	79.1±0.9	74.7±1.1
DB8	69.9±0.6	**70.5±0.4**	55.9±0.4	68.1±0.3	62.4±0.9	67.5±1.2	**70.5±0.6**
DB9	78.1±0.2	**81.7±0.2**	74.5±0.6	75.6±0.7	76.6±0.4	80.3±0.3	71.4±0.4
DB10	35.3±0.3	36.4±0.2	30.8±0.6	35.0±0.2	34.4±0.7	14.9±0.3	**54.6±0.5**

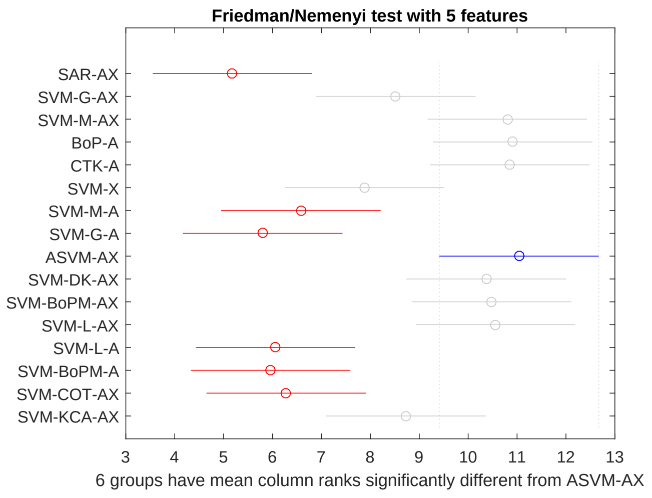


FIGURE 6.2: Mean rank (circles) and critical difference (plain line) of the Friedman/Nemenyi test, over 5 runs and all datasets, obtained on partially labeled graphs. The blue method has the best mean rank and is statistically better than red methods. Labeling rate is 20% and the critical difference is 3.26. This figure shows the results when only 5 node features are considered (5F datasets).

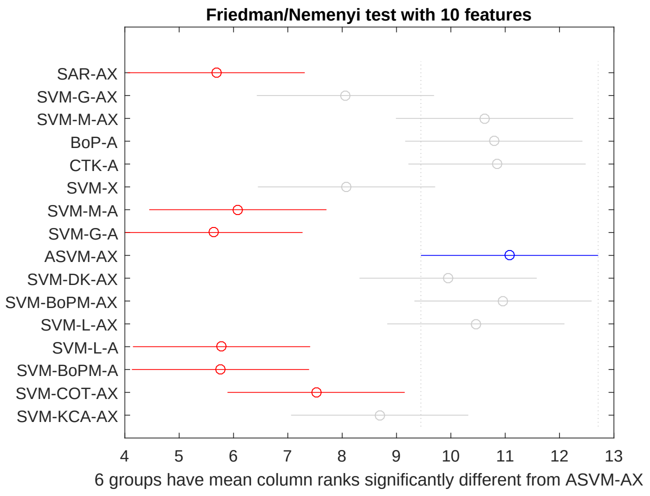


FIGURE 6.3: Friedman/Nemenyi test considering 10 node features (10F datasets); see Figure 6.2 for details. The critical difference is 3.26.

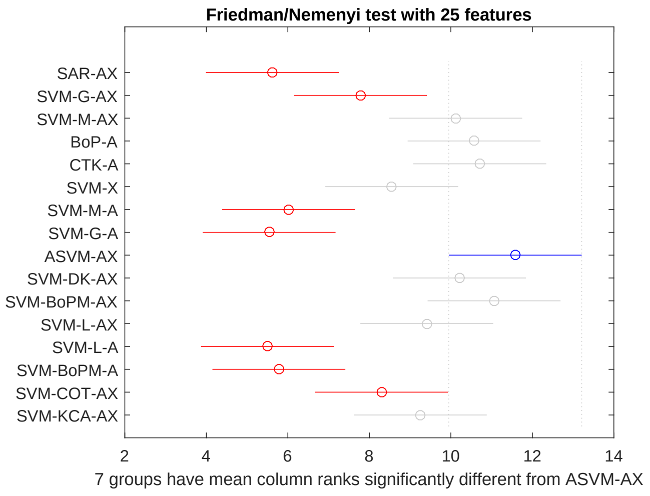


FIGURE 6.4: Friedman/Nemenyi test considering 25 node features (25F datasets); see Figure 6.2 for details. The critical difference is 3.26.

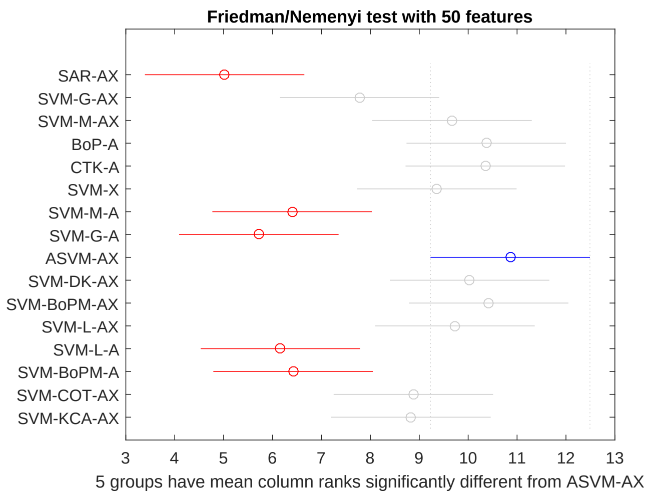


FIGURE 6.5: Friedman/Nemenyi test considering 50 node features (50F datasets); see Figure 6.2 for details. The critical difference is 3.26.

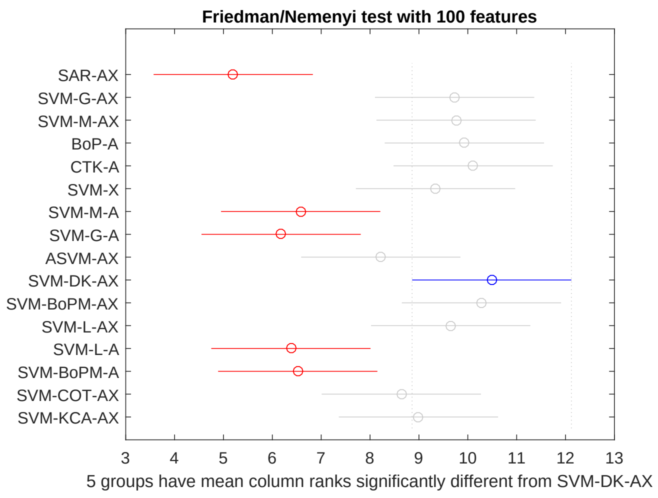


FIGURE 6.6: Friedman/Nemenyi test considering 100 node features (100F datasets); see Figure 6.2 for details. The critical difference is 3.26.

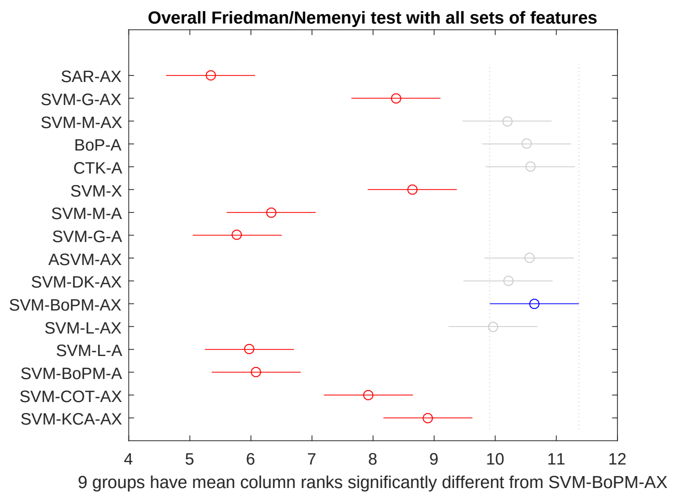


FIGURE 6.7: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F , 50F, 100F). The critical difference is 1.46; see Figure 6.2 for details.

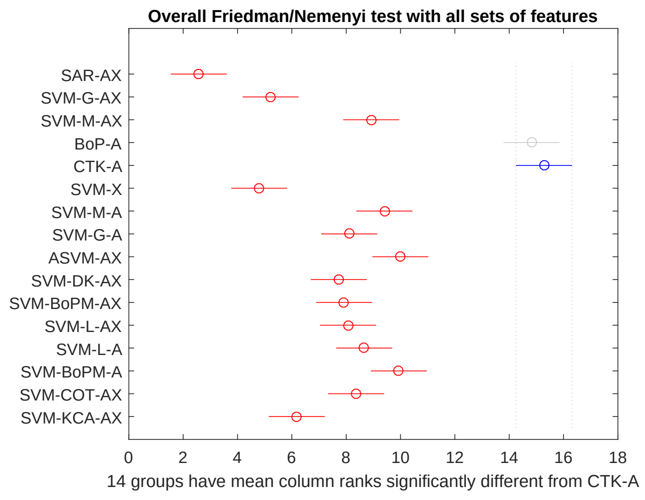


FIGURE 6.8: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F, 50F, 100F), but computed only on datasets DB5 to DB9 (driven by graph structure, A); see Figure 6.2 for details. The critical difference is 2.06.

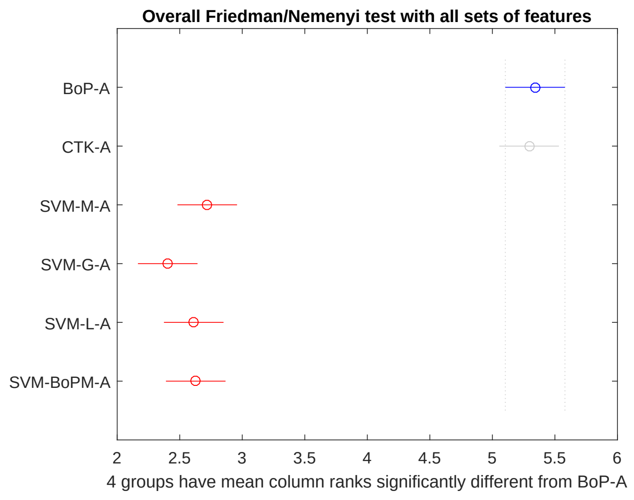


FIGURE 6.9: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F, 50F, 100F), for methods based on graph information alone (A); see Figure 6.2 for details. The critical difference is 0.48.

Figure 6.7.

Overall performances on all datasets and all node feature sets

From Tables 6.6 to 6.7 and Figure 6.7, overall best performances on all dataset and all node features sets are often obtained either by a SVM based on node features combined with new features derived from the graph structure (Section 6.3.1), or, unexpectedly, by the CTK-A sum-of-similarities method (using graph structure only; see Section 6.3.3), which performs quite well on datasets five to nine. The BoP-A node betweenness (using graph structure only, see Section 6.3.3) is also competitive and achieves results similar to the sum-of-similarities CTK-A method (as already observed in [156]).

The best method among the graph structure plus node features SVM is not straightforward to determine (see Figure 6.7). From Figures 6.2 to 6.6, the main trend is that the performance decreases when the number of features decreases, which seems normal.

However, this trend is not always observed; for example, with the SVM-M-AX method (SVM with features extracted from Moran's index and features on nodes, see Section 6.3.1) and dataset DB5, the performances rise when the

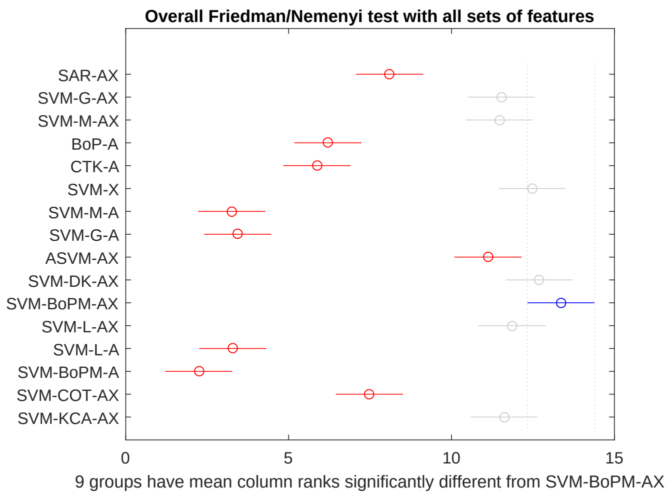


FIGURE 6.10: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F, 50F, 100F), but computed only on datasets DB1 to DB4 and DB10 (driven by node features, X); see Figure 6.2 for details. The critical difference is 2.06.

number of features decreases. This can be explained by observing that each dataset labeling can be better explained in terms of its graph structure (graph-driven datasets, DB5 to DB9), or by its node features (features-driven datasets, DB1 to DB4, plus DB10).

To confirm this fact, the network structure autocorrelation was computed for each class (i.e. for each $\mathbf{y}^c$) and the average is reported for each dataset. This measure quantifies to what extent the target variable is correlated with its neighboring nodes. The values are reported on Table 6.8 for Moran's I, Geary's c and the LPCA contiguity ratio (see Section 6.3.1). For Moran's I, high values (large autocorrelation) indicate that the graph structure is highly informative. This is the opposite for Geary and LPCA, as small values correspond to a large autocorrelation. It can be observed that our hypothesis is clearly confirmed.

Nevertheless, from Tables 6.6 and 6.7 and Figure 6.7, the best overall performing methods combining node features and graph structure are (excluding the methods based on the graph alone, BoP-A and CTK-A), SVM-BoPM-AX (see Section 6.3.1) and ASVM-AX (see Section 6.3.2). While performing better on our datasets (their mean rank is slightly higher), they are however not statistically different from SVM-M-AX, SVM-L-AX, and SVM-DK-AX.

Notice also that, from Figure 6.7, if we look at the performances obtained by

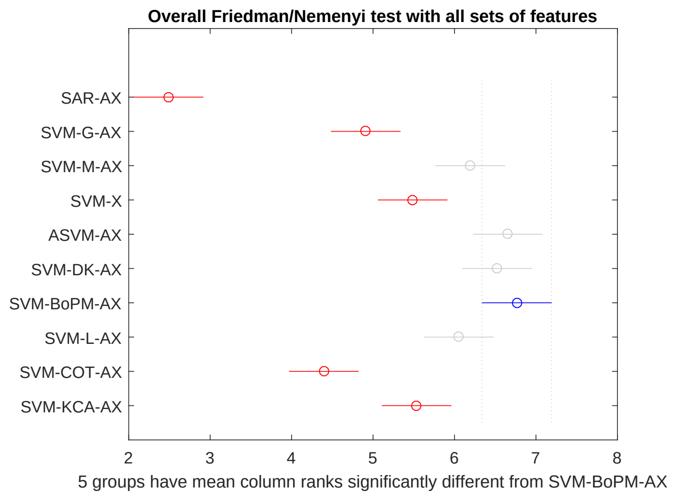


FIGURE 6.11: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F, 50F, 100F), performed only on methods combining graph structure and node features information (AX, plus simple SVM-X as baseline). See Figure 6.2 for details. The critical difference is 0.86.

a baseline linear SVM based on node features only (SVM-X), we clearly observe that integrating the information extracted from the graph structure improves the results. Therefore, it seems to be a good idea to consider collecting graph-based information, which could improve the classification results.

Exploiting either the structure of the graph or the node features alone

Obviously, as already mentioned, datasets DB5 to DB9 are graph-driven, which explains the good performances of the sum-of-similarities CTK-A and BoP-A on these data. For these datasets, the features on the nodes do not help much for predicting the class label, as observed when looking to Figure 6.8 where results are displayed only on these datasets. It also explains the behavior of method SVM-M-AX on dataset DB5, among others.

In this case, the best performing methods are the sum-of-similarities CTK-A and the bag-of-paths betweenness BoP-A (see Section 6.3.3). This is clearly

[2]LPCA contiguity ratio is positive and lower-bounded by $cr_0 = 1 - \sqrt{\lambda_{\max}}$ (which tends to be close to zero) where $\lambda_{\max}$ is the largest eigenvalue of $\mathbf{A}$. The upper bound is unknown [154].

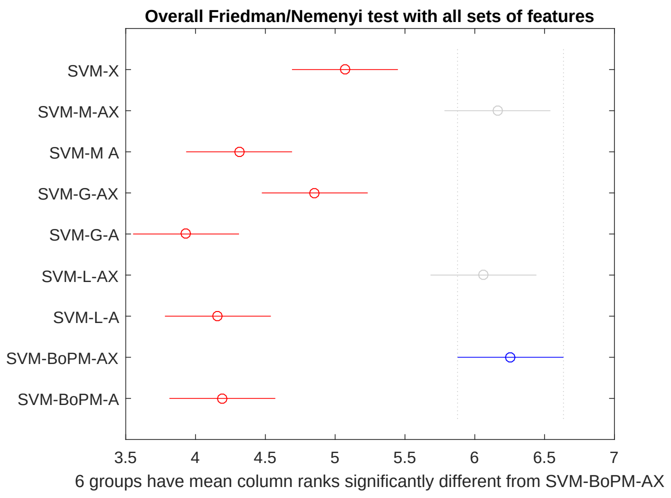


FIGURE 6.12: Friedman/Nemenyi test considering all feature sets (5F, 10F, 25F, 50F, 100F), only considering methods based on a graph embedding (plus regular linear SVM for comparison). See Figure 6.2 for details. The critical difference is 0.76.

confirmed by displaying the results of the methods based on the graph structure only in Figure 6.9 and the results obtained on the graph-driven datasets in Figure 6.8. Interestingly, in this setting, these two methods ignoring the node features (CTK-A and BoP-A) are outperforming the SVM-based methods.

Conversely, on the node features-driven datasets (DB1 to DB4 and DB10; results displayed in Figure 6.10 and Tables 6.6 and 6.7), all SVM methods based on node features (and graph structure) perform well while methods based on the graph structure only obtain much worse results, as expected. In this setting, the situation is rather similar to the overall results case (see Figure 6.7). The two best techniques are SVM-BoPM-AX (see Section 6.3.1) and SVM-DK-AX (see Section 6.3.2). However, this time, these two first ranked methods are not significantly better than the simple linear SVM based on features only (SVM-X), as shown in Figure 6.10. Thus, for the node features-driven datasets, the graph structure does not bring much additional information.

From another point of view, Figure 6.11 takes into account all datasets and compares only the methods combining node features and graph structure. In this setting, the best ranked methods are again SVM-BoPM-AX, ASVM-AX and SVM-DK-AX which are now significantly better than the baseline SVM-X

TABLE 6.8: Mean autocorrelation of class membership computed on all the investigated datasets. For Moran's I, a high value corresponds to a large autocorrelation. Conversely, for Geary's c and LPCA, a small value implies a large autocorrelation. For each autocorrelation measure, **–** indicates the presence of negative autocorrelation, a value close to **0** indicates a lack of structural association and **+** indicates presence of positive autocorrelation. See Section 6.3.1 and [134] for details. Datasets can be divided into two groups (more driven by graph structure (A) or by features on nodes (X)), according to these measures. I_0 is equal to $-1/(n-1) \approx 0$, where n is the number of nodes.

A-driven	**+**	**0**	**–**	DB 5	DB 6	DB 7	DB 8	DB 9
Moran's I	$> I_0$	$= I_0$	$< I_0$	1.27	1.09	0.66	0.53	0.79
Geary's c	< 1	1	> 1	0.09	0.09	0.33	0.19	0.12
LPCA c. ratio [2]	cr_0	$> cr_0$		0.20	0.13	0.58	0.67	0.26
X-driven	**+**	**0**	**–**	DB 1	DB 2	DB 3	DB 4	DB 10
Moran's I	$> I_0$	$= I_0$	$< I_0$	-0.22	-0.12	-0.15	-0.06	0.15
Geary's c	< 1	1	> 1	0.78	0.59	0.63	0.57	0.43
LPCA c. ratio [2]	cr_0	$> cr_0$		2.54	2.10	1.86	1.90	0.82

(less methods are compared on more datasets).

Notice that SVM-KCA-AX performs better than SVM-COT-AX, but is not significantly different from SVM-DK-AX, although SVM-DK-AX's mean rank is higher than SVM-KCA-AX's mean rank.

Finally, the worst performing method is always SAR-AX, except if only features-driven datasets are considered. Even in this case, SAR-AX outperforms only graph-based methods (see Figure 6.10).

Comparison of graph embedding methods

Concerning the embedding methods described in Section 6.3.1, we can conclude that Geary's index (SVM-G-A and SVM-G-AX) should be avoided by preferring the bag-of-paths modularity (SVM-BoP-AX), Moran's index (SVM-M-AX) or local principal component Analysis (SVM-L-AX). This is clearly observable when displaying only the results of the methods combining node features and graph structure in Figure 6.11. This result is further confirmed when comparing only the methods based on a graph embedding in Figure 6.12.

Discussion about time complexity

Table 6.5 shows a comparison of the computation times of the 16 classifiers. Notice that, from columns **Time 251** to **Time 756**, the number of nodes is multiplied by 3.012. In terms of ratio, the fastest methods are CTK-A (which has an efficient sparse implementation, quasi-linear in number of samples/modes) for graph-based classifiers and ASVM-AX for classifiers combining features and structural information. Interestingly, those two classifiers actually achieved good overall results previously. Notice that BoP-based methods are the slowest due to a full matrix inversion ($O(n^3)$). Most other methods exhibit a nearly quadratic behavior. Finally, notice that most SVM methods use the Liblinear library [85], allowing Matlab to execute C code through a MEX file.

Summary of main findings

To summarize, the experiments lead to the following conclusions: The best performing methods are highly dependent on the dataset. We observed (see Table 6.8) that, quite naturally, some datasets are more graph-driven in the sense that the network structure conveys important information for predicting the class labels, while other datasets are more node features-driven and, in this case, the graph structure does not help much. However, it is probably a good idea to take into consideration information about the graph structure, because this additional information can improve significantly the results, depending on the dataset (see Figures 6.11 and 6.12).

If we consider the graph structure alone, the two best investigated methods are the sum-of-similarities (CTK-A) and the bag-of-paths betweenness (BoP-A, see Section 6.3.3). They clearly outperform the graph embedding methods, but also the SVMs on some datasets. This is confirmed by a paired signed Wilcoxon test: BoP-A and CTK-A outperform SVM-X at $p < 10e^{-5}$.

When, in addition, informative features on nodes are available, it is worth considering combining the information, and, in this context, we found that the best performing methods are SVM-BoPM-AX (SVM with bag-of-paths modularity, see Section 6.3.1), ASVM-AX (SVM based on autocovariates, see Section 6.3.2), and SVM-DK-AX (SVM based on a double kernel, see Section 6.3.2) (see Figure 6.11). Taking the graph structure into account improves the results over a baseline SVM considering node features only. This is confirmed (but only at $p < 0.05$) for the two first methods by a paired signed Wilcoxon test: SVM-BoPM-AX outperforms SVM-X with $p = 0.042$ and ASVM-AX outperforms SVM-X with $p = 0.029$. On the contrary, the p-value for SVM-DK-AX against SVM-X is only 0.182.

6.5 Conclusion

This work considered a data structure consisting in a graph and plain features defined on the nodes of the graph. In this context, 16 semi-supervised classification methods were investigated to compare the feature-based approach, the graph structure-based approach, and the dual approach combining both information sources.

It appears that the best results are often obtained either by a SVM method (the considered baseline classifier) based on plain node features combined with a given number of new features derived from the graph structure (namely from the BoP modularity or autocovariates), or by the sum-of-similarities and the bag-of-paths modularity method, based on the graph structure only, which perform well on some datasets for which the graph structure carries important class information.

Indeed, we observed empirically that, quite naturally, some datasets can be better explained by their graph structure (i.e. graph-driven datasets), or by their node features (i.e. features-driven datasets). Consequently, neither the graph-derived features alone or the plain features alone are sufficient to achieve optimal performances. In other words, in some situations, standard feature-based classification results can be improved significantly by integrating information from the graph structure. In particular, the most effective methods were based on the bag-of-paths modularity (SVM-BoPM-AX), autocovariates (ASVM-AX) or a double kernel (SVM-DK-AX).

The take-away message can be summarize as follows: if the dataset is graph-driven, a simple sum-of-similarities or a bag-of-paths betweenness already perform well, but this is not the case if the features on the nodes are (more) informative. In both cases, SVM-BoPM-AX, ASVM-AX, SVM-DK-AX still ensured good overall performances, as shown on the investigated datasets.

A key point is therefore to determine a priori if a given dataset is graph-driven or features-driven. In this chapter, we proposed to use some well-known spatial autocorrelation indexes to tackle this issue. Further investigations will be carried in that direction. In particular, how can we automatically infer properties of a new dataset (graph-driven or features-driven) if all class labels are not known? For instance, can we rely on measuring autocorrelation based on features?

Finally, the present work does not analyze the scalability of the methods; this is also left for further work.

Chapter 7

A bag-of-paths node criticality measure

The analysis and the modeling of network data has become a popular research topic in the last decade and is now often referred to as **link analysis** (in computer science) and **network science** (in physics). Network data appear in virtually every field of science and is therefore studied in many different disciplines, such as social sciences, applied mathematics, physics, computer science, chemistry, biology, economics, ... Within this context, one important question that is often addressed is the following: Which node seems to be the most critical, or vital, in the network? The present work introduces such a new **node criticality measure**, also called **vulnerability**, quantifying to what extent the deletion of each node hurts the connectivity within the network in a broad sense, e.g., in terms of communication, proximity, or movement. Criticality measures are often considered as a subset of centrality measures, which are frequently used as a proxy for quantifying criticality. Interested readers are invited to consult the recent comprehensive review [162].

Indeed, a large number of centrality measures have been defined in various fields, starting from social science (see, e.g., [96, 181, 83, 251, 146, 236] and [50] for a survey). These quantities assign a score to each node of the graph G which reflects to what extent this node is **central** by exploiting the structure of the graph G, or with respect to the communication flow between nodes. Centrality measures try to answer the following questions [149]: What is the most representative, or central, node within a given graph (closeness centrality)? How critical is a given node with respect to the information flow in a network (criticality)? Which node is the most peripheral in a social network (eccentricity)? Which node is the most important intermediary in the network (betweenness centrality)? Centrality scores try to answer to these questions by proposing measures modeling and quantifying these different, somewhat vague, properties of the nodes.

This chapter introduces a new, efficient and effective, criticality measure: the bag-of-paths (BoP) criticality. The quantity relies on the bag-of-paths framework introduced in Chapter 4: the BoP criticality of a node measures the impact of the node deletion on the total accessibility between nodes within the network. More precisely, we propose to use the Kullback-Leibler divergence between the bag-of-paths probabilities, computed before and after removal of a node of interest, to quantifying relative **accessibilities**. The larger this decrease in accessibility, the higher the impact of the node deletion, and thus the higher its criticality.

The novelty of this approach can be understood as follows. Most of the traditional criticality measures are essentially based on two different paradigms about the communication occurring in the network: optimal communication based on shortest paths and random communication based on a random walk on the graph. For instance, the Wiener index (described later in this chapter) is based on shortest paths and the Kirchhoff index on random walks. However, both the shortest path and the random walk have some drawbacks [90]: shortest paths do not integrate the amount of connectivity between the two nodes whereas random walks quickly loose the notion of proximity to the initial node when the graph becomes larger [244].

Contrary to traditional measures, our criticality measure integrates both proximity and amount of connectivity in the bag-of-paths framework [94]. Nodes that are both close and highly connected are qualified as highly **accessible**. Our introduced bag-of-paths measures aim to quantify the accessibility between the nodes. When the temperature parameter of the model is low (i.e. close to zero, see Chapter 4), communication occurs through a random walk, while for large temperatures, short paths are promoted.

The introduced measure is compared experimentally to already developed criticality measures as well as to a sample of popular centrality measures, briefly reviewed in this chapter. All those measures are compared through a Kendall's correlation analysis and a **disconnection methodology** [9, 123] in Section 7.4. This empirical analysis is performed on a large number, and two types, of randomly generated graphs (see Section 7.4.1).

In summary, this work has the following main contributions,

- ▶ A new criticality measure, showing good performance in the identification of the most critical nodes of a network, is introduced.
- ▶ Many criticality measures introduced in the literature are reviewed and are grouped using a Kendall's correlation and two dendrograms.
- ▶ All those methods are compared experimentally using two disconnection strategies on a large number of randomly generated graphs and on small real-life social networks.

Finally, this chapter is organized as follows: First, the underlying background and various notations are discussed in Section 7.1, then Section 7.2 introduces ten centrality and criticality measures (some being quite well-known). The bag-of-paths model described in [94] is summarized and the new BoP criticality measure is derived in Section 7.3. Finally, those measures are assessed and compared in Section 7.4.

7.1 Background and Notation

This section aims to introduce the necessary background and notation used in this chapter. Most of them has already been presented in Chapter 2.

Let us just recall the Laplacian matrix $\mathbf{L}$ of the graph, introduced in Section 2.4,

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{7.1}$$

where $\mathbf{D} = \mathbf{Diag}(\mathbf{Ae})$ is the diagonal (out)degree matrix of the graph G containing the $a_{i\bullet}$ on its diagonal. One interesting property of $\mathbf{L}$, widely used in this Chapter, is that its eigenvalues provide important information about the connectivity of the graph [63]. Another interesting property is that the number of zero eigenvalues of $\mathbf{L}$ is equal to the number of disconnected subgraphs, or connected components, of G [63]. Then, for a connected graph the smallest eigenvalue of $\mathbf{L}$ is called the **algebraic connectivity** and has been shown to be a good indicator of its overall **connectedness** (G is disconnected when its algebraic connectivity is equal to zero).

Finally, the Moore-Penrose pseudoinverse of $\mathbf{L}$ is denoted as $\mathbf{L}^+$, and contains elements l_{ij}^+. Due to the properties of the Moore-Penrose pseudoinverse, its largest eigenvalue is the algebraic connectivity.

7.2 Related Work

In Section 7.4, a large set of criticality measures (see Table 7.1 for acronyms) are compared experimentally, and briefly reviewed in this section (see [50, 90] for a more thorough description of these measures). It is convenient to categorize them into three classes: node betweenness centrality measures (see Section 7.2.1), node graph criticality measures (see Section 7.2.2), and global criticality measures (see Section 7.2.3).

Notice that, in general, these centrality measures are computed on undirected graphs (see Section 2.2), or, when dealing with a directed graph, by ignoring the direction of edges. They are therefore denoted as **undirectional** [250]. Measures defined on directed graphs are often called **importance** or **prestige** measures. They capture to what extent a node is **important**, **prominent**, or **prestigious** with respect to the whole directed graph by considering directed edges as representing some kind of endorsement. However, this kind of measure is not discussed here.

One of the most interesting global criticality measure of the graph G, the so-called connectivity, is often defined as the minimum number of nodes that need to be removed to separate it into two disconnected sub-graphs [115, 216]. Unfortunately, this quantity is hard to compute (the problem is NP-hard) and cannot be easily exploited in practice for this reason.

7.2.1 Node betweenness centralities

As already mentioned, the concept of criticality is closely related to the concept of betweenness centrality; we therefore also investigate a few of the most well-known betweenness and centrality measures. The measure is defined on each node, identified by its index j.

- ▶ The simple **node degree**, or **edge connection** (EC). This quantity is simply the number of nodes connected to a node j, weighted by edge weights in the case of a weighted graph. It is obtained by summing the entries on the jth row of the adjacency matrix $\mathbf{A}$. The idea is that if a node has a high degree, it is more likely to hurt or disconnect the graph when removed. It can be computed by

$$\mathrm{EC}_j = \mathbf{e}_j^{\mathrm{T}}\mathbf{A}\mathbf{e} \tag{7.2}$$

- ▶ The famous **shortest path betweenness** (SPB), introduced by Freeman [96]. It counts the proportion of shortest paths connecting any two nodes i and k, and passing through an intermediate node j of interest (with $i \neq j \neq k \neq i$). The idea is that if a node contributes to a large number of shortest paths, it can be considered as an important intermediary between nodes when the information is spread optimally along shortest paths. More precisely,

$$\mathrm{SPB}_j = \sum_{\substack{i=1 \\ i\neq j}}^{n} \sum_{\substack{k=1 \\ k\neq i,j}}^{n} \frac{\eta(j \in \mathcal{P}_{ik}^*)}{|\mathcal{P}_{ik}^*|} \tag{7.3}$$

where $\mathcal{P}_{ik}^*$ is the set of all shortest paths from i to k, $|\mathcal{P}_{ik}^*|$ is the total number of such shortest paths $\wp_{ik}^*$ and $\eta(j \in \mathcal{P}_{ik}^*) = \sum_{\wp_{ik}^* \in \mathcal{P}_{ik}^*} \delta(j \in \wp_{ik}^*)$ is the total number of such paths visiting node j. We used Brandes' algorithm [49] to compute the SPB of each node of the graph.

- The **random walk betweenness** (RWB), introduced by Newman [181] and closely related to Brandes' electrical centrality [51]. Newman introduced the current flow betweenness centrality, which measures the centrality of a node as the total sum of electrical currents that flow through it, when considering all node pairs as source-destination pairs with a unit current flow. The current flow betweenness is also called the **random walk betweenness centrality** because of the well-known connection between electric current flows and random walks [78, 90]. The idea is thus the same as for the SPB, but taking into account a random walk-based diffusion of information instead of shortest paths. Notice that Brandes and Fleischer [51] proposed a more efficient algorithm computing the random walk betweenness for all nodes of a network. The properties and computation of the current flow betweenness have also been discussed by Bozzo and Franceschet [47]. Kivimaki et al. proposed a new betweenness measure interpolating between the shortest path betweenness and the random walk betweenness [145].

- **Estrada's centrality** (EST). In [83], Estrada et al. defined a centrality measure called **subgraph centrality** for a weighted undirected graph or subgraph. It summarizes simply as

$$\mathrm{EST}_j = \mathbf{e}_j^{\mathrm{T}} \left(\sum_{k=0}^{\infty} \frac{\mathbf{A}^k}{k!} \right) \mathbf{e}_j = \mathbf{e}_j^{\mathrm{T}}\, \mathbf{diag}(\mathrm{expm}(\mathbf{A})) \qquad (7.4)$$

  where $\mathrm{expm}(\mathbf{A})$ is the matrix exponential of $\mathbf{A}$ and $\mathbf{diag}(\mathbf{X})$ extract the main diagonal of $\mathbf{X}$. It is well-known that element $a_{ij}^{(k)} = [\mathbf{A}^k]_{ij}$ of matrix $\mathbf{A}^k$ ($\mathbf{A}$ to the power k) is the weighted number of paths between node i and node j with exactly k steps (see Chapter 2). The subgraph centrality measure therefore integrates a contribution from all paths connecting node j to himself, discounting paths according to their number of steps (it favors shorter paths in terms of length). The intuition is that a node should have a high centrality score if the closed paths (cycles) starting from it are short and are visiting many different nodes [83].

7.2.2 Node criticalities

We now introduce the node criticalities studied in this work. As for the betweenness, the criticality measure is defined on each node j.

- **Wehmuth's criticality** K (WK) is introduced in [251],

$$\mathrm{WK}_j = \frac{\lambda_2^{(j)}}{\log_2(d_j)} \tag{7.5}$$

  where $\lambda_2^{(j)}$ is the algebraic connectivity of the h-neighbourhood of node j (the subnetwork composed by all nodes within h hops, or steps, of node j) and d_j is the degree of node j. Recall that the algebraic connectivity is the second smallest eigenvalue of the Laplacian matrix $\mathbf{L}$. The idea is to take advantage of the algebraic connectivity property; the higher the value of $\lambda_2^{(j)}$, the higher the connectivity/density of the subnetwork. Then, $\lambda_2^{(j)}$ is divided by the logarithm of the node degree as locally computed algebraic connectivities show a bias towards higher values on nodes with high degree. This bias causes $\lambda_2^{(j)}$ to be over-sensitive to the presence of hubs [251].

- **Klein's edge criticality** (KLE). Klein derived the analytical form of this node criticality measure for several global measures, including the Wiener index and the Kirchhoff index [146]. We used the measure based on the Kirchhoff index here:

$$\mathrm{KLE}_j = \sum_{i=1}^{n} a_{ij}(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}(\mathbf{L}^{+})^2(\mathbf{e}_i - \mathbf{e}_j) \tag{7.6}$$

  The intuition behind the measure is the following. Klein's edge (i, j) criticality is defined as the sensitivity of the global network criticality index (here the Kirchhoff index – defined in the next section) with respect to the increase in the resistance of the edge (i, j) [146]. In other words, it quantifies the impact of an increase in this resistance on the global network. Edges having a high impact on the global network criticality hurt most the network and are considered as highly critical. Then, edge criticality is summed up over incident edges to provide a node criticality.

7.2.3 Global network criticalities

The following **global criticality** indexes are defined on the whole network G. They quantify to what extent the network as a whole is efficient, that is, highly interconnected and cohesive, with high accessibility. For a communication network, this measure can be, e.g., the **Wiener index** – the sum of the shortest path distances (which can be travel time, travel cost, . . .) between all pairs of nodes. An effective network is characterized by a **low value** of the Wiener index as, then, distances between nodes are small in average.

The impact of a node of interest on the global network accessibility measure – the **derived node criticality** – is then quantified by evaluating the marginal loss in global accessibility when the node of interest is not operating (i.e. has simply been removed). This measure therefore reports how critical the node is, relative to the entire graph. To evaluate the criticality of a particular node j in a fixed graph G, the difference between the initial global network criticality, cr(G), and the global criticality after deleting this node j, cr($G \setminus j$), is computed [50],

$$\text{cr}_j(G) = \text{cr}(G) - \text{cr}(G \setminus j) \tag{7.7}$$

and the higher this value, the more critical node j is. Here, $G \setminus j$ is graph G whose node j and incident edges have been removed.

This node criticality is computed on several well-known global criticality measures which are described now. We could also normalize the quantity when it corresponds to a sum over all pairs of nodes by something like $\mathrm{cr}(G \setminus i)/((n-1)(n-2)) - \mathrm{cr}(G)/(n(n-1))$. However, this would not change the ranking of the nodes as the second term is a constant.

- The **Wiener index** (WIE) is defined as the sum of the shortest path distances between all node pairs (see, e.g., [50]),

$$\text{WIE}(G) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \Delta_{ij}^{\text{SP}} \tag{7.8}$$

  where Δ_{ij}^{SP} is the shortest path distance. The underlying idea is that if the sum of the distances between every node pairs is small, the network is more likely to be well-connected.

- The **Kirchhoff index** (KIR) is similar to the Wiener index but uses the resistance distance (the effective resistance, proportional to the commute-time distance based on a random walk on the graph) [147], instead of the shortest path distance, and has been recently used by Tizghadam and al. in network theory for quantifying the robustness of a communication network [236]. It can be easily computed by

$$\text{KIR}(G) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \Delta_{ij}^{\text{ER}} \tag{7.9}$$

  where Δ_{ij}^{ER} is now the effective resistance between i and j, with $\Delta_{ii}^{\text{ER}} = 0$ for each node i. The idea is thus the same as for WIE, but with a different concept of distance.

- The **Kemeny index** (KEM) represents the expected number of steps needed by a random walker for reaching an arbitrary node from some arbitrary starting node [142], when the starting and ending nodes are selected according to the equilibrium distribution of the Markov chain. Indeed, for an irreducible, aperiodic, Markov chain, it is known (see Chapter 2.7.6 or [164, 185]) that the stationary distribution exists and is independent of the initial state i. More precisely, the Kemeny index is

$$\text{KEM}(G) = \sum_{i=1}^{n} \pi_i \sum_{j=1}^{n} \pi_j m_{ij} = \sum_{j=1}^{n} \pi_j m_{ij} \tag{7.10}$$

where m_{ij} is the average first-passage time between node i and node j and $\boldsymbol{\pi}$ is the stationary distribution. Equation (7.10) holds because it can be shown that the quantity $\sum_{j=1}^{n} \pi_j m_{ij}$ is independent of the starting node i [77]. This index measures the relative accessibility of all pairs of nodes, putting more weight on the long-term frequently visited nodes according to the stationary distribution.

- The **Shield value** (SHV) has recently been introduced [239]:

$$\text{SHV}(G) = \lambda_1 \tag{7.11}$$

where λ_1 is the dominant eigenvalue of the adjacency matrix $\mathbf{A}$. It is closely related to the loop capacity and the path capacity of the graph, that is, the number of loops and paths of finite length. The higher λ_1, the more loops and long path in the graph. As for Estrada's centrality, the underlying idea is that if a graph has many such loops and paths then it is more likely to be well connected. The more the deletion of a node lowers this value, the less the graph becomes connected, and therefore the larger its criticality value.

- Assuming a connected graph, the **Algebraic connectivity** (ALC) is computed from the Laplacian matrix $\mathbf{L}$:

$$\text{ALC}(G) = \lambda_{n-1} \tag{7.12}$$

where λ_{n-1} is the second smallest eigenvalue of $\mathbf{L}$, see Section 2.4 for details. λ_{n-1} is related to the robustness of communication through the graph: the larger the algebraic connectivity, the more difficult it is to cut the graph into disconnected components [129].

7.3 The proposed BoP Criticality

We now derive a new node criticality measure called the bag-of-paths criticality (BPC). It is based on computing the effect of a node removal in the bag-of-paths framework (BoP, see Chapter 4). An illustrative example is shown in Section 7.3.3.

7.3.1 The bag-of-paths criticality: basic, standard case (BPC)

We now derive a closed-form formula for computing these probabilities when an intermediate node j is deleted from the graph. Then, our BoP criticality measure for node j is the relative entropy (or Kullback-Leibler divergence) between the bag-of-paths probabilities – the relative accessibility (see Equation (4.9)) – before and after removing node j from G. It therefore quantifies to what extent the relative accessibility is affected by the deletion of node j.

The intuition is the following. The bag-of-paths criticality measures the global impact of a node deletion on the total relative accessibility of the nodes in the network

- by computing this accessibility **before** and **after** node deletion,
- and then by computing their difference by means of the Kullback-Leibler divergence.
- This difference computes the **loss in accessibility** when deleting each node in turn.

Thus, in this work, a critical node is defined as a node whose deletion greatly affects the relative accessibility between the remaining nodes. This criticality measure will be referred as BPC. We now detail its derivation.

Reducing the support of the bag-of-paths probability distribution

First, let us introduce some new notation. In Equation (4.9), z_{ik} will be denoted as $z_{ik}(\mathbf{A})$ and $\mathbf{Z}$ as $\mathbf{Z}(\mathbf{A})$ since they are based on adjacency matrix $\mathbf{A}$. Then, as our criticality measure relies on the deletion of a node (say, node j), we need to reduce the support of the bag-of-paths probability distribution to $\mathcal{V} \setminus j$ (the set of nodes of G, with node j removed) by eliminating paths starting or ending in j.

To do this, we introduce $\mathbf{Z}^{(-j)}(\mathbf{A})$, which is $\mathbf{Z}$ based on $\mathbf{A}$ (the original graph), but where the jth column and the jth row of **Z have been removed**. Then, $z_{ik}^{(-j)}(\mathbf{A})$ with $i \neq j$ and $k \neq j$ is its i, k element. We further define

$\mathrm{P}_{ik}^{(-j)}(\mathbf{A}) = \mathrm{P}^{(-j)}(s = i, e = k)$ with support $\mathcal{V} \setminus j$ based on the elements of $\mathbf{Z}^{(-j)}(\mathbf{A})$,

$$\mathrm{P}_{ik}^{(-j)}(\mathbf{A}) = \frac{z_{ik}^{(-j)}(\mathbf{A})}{\sum\limits_{\substack{i',k'=1 \\ i',k' \neq j}}^{n} z_{i'k'}^{(-j)}(\mathbf{A})}, \text{ with } i, k \neq j \tag{7.13}$$

which corresponds to the BoP probabilities (see Equation (4.9)) based on the whole original graph ($\mathbf{A}$), but where the **support of the discrete probability distribution is reduced to the set of nodes different from** j – we do not consider node j as a potential source or destination node.

In practice, from this last equation, we observe that this can be done by putting both row j and column j of $\mathbf{Z}$ to 0 and then summing over its elements, as it is done in Algorithm 2 (line 7).

Computing the fundamental matrix after deleting one node from the graph

We now turn to the computation of the fundamental matrix $\mathbf{Z}$ of the graph after deleting node j from $\mathbf{A}$.

In this context, it is important not to confuse $\mathbf{Z}^{(-j)}(\mathbf{A})$ (introduced in the previous subsection) with $\mathbf{Z}(\mathbf{A}^{(-j)})$, which is defined as matrix $\mathbf{Z}$ computed from Equation (4.9), but based this time on $\mathbf{A}^{(-j)}$: the adjacency matrix $\mathbf{A}$ whose jth row and column have been removed (node j is deleted so that paths in the graph cannot visit this node any more). In other words, $\mathbf{A}^{(-j)}$ is the adjacency matrix of $G \setminus j$. Thus the $z_{ik}(\mathbf{A}^{(-j)})$ with $i \neq j$ and $k \neq j$ are the elements of $\mathbf{Z}(\mathbf{A}^{(-j)})$. Notice that $\mathbf{Z}^{(-j)}(\mathbf{A})$ and $\mathbf{Z}(\mathbf{A}^{(-j)})$ have the same size; both are $(n-1) \times (n-1)$ square matrices (node j is dismissed in both cases).

Then, from Equation (4.9), we define the bag-of-paths probabilities $\mathrm{P}(s = i, e = k | s, e \neq j)$ based on $\mathbf{A}^{(-j)}$ as

$$\mathrm{P}_{ik}(\mathbf{A}^{(-j)}) = \frac{z_{ik}(\mathbf{A}^{(-j)})}{\sum\limits_{\substack{i',k'=1 \\ i',k' \neq j}}^{n} z_{i'k'}(\mathbf{A}^{(-j)})}, \text{ with } i, k \neq j \tag{7.14}$$

corresponding to the graph G with node j removed.

The bag-of-paths criticality

Finally, the **bag-of-paths criticality** (BPC) is the **Kullback-Leibler divergence** between the bag-of-paths probabilities, quantifying relative accessibilities, before and after node removal,

$$\mathrm{cr}_j = \sum_{\substack{i,k=1 \\ i,k\neq j}}^{n} \mathrm{P}_{ik}^{(-j)}(\mathbf{A}) \log \left(\frac{\mathrm{P}_{ik}^{(-j)}(\mathbf{A})}{\mathrm{P}_{ik}(\mathbf{A}^{(-j)})} \right) \tag{7.15}$$

and the larger this divergence, the larger the impact of the deletion of node j on the overall accessibility.

Note that computing the bag-of-paths criticality for all the n nodes has a time complexity of about $O(n^3 + n(n-1)^3)$. The first term corresponds to the evaluation of $\mathrm{P}^{(-j)}(\mathbf{A})$ (which requires a matrix inversion) and the second term to n evaluations of $\mathrm{P}(\mathbf{A}^{(-j)})$ (inversion of n matrices, after deleting each node). This leads to an overall $O(n^4)$ time complexity. We now turn to a fast approximation of this quantity.

7.3.2 The bag-of-paths criticality: a fast approximation (BPCf)

In this subsection, we modify the bag-of-paths criticality to obtain a $O(n^3)$ time complexity instead of $O(n^4)$. It relies on the efficient approximation of the entries of $\mathbf{Z}^{(-j)}$ in terms of the fundamental matrix $\mathbf{Z} = (\mathbf{I} - \mathbf{W})^{-1}$. This version will be referred as BPCf.

The fast, approximate, bag-of-paths criticality

Let us first define

- $\mathbf{z}_j^{\mathrm{c}} = \mathbf{col}_j(\mathbf{Z}) = \mathbf{Z}\mathbf{e}_j$ and $\mathbf{z}_j^{\mathrm{r}} = \mathbf{row}_j(\mathbf{Z}) = \mathbf{e}_j^{\mathrm{T}}\mathbf{Z}$
- $\mathbf{w}_j^{\mathrm{c}} = \mathbf{col}_j(\mathbf{W}) = \mathbf{W}\mathbf{e}_j$ and $\mathbf{w}_j^{\mathrm{r}} = \mathbf{row}_j(\mathbf{W}) = \mathbf{e}_j^{\mathrm{T}}\mathbf{W}$

where $\mathbf{col}_j$ and $\mathbf{row}_j$ are respectively the jth column (a column vector) and the jth row (a row vector) of the matrix.

The main idea behind the approximation is to set row j of matrix $\mathbf{W}$ to zero[1] (providing $\mathbf{W}^{(-j)}$), instead of deleting row and column j of adjacency matrix $\mathbf{A}$, as required by the exact bag-of-paths criticality (see Equation (7.15)). Indeed, this approximation appears to be much simpler that the original problem and

[1]Note that we obtain the same result if we set both row j and column j of $\mathbf{W}$ to zero – this way of doing is equivalent to deleting row and column j of $\mathbf{W}$ and computing the fundamental matrix and the bag-of-paths probabilities from this reduced matrix. However, setting only row j to zero is simpler and provides the same results.

reduces the set of paths to paths avoiding j (as if node j was deleted), as shown below. Then, the bag-of-paths criticality is approximated by the Kullback-Leibler divergence between the bag-of-paths probabilities, as before,

$$\mathrm{cr}_j^{\mathrm{f}} = \sum_{\substack{i,k=1 \\ i,k \neq j}}^{n} \mathrm{P}_{ik}^{(-j)}(\mathbf{A}) \log \left(\frac{\mathrm{P}_{ik}^{(-j)}(\mathbf{A})}{\mathrm{P}_{ik}(\mathbf{W}^{(-j)})} \right) \tag{7.16}$$

using this time $\mathbf{W}^{(-j)}$ for computing the fundamental matrix and the bag-of-paths probabilities (instead of $\mathbf{A}^{(-j)}$ in Equation (7.15)). However, this only results in an *approximation* of the exact solution, as discussed later in Subsection 7.3.2. We now detail how to approximate efficiently the bag-of-paths probabilities from the matrix $\mathbf{W}^{(-j)}$.

Computing the fundamental matrix after setting row j of matrix W to zero

We now turn to the computation of the fundamental matrix $\mathbf{Z}$ of the graph after setting row j of $\mathbf{W}$ to zero.

Indeed, turning node j into a killing, absorbing, node (no outgoing edges from this node) can be achieved by defining a new matrix $\mathbf{W}^{(-j)} = \mathbf{W} - \mathbf{e}_j \mathbf{w}_j^{\mathrm{r}}$ as $\mathbf{W}$ is the elementwise (Hadamard) product between $\mathbf{P}^{\mathrm{ref}}$ and $\mathbf{C}$ (see Equation (4.4)). Doing so, row j of $\mathbf{W}$ is set to zero, meaning that node j cannot be an intermediate node anymore, as if node j was deleted. Thus paths connecting i and k (with $i, k \neq j$) cannot visit j any more: this node is excluded from the paths. Moreover, this actually corresponds to a simple rank-one matrix update.

By exploiting this property, we obtain a simple formula for the update of the fundamental matrix:

$$\mathbf{Z}(\mathbf{W}^{(-j)}) = (\mathbf{I} - \mathbf{W}^{(-j)})^{-1} = \mathbf{Z} - \frac{\mathbf{z}_j^{\mathrm{c}} \mathbf{z}_j^{\mathrm{r}}}{z_{jj}} \tag{7.17}$$

where only the entries $i, k \neq j$ of $\mathbf{Z}(\mathbf{W}^{(-j)})$ are meaningful. Recall that $\mathbf{z}_j^{\mathrm{c}}$ is a column vector while $\mathbf{z}_j^{\mathrm{r}}$ is a row vector. The rest of the subsection is dedicated to the derivation of this result and can be skipped at first reading.

Indeed, this results from a simple application of the Sherman-Morrison formula (see, e.g., [109, 174, 214]) for the inverse of a rank-one update of a matrix: if $\mathbf{c}$ and $\mathbf{d}$ are column vectors,

$$(\mathbf{B} + \mathbf{c}\mathbf{d}^{\mathrm{T}})^{-1} = \mathbf{B}^{-1} - \frac{\mathbf{B}^{-1}\mathbf{c}\mathbf{d}^{\mathrm{T}}\mathbf{B}^{-1}}{1 + \mathbf{d}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{c}} \tag{7.18}$$

Now, from $\mathbf{W}^{(-j)} = \mathbf{W} - \mathbf{e}_j \mathbf{w}_j^{\mathrm{r}}$, we have $(\mathbf{I} - \mathbf{W}^{(-j)}) = (\mathbf{I} - \mathbf{W}) + \mathbf{e}_j \mathbf{w}_j^{\mathrm{r}}$. By setting $\mathbf{B}^{-1} = \mathbf{Z}$, $\mathbf{B} = (\mathbf{I} - \mathbf{W})$, $\mathbf{c} = \mathbf{e}_j$ and $\mathbf{d} = (\mathbf{w}_j^{\mathrm{r}})^{\mathrm{T}}$ in Equation (7.18), we obtain for (7.17)

$$\mathbf{Z}(\mathbf{W}^{(-j)}) = (\mathbf{I} - \mathbf{W}^{(-j)})^{-1} = \mathbf{Z} - \frac{\mathbf{Z}\mathbf{e}_j \mathbf{w}_j^{\mathrm{r}} \mathbf{Z}}{1 + \mathbf{w}_j^{\mathrm{r}} \mathbf{Z} \mathbf{e}_j} \tag{7.19}$$

Let us first compute the term $\mathbf{w}_j^{\mathrm{r}}\mathbf{Z}$ appearing both in the numerator and the denominator of the previous equation. Since $\mathbf{Z} = (\mathbf{I} - \mathbf{W})^{-1}$, $(\mathbf{I} - \mathbf{W})\mathbf{Z} = \mathbf{I}$, and thus

$$\begin{aligned}\mathbf{w}_j^{\mathrm{r}}\mathbf{Z} &= ((\mathbf{w}_j^{\mathrm{r}})^{\mathrm{T}} - \mathbf{e}_j + \mathbf{e}_j)^{\mathrm{T}}\mathbf{Z} \\ &= -(\mathbf{e}_j - (\mathbf{w}_j^{\mathrm{r}})^{\mathrm{T}})^{\mathrm{T}}\mathbf{Z} + \mathbf{e}_j^{\mathrm{T}}\mathbf{Z} \\ &= -\mathbf{e}_j^{\mathrm{T}} + \mathbf{z}_j^{\mathrm{r}} = \mathbf{z}_j^{\mathrm{r}} - \mathbf{e}_j^{\mathrm{T}} \end{aligned} \tag{7.20}$$

From Equation (7.20), the denominator of the second term in the right-hand side of Equation (7.19) becomes

$$1 + \mathbf{w}_j^{\mathrm{r}}\mathbf{Z}\mathbf{e}_j = 1 + \left(\mathbf{z}_j^{\mathrm{r}} - \mathbf{e}_j^{\mathrm{T}}\right)\mathbf{e}_j = \mathbf{z}_j^{\mathrm{r}}\mathbf{e}_j = z_{jj} \tag{7.21}$$

Moreover, also from (7.20), the numerator of the second term in the right-hand side of Equation (7.19) is

$$\mathbf{Z}\mathbf{e}_j\mathbf{w}_j^{\mathrm{r}}\mathbf{Z} = \mathbf{z}_j^{\mathrm{c}}\left(\mathbf{z}_j^{\mathrm{r}} - \mathbf{e}_j^{\mathrm{T}}\right) \tag{7.22}$$

We substitute the results (7.21) and (7.22) in the denominator and the numerator of Equation (7.19), providing

$$\mathbf{Z}(\mathbf{W}^{(-j)}) = \mathbf{Z} - \frac{\mathbf{z}_j^{\mathrm{c}}\left(\mathbf{z}_j^{\mathrm{r}} - \mathbf{e}_j^{\mathrm{T}}\right)}{z_{jj}} \tag{7.23}$$

However, row and column j should neither be taken into account, nor used, and can therefore be put to zero. Indeed, since the last term of the numerator in Equation (7.23), $\mathbf{z}_j^{\mathrm{c}}\mathbf{e}_j^{\mathrm{T}}$, only updates the jth column, it can safely be ignored (this column j is useless and will never be used, as it corresponds to the deleted node), resulting in redefining the quantity as

$$\mathbf{Z}(\mathbf{W}^{(-j)}) = (\mathbf{I} - \mathbf{W}^{(-j)})^{-1} = \mathbf{Z} - \frac{\mathbf{z}_j^{\mathrm{c}}\mathbf{z}_j^{\mathrm{r}}}{z_{jj}}$$

and now the jth row as well as the jth column of $\mathbf{Z}(\mathbf{W}^{(-j)})$ are equal to zero. Indeed, elementwise, this last equation reads $z_{ik}(\mathbf{W}^{(-j)}) = z_{ik} - z_{ij}z_{jk}/z_{jj}$, which is equal to zero both when $i = j$ and $k = j$. We therefore obtain exactly Equation (7.17). Thus, the fundamental matrix $\mathbf{Z}$ needs to be **inverted only once** and the elements $z_{ik}(\mathbf{A}^{(-j)})$ in Equation (7.14) are approximated by $z_{ik}(\mathbf{W}^{(-j)})$ for computing the approximate bag-of-paths probabilities.

The resulting matrix has a jth row as well as a jth column equal to zero and it can be shown that each element $z_{ik}(\mathbf{W}^{(-j)})$ of $\mathbf{Z}(\mathbf{W}^{(-j)})$ corresponds to

$$z_{ik}(\mathbf{W}^{(-j)}) = \sum_{\wp \in \mathcal{P}_{ik}^{(-j)}} \tilde{\pi}^{\text{ref}}(\wp) \exp\left[-\theta c(\wp)\right] \tag{7.24}$$

where $\mathcal{P}_{ik}^{(-j)}$ is the set of paths avoiding node j.

The approximate bag-of-paths probabilities

The approximate bag-of-paths probabilities are computed from $\mathbf{W}^{(-j)}$ in the same way as for the standard bag-of-paths (see Equation (7.14)),

$$\mathrm{P}_{ik}(\mathbf{W}^{(-j)}) = \frac{z_{ik}(\mathbf{W}^{(-j)})}{\sum\limits_{\substack{i',k'=1 \\ i',k' \neq j}}^{n} z_{i'k'}(\mathbf{W}^{(-j)})}, \text{ with } i, k \neq j \tag{7.25}$$

where the elements of the fundamental matrix are computed from Equation (7.17) this time.

Finally, the fast approximation of the criticality measure is computed from these approximate bag-of-paths probabilities through Equation (7.16). The algorithm is detailed in Algorithm 2, where the probabilities $\mathrm{P}_{ik}^{(-j)}(\mathbf{A})$ and $\mathrm{P}_{ik}(\mathbf{W}^{(-j)})$ are respectively gathered in matrices $\mathbf{\Pi}$ and $\mathbf{\Pi}^{(-j)}$.

Discussion of the approximation

It should be noted that this procedure only computes an **approximation** of the BoP probabilities $\mathrm{P}_{ik}(\mathbf{A}^{(-j)})$ (defined in Equation (7.14)) when removing an intermediate node j. Indeed, for computing the exact probabilities on the graph $G \setminus j$, the natural random walk transition probabilities (the reference probability matrix $\mathbf{P}^{\text{ref}}$) should also be updated, as the edges entering node j cannot be followed any more. In our approximate procedure, these reference probabilities are **not updated** when computing $\mathbf{W}^{(-j)}$ (see Equation (4.4)), causing some (usually small) disturbance in comparison with explicitly deleting the node j and recomputing the quantities (including transition probabilities) from this new graph $G \setminus j$. Relative performances of the exact BoP criticality and the approximated BPCf criticality are investigated in the experiments.

Note that the expression could be adapted to exactly reflect node deletion, but the update formula becomes much more complex and we did not observe any significant difference between the two approaches in our experiments (see the experimental section).

One way to render the procedure exact would be to instead minimize expected cost subject to a fixed **entropy** constraint (as in [208]), instead of the **Kullback-Leibler** divergence in Equation (4.1). This results in redefining the $\mathbf{W}$ matrix as

$$\mathbf{W} = \exp[-\theta \mathbf{C}] \tag{7.26}$$

instead of (4.4). This solves the problem of the $\mathbf{P}^{\text{ref}}$ update since this transition matrix does not appear any more in the computation of $\mathbf{W}$ and $\mathbf{Z}$. However, experiments showed that this choice performs slightly worse (therefore not reported here) than the approximate update introduced in this section.

An elementary study of the empirical time complexity of the two versions BPC and BPCf is reported in Figure 7.1. Recall that the overall complexity for BPC is $O(n^4)$ and $O(n^3)$ for BPCf. For a 3000-nodes graph, the saving factor is greater than 10. Notice that no sparse or optimized implementation were used in the study. The CPU is a simple Intel(R) Core(TM) i5-4310 at 2.00 GHz with 8 Go RAM and the programming language is Matlab.

7.3.3 Illustrative example

A small toy graph, depicted on Figure 7.2, is now used as an illustrative example. This graph has six nodes: the (rounded) BPC value for each node is 6.3, 8.5, 5.5, 6.2, 7.1, 6.3, respectively. It corresponds to node ranking 2, 5, 6, 1, 4, 3.

7.4 Experimental comparisons

In this section, the bag-of-paths criticalities (both the exact one (BPC) and the fast approximate one (BPCf)) and the other centrality measures introduced in Section 7.2 are compared (see Table 7.1 and 7.2 for a reminder) on the two types of graphs described in Subsection 7.4.1. To do so, we followed a common methodology [9, 123, 82, 191, 209] described in subsection 7.4.2 and we report first a simple correlation analysis between rankings in Subsection 7.4.3. Then, results are compared and discussed in Subsection 7.4.4. Finally, the same comparison and discussion is applied to 20 real-life small social networks in Subsection 7.4.5.

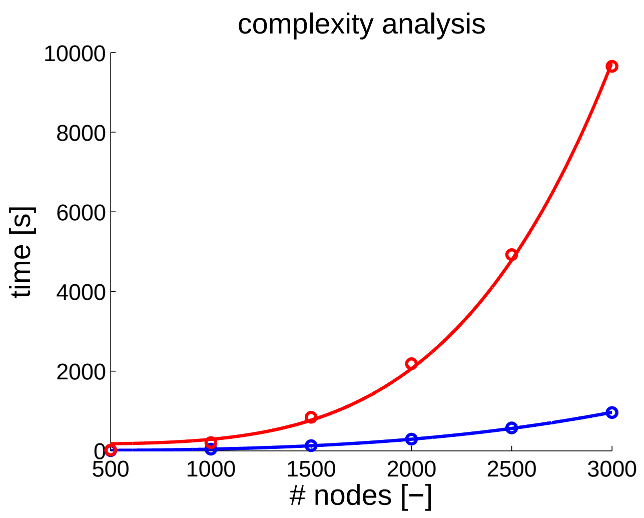


FIGURE 7.1: Empirical complexity analysis: computation time (in seconds) in function of network size (number of nodes). The overall complexity for BPC (upper curve) is $O(n^4)$ (a matrix inversion per node) and $O(n^3)$ for BPCf (lower curve, only one matrix inversion plus fast updates). We observe that BPCf scales better than BPC; for instance, for a 3000-nodes graph, the saving factor is larger than 10.

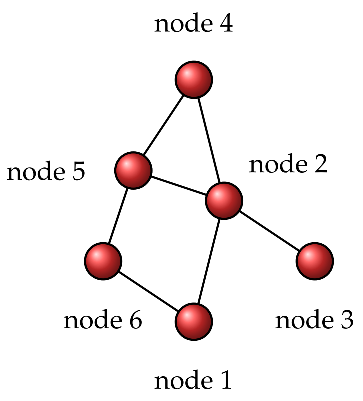


FIGURE 7.2: A small toy graph. The (rounded) BPC value for each node is 6.3, 8.5, 5.5, 6.2, 7.1, 6.3, respectively. It corresponds to the node ranking 2, 5, 6, 1, 4, 3, which seems legit. Conversely, WIE succeeds to identify node 2 as the most critical, but the second node in the ranking is node 3, which looks counter-intuitive.

TABLE 7.1: List of all measures compared in this study, together with their type, acronym, and computational time. Notice that Shortest Path and Random Walk Betweenness algorithms are fast, optimized, versions. The other algorithms were implemented in Matlab, as described in Section 7.2. Further notice that the Matlab implementation of the matrix exponential is very efficient (it is used for computing Estrada's node betweenness).

Name	**Type**	**Acronym**	**Description**	**Time**
Baseline (random disconnection)	-	BL	Subsection 7.4.2	$< 0.1s$
Edge Connectivity	Node Betw.	EC	See Eq. 7.2	$< 0.1s$
Shortest Path Betweenness	Node Betw.	SPB	See Eq. 7.3	$0.14s$
Random Walk Betweenness	Node Betw.	RWB	Subsection 7.2.1	$< 0.1s$
Estrada Index	Node Betw.	EST	See Eq. 7.4	$< 0.1s$
Wehmuth's K	Node Crit.	WK	See Eq. 7.5	$5.11s$
Klein Index	Node Crit.	KLE	See Eq. 7.6	$12.4s$
Wiener Index	Graph Crit.	WIE	See Eq. 7.8	32.0
Kirchhoff Index	Graph Crit.	KIR	See Eq. 7.9	23.5
Kemeny Index	Graph Crit.	KEM	See Eq. 7.10	51.2
Shield Value	Graph Crit.	SHV	See Eq. 7.11	$8.03s$
Algebraic connectivity	Graph Crit.	ALC	See Eq. 7.12	$7.09s$
Bag-of-paths criticality (standard)	Node Crit.	BPC	See Eq. 7.17	$10.0s$
Bag-of-paths criticality (fast)	Node Crit.	BPCf	See Eq. 7.15	$3.59s$

TABLE 7.2: List of measures requiring the tuning of a parameter. Tested values as well as the most frequent value (mode) are reported.

Name	**Param.**	**Tested values**	**Mode**
Wehmuth's K	h	[1,2,3,4,5,6]	1 (34%)
Bag-of-paths criticality (standard version)	θ	$10^{[-6,-3,-2,-1,0,1]}$	1 (48%)
Bag-of-paths criticality (fast version)	θ	$10^{[-6,-3,-2,-1,0,1]}$	10 (42%)

Algorithm 2 Computing the approximate bag-of-paths criticality of the nodes of a graph.

```
Input:
  – A weighted undirected graph $G$ containing $n$ nodes.
  – The $n \times n$ adjacency matrix $\mathbf{A}$ associated to $G$, containing affinities.
  – The $n \times n$ cost matrix $\mathbf{C}$ associated to $G$.
  – The inverse temperature parameter $\theta$.
Output:
  – The $n \times 1$ approximate bag-of-paths criticality vector $\mathbf{cr}$ containing the change in
    the probability distribution of picking a path starting in node $i$ and ending in node
    $k$, when each node $j$ is deleted in turn.
1.  $\mathbf{D} \leftarrow \mathbf{Diag}(\mathbf{Ae})$   ▷ the row-normalization matrix; $\mathbf{e}$ is a column vector full of 1s
2.  $\mathbf{P}^{\text{ref}} \leftarrow \mathbf{D}^{-1}\mathbf{A}$   ▷ the reference transition probability matrix
3.  $\mathbf{W} \leftarrow \mathbf{P}^{\text{ref}} \circ \exp[-\theta\mathbf{C}]$   ▷ elementwise exponential and multiplication $\circ$
4.  $\mathbf{Z} \leftarrow (\mathbf{I} - \mathbf{W})^{-1}$   ▷ the fundamental matrix
5.  for $j = 1$ to $n$ do   ▷ compute criticality for each node $j$ in turn
6.      $\mathbf{z}_j^{\text{r}} \leftarrow \mathbf{e}_j^{\text{T}}\mathbf{Z}$ and $\mathbf{z}_j^{\text{c}} \leftarrow \mathbf{Ze}_j$   ▷ copy row $j$ and column $j$ of $\mathbf{Z}$
7.      $\mathbf{Z}' \leftarrow \mathbf{Z} - \mathbf{e}_j\mathbf{z}_j^{\text{r}} - \mathbf{z}_j^{\text{c}}\mathbf{e}_j^{\text{T}} + z_{jj}\mathbf{e}_j\mathbf{e}_j^{\text{T}}$   ▷ set row $j$ and column $j$ of $\mathbf{Z}$ to 0 for dis-
        regarding paths starting and ending in $j$, but keeping those passing through
        $j$. Note that the last term is introduced because the diagonal element $z_{jj}$ is
        subtracted twice.
8.      $\mathbf{\Pi} \leftarrow \dfrac{\mathbf{Z}'}{\mathbf{e}^{\text{T}}\mathbf{Z}'\mathbf{e}}$   ▷ normalize in order to obtain the bag-of-paths probability
        matrix whose support is now $\mathcal{V} \setminus j$
9.      $\mathbf{Z}^{(-j)} \leftarrow \mathbf{Z} - \dfrac{\mathbf{z}_j^{\text{c}}\mathbf{z}_j^{\text{r}}}{z_{jj}}$   ▷ update of matrix $\mathbf{Z}$ when removing row $j$ from $\mathbf{W}$
10.     $\mathbf{\Pi}^{(-j)} \leftarrow \dfrac{\mathbf{Z}^{(-j)}}{\mathbf{e}^{\text{T}}\mathbf{Z}^{(-j)}\mathbf{e}}$   ▷ normalize in order to obtain the corresponding bag-
        of-paths probabilities after deletion of row $j$ of $\mathbf{W}$
11.     Remove both row $j$ and column $j$ from $\mathbf{\Pi}$ and $\mathbf{\Pi}^{(-j)}$
12.     $\boldsymbol{\pi} \leftarrow \mathbf{vec}(\mathbf{\Pi})$ and $\boldsymbol{\pi}^{(-j)} \leftarrow \mathbf{vec}(\mathbf{\Pi}^{(-j)})$   ▷ stack probabilities into column
        vectors by using the vec operator
13.     $\text{cr}_j \leftarrow (\boldsymbol{\pi}^{(-j)})^{\text{T}} \log(\boldsymbol{\pi}^{(-j)} \div \boldsymbol{\pi})$   ▷ compute Kullback-Leibler divergence
        with $\div$ being the elementwise division. It is assumed that $0 \log 0 = 0$ and
        $0 \log(0/0) = 0$
14. end for
15. return $\mathbf{cr}$
```

7.4.1 Datasets

We used two well-known graph generators [24, 43] to build a set of 200 graphs: 100 are generated using Erdős-Rényi's model and an additional 100 using Albert-Barabási's model. Each of these models has different variants; the one we used is detailed in Section 2.8. The number of nodes is set randomly for each graph between 5 and 500. Notice that by construction, all generated graph are unweighted.

7.4.2 Disconnection strategies

To study the performances of the different centrality/criticality measures, we simulate the effect of network attacks consisting in deleting its nodes sequentially in the order provided by the measure – the most critical nodes being deleted first. This is a natural way of assessing node criticality [9, 123]. We then record, for each network and each measure, the results of this sequential node deletion by measuring its gradual impact on network connectivity. A good criticality measure hurts most the network by, e.g., disconnecting it in several connected components, each preferably having an equal size – a balanced partition.

In practice, we first compute a criticality ranking of all nodes according to each different centrality/criticality measure introduced in the previous section. This ranking can be achieved in two different way: (1) it is computed once for all from the whole graph G (one **single ranking**), or (2) it is re-computed after each node deletion. With this last option, the centrality/criticality measures must be re-computed $n - 1$ times which is time-consuming. We therefore decided to update the ranking only 100 times in total (except, obviously, for graphs with $n < 100$ nodes). This last option will be referred as **100-ranking**.

Recall that, to evaluate the criticality of a node j with respect to a global graph criticality measure, the difference between the graph criticality of $G \setminus j$ and the global graph G criticality is computed (see Equation (7.7)).

Once those node rankings have been computed for each measure, the simulated attacks can start. Nodes are deleted in decreasing order of criticality. After each node deletion, the **Biggest Connected Component** size (BCC, the number of nodes contained in the largest connected component) is recorded [9, 123]. The smaller this value, the more effective the attack and thus the more effective the criticality index (see Figure 7.3 for an example). This performance measure quantifies to what extent the network is decomposed in several balanced parts (no **giant component** is left). If, for example, the node deletion strategy (the criticality ranking) is very inefficient, and it never disconnects the network, the BCC only decreases by one unit at a time. On the contrary, if it cuts the network into two equally sized parts, the BCC is divided by two, which corresponds to a large decrease.

By further normalizing with respect to the size of the graph, that is, dividing BCC by the current number of nodes, we get the **Relative Biggest Connected Component** size (RBCC) which is the performance indicator used in the experiments. It is then possible to draw a plot of RBCC versus the number of deleted nodes $(1, 2, 3, \ldots, n)$ [9, 123]. Then, to summarize those plots, we sum up the **Area Under the Curve** (AUC). The smaller this AUC, the better the method

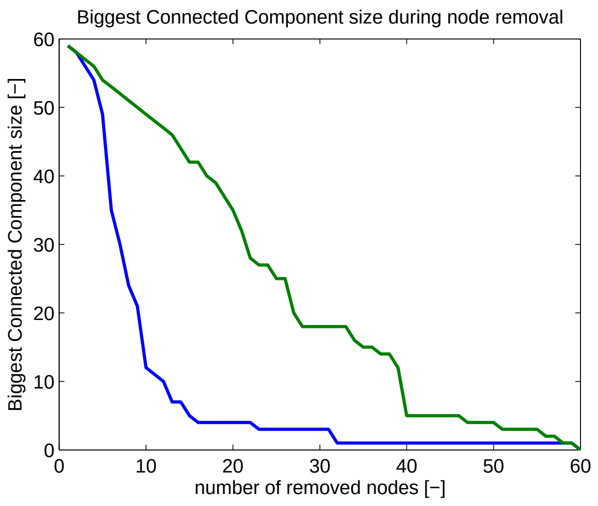


FIGURE 7.3: Example of Biggest Connected Component size recorded when nodes are removed following criticality rankings. The network is an Albert-Barabási (AB) 60-nodes graph. The two criticality rankings are BPC (lower curve) and BL (upper curve) and are computed once before starting to remove nodes. The BPC ranking is more efficient in detecting the critical nodes, as their removal quickly disconnects the network.

since the deletion of the most critical nodes (according to the ranking) quickly disconnects the network into balanced components, leading to smaller RBCC (see the illustrative example in Figure 7.3).

Finally, we report our results as follows: we perform a Friedman/Nemenyi test [74] and, in addition, we also compute the mean and the standard deviation of the AUC across all of the AB and ER generated graphs, providing more detailed results. Results can be found on Table 7.5 and 7.6; the higher the ranking, the better the criticality measure.

If a parameter is present, it is tuned as follows: for each graph, a range of values is tested and the best one is chosen for the disconnection experiment (the size of the graph can influence the parameter choice). This reflects the case of a real attack (we assume that the attacker has access to the network structure and can test the effect of different parameters). Parameters could be tuned again after each node deletion, but it would be too computationally intensive, so we did not investigate this approach. For information, best value of parameters h and θ are reported on Table 7.2.

For comparison, we also consider the case where nodes are simply removed at random and independently (BL for baseline). It corresponds to a **random failure** or a **random attack**, which has been studied theoretically in the literature (see [9] for an example).

7.4.3 Preliminary exploration: correlation analysis

The different centrality/criticality measures were first compared by computing two Kendall's correlation tests between each ranking. This is reported on Table 7.3 for both a small and a larger value of the parameters of our centrality/criticality measures: θ (BPCf and BPC) and h (WK). The small θ and h were set to 10^{-6} and 1, respectively, while the larger θ and h were 10 and 6. To summarize and to make things more visual, dendrograms were built above with a Ward hierarchical clustering (see, e.g., [249, 232, 155]) based on Kendall's correlation matrices (Figure 7.4).

7.4.4 Experimental results and discussion

Detailed results are presented in Tables 7.5, 7.6, 7.7, and 7.1 lists the different tested methods together with their acronym. Note that, when performing the Friedman/Nemenyi test comparing the different rankings provided by the methods, the critical difference is equal to 1.97, meaning that a measure is considered as significantly better than another if its rank is larger than this amount.

On Tables 7.5 and 7.6, we observe that on Albert-Barabási (AB), single ranking and 100-ranking, the Friedman/Nemenyi test [74] cannot conclude that our proposed model (BPC) is better than its approximation (BPCf), and vice versa. On the ER graphs, single ranking and 100-ranking, the results obtained by BPC and BPCf are significantly different but still close in comparison to the other criticalities. It means that the considered approximation seems reasonable, at least on the studied datasets.

We further observe from the same experiments (Tables 7.5 and 7.6) that BPC is significantly better than all the other tested measures on ER graphs. On AB graphs, it cannot be concluded that BPC is significantly better than RWB in the case where only one ranking is performed (single ranking). This is probably related to the fact that BPC is based on a random walk, as RWB is. Moreover, if an updated ranking is used instead (100-ranking), then BPC is not significantly better than WK – while still obtaining better performances. We conclude that the introduced criticality measures (BPC and BPCf) perform well in all contexts as they always perform better (and, most of the time, significantly better) than the competing measures, at least on the investigated data sets. However, this advantage is not always statistically significant when compared to RWB (single

TABLE 7.3: Mean Kendall's correlation between the investigated measures over our 200 graphs. Above the main diagonal: results for the larger θ and h (respectively 10 and 6). Below the main diagonal: results for the smaller θ and h (respectively 10^{-6} and 1).

	EC	SPB	RWB	EST	WK	KLE	WIE	KIR	KEM	SHV	ALC	BPC	BPCf
EC	**1.000**	0.878	0.889	0.776	0.975	0.345	0.709	0.756	0.685	0.735	0.433	0.764	0.852
SPB	0.878	**1.000**	0.879	0.673	0.856	0.399	0.667	0.736	0.642	0.637	0.452	0.765	0.817
RWB	0.889	0.879	**1.000**	0.632	0.874	0.404	0.637	0.804	0.627	0.599	0.423	0.798	0.871
EST	0.776	0.673	0.632	**1.000**	0.785	0.134	0.756	0.592	0.696	0.948	0.311	0.503	0.559
WK	0.338	0.311	0.293	0.134	**1.000**	0.329	0.747	0.793	0.721	0.752	0.427	0.738	0.815
KLE	0.345	0.399	0.404	0.134	0.332	**1.000**	0.140	0.332	0.109	0.106	0.356	0.465	0.443
WIE	0.709	0.667	0.637	0.756	0.075	0.140	**1.000**	0.782	0.838	0.768	0.301	0.569	0.576
KIR	0.756	0.736	0.804	0.592	0.161	0.332	0.782	**1.000**	0.734	0.600	0.350	0.759	0.770
KEM	0.685	0.642	0.627	0.696	0.086	0.109	0.838	0.734	**1.000**	0.708	0.297	0.504	0.535
SHV	0.735	0.637	0.599	0.948	0.096	0.106	0.768	0.600	0.708	**1.000**	0.292	0.490	0.539
ALC	0.433	0.452	0.423	0.311	0.302	0.356	0.301	0.350	0.297	0.292	**1.000**	0.377	0.392
BPC	0.756	0.757	0.789	0.476	0.307	0.504	0.482	0.682	0.445	0.454	0.411	**1.000**	0.885
BPCf	0.923	0.858	0.907	0.687	0.296	0.377	0.687	0.817	0.667	0.661	0.424	0.748	**1.000**

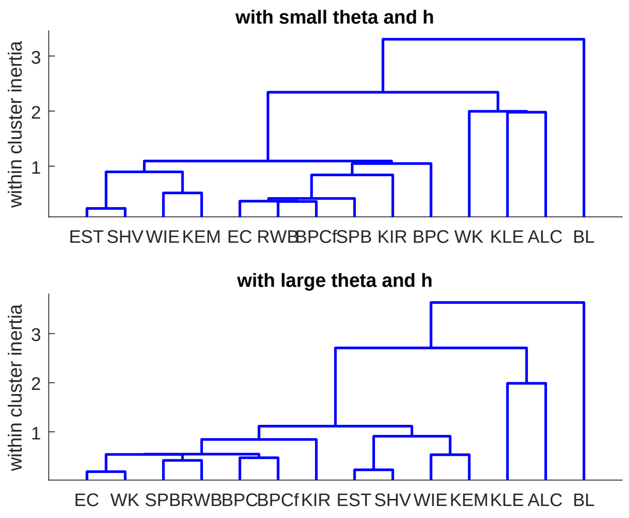


TABLE 7.4: Ward dendrograms of studied criticality measures. Distances are based on Kendall's correlation of Table 7.3. The smaller the height (Y-axis) of joining branches, the closer the measures. As BPCf, BPC and WK depend on a parameter, two cases are considered: a larger value of the parameters and a smaller value. The small θ and h are 10^{-6} and 1, respectively, while the larger θ and h are 10 and 6.

TABLE 7.5: Results obtained with the **single ranking** disconnection strategy described in Subsection 7.4.2. The Friedman/Nemenyi ranking is presented for 100 AB graphs and 100 ER graphs, together with the mean ± standard deviation of the obtained relative biggest connected component area under the curve (AUC). Critical difference is equal to 1.97. For the ranking, the larger is the better whereas, for AUC, smaller is better. Bold methods are not significantly different from the top method.

100 AB graphs: single ranking			100 ER graphs: single ranking		
measure	ranking	AUC	measure	ranking	AUC
BPC	**12.750**	**0.3092** ±**0.163**	**BPC**	**12.925**	**0.8634** ±**0.180**
BPCf	**12.265**	**0.3103** ±**0.164**	BPCf	10.695	0.8773 ±0.182
RWB	**11.415**	**0.3158** ±**0.167**	ALC	10.155	0.8851 ±0.167
KIR	10.285	0.4550 ±0.255	SPB	10.010	0.8851 ±0.167
WK	9.845	0.3246 ±0.175	RWB	9.780	0.8827 ±0.175
SPB	9.210	0.3283 ±0.172	KIR	8.925	0.8954 ±0.150
EC	8.780	0.3276 ±0.176	WK	7.855	0.8937 ±0.168
KLE	7.840	0.3577 ±0.208	EC	7.520	0.8959 ±0.165
WIE	5.015	0.5188 ±0.242	WIE	6.800	0.9112 ±0.131
ALC	4.890	0.4023 ±0.194	KEM	5.845	0.9092 ±0.145
KEM	4.830	0.5226 ±0.246	EST	4.405	0.9113 ±0.156
EST	3.790	0.4666 ±0.224	SHV	3.935	0.9207 ±0.129
SHV	2.615	0.5035 ±0.185	BL	3.400	0.9368 ±0.106
BL	1.470	0.7078 ±0.193	KLE	2.750	0.9273 ±0.134

TABLE 7.6: Results of the Friedman/Nemenyi test obtained with the **100-ranking** disconnection strategies. See Table 7.5 for details.

100 AB graphs: 100-ranking			100 ER graphs: 100-ranking		
measure	ranking	AUC	measure	ranking	AUC
BPCf	**12.565**	**0.2866** ±**0.151**	**BPC**	**13.285**	**0.7837** ±**0.173**
BPC	**11.860**	**0.2896** ±**0.152**	**ALC**	**11.535**	**0.7987** ±**0.150**
WK	**11.410**	**0.2924** ±**0.155**	BPCf	11.260	0.7994 ±0.175
RWB	10.555	0.2937 ±0.152	RWB	9.765	0.8052 ±0.177
EC	10.220	0.2953 ±0.157	KIR	8.665	0.8583 ±0.145
EST	9.110	0.2986 ±0.158	WK	8.515	0.8114 ±0.177
SPB	8.875	0.3048 ±0.155	SPB	8.325	0.8122 ±0.174
KLE	6.705	0.3278 ±0.170	EC	7.845	0.8143 ±0.175
KEM	5.150	0.5455 ±0.273	KEM	5.905	0.8695 ±0.141
KIR	5.080	0.5485 ±0.280	EST	5.440	0.8260 ±0.176
ALC	4.530	0.3557 ±0.169	WIE	5.305	0.8843 ±0.121
SHV	4.190	0.3701 ±0.171	SHV	4.010	0.8479 ±0.158
WIE	2.635	0.6075 ±0.271	KLE	3.095	0.8631 ±0.171
BL	2.115	0.6220 ±0.217	BL	2.050	0.9022 ±0.117

TABLE 7.7: Another perspective on the results obtained with the disconnection strategies described in Subsection 7.4.2. See Table 7.5 for details. The critical difference is equal to 4.30, meaning that a measure is significantly better than another if their rank difference is larger than this amount. For the ranking, the larger is the better while for AUC, the smaller is the better. In each column, the methods in bold are the best ones or are not significantly different from the overall best one.

100 AB graphs			100 ER graphs		
measure	ranking	AUC	measure	ranking	AUC
100-BPC	**25.790**	0.2896 ±0.152	**100-BPC**	**27.185**	0.7837 ±0.173
100-BPCf	**25.025**	0.2866 ±0.151	**100-BPCf**	**25.200**	0.7994 ±0.175
100-WK	**24.440**	0.2924 ±0.155	**100-ALC**	**24.945**	0.7987 ±0.150
100-RWB	**23.480**	0.2937 ±0.152	**100-RWB**	**23.640**	0.8052 ±0.177
100-EC	**23.065**	0.2953 ±0.157	**100-WK**	22.455	0.8114 ±0.177
100-EST	**21.810**	0.2986 ±0.158	**100-SPB**	22.130	0.8122 ±0.174
1-BPC	20.410	0.3092 ±0.163	**100-EC**	21.645	0.8143 ±0.175
1-BPCf	19.945	0.3103 ±0.164	**100-KIR**	20.045	0.8583 ±0.145
100-SPB	19.835	0.3048 ±0.155	**100-EST**	19.210	0.8260 ±0.176
1-RWB	18.560	0.3158 ±0.167	**100-SHV**	17.265	0.8479 ±0.158
1-WK	16.560	0.3246 ±0.175	**100-KEM**	17.220	0.8695 ±0.141
1-KIR	15.915	0.4550 ±0.255	**100-WIE**	16.195	0.8843 ±0.121
1-SPB	15.610	0.3283 ±0.172	**100-KLE**	15.110	0.8631 ±0.171
100-KLE	15.460	0.3278 ±0.170	**1-BPC**	14.960	0.8634 ±0.180
1-EC	15.275	0.3276 ±0.176	**1-BPCf**	12.325	0.8773 ±0.182
1-KLE	13.425	0.3577 ±0.208	**1-RWB**	11.435	0.8827 ±0.175
100-KIR	11.455	0.5485 ±0.280	**1-SPB**	11.400	0.8851 ±0.167
100-KEM	11.315	0.5455 ±0.273	**100-BL**	11.345	0.9022 ±0.117
100-ALC	10.930	0.3557 ±0.169	**1-ALC**	11.290	0.8851 ±0.167
100-SHV	10.835	0.3701 ±0.171	**1-KIR**	10.180	0.8954 ±0.150
1-ALC	7.890	0.4023 ±0.194	**1-WK**	9.095	0.8937 ±0.168
1-KEM	7.670	0.5226 ±0.246	**1-EC**	8.665	0.8959 ±0.165
1-WIE	7.630	0.5188 ±0.242	**1-WIE**	7.600	0.9112 ±0.131
1-EST	6.460	0.4666 ±0.224	**1-KEM**	7.040	0.9092 ±0.145
100-WIE	6.235	0.6075 ±0.271	**1-EST**	5.485	0.9113 ±0.156
1-SHV	4.765	0.5035 ±0.185	**1-SHV**	4.985	0.9207 ±0.129
100-BL	3.625	0.6220 ±0.217	**1-BL**	4.345	0.9368 ±0.106
1-BL	2.585	0.7078 ±0.193	**1-KLE**	3.605	0.9273 ±0.134

ranking on AB graphs) and WK (100-ranking on AB graphs). Notice that ALC obtains good performance on ER graphs (single ranking and 100-ranking), but is less efficient for AB graphs.

Besides this, when examining the results of the other criticality measures, we often find the RWB, KIR, WK, and SPB measures in the top-5 best methods (Tables 7.5 and 7.6). Note also that the EC (the degree) is quite efficient combined with multiple ranking on AB graphs, given its simplicity. At the bottom of the rankings, KLE, WIE, KEM, EST, and, SHV often appear to be even less effective than EC. Since EC is a really obvious measure that can be easily computed, it would certainly be interesting to use EC instead of other, more sophisticated, measures in many situations. In particular, EC is quite efficient on AB graphs, if recomputed after each node deletion. It can also be noted that KLE is not performing well on ER graphs (it can even be worse than the random baseline BL, but its mean AUC is still better). We unfortunately do not have a clear explanation of why this is the case.

All these conclusions are confirmed in Table 7.7 where the results of both disconnection strategies (single ranking and ranking updated (100-ranking)) are pooled in order to have an idea of the best method, independently of the ranking strategy.

It is also interesting to identify the most chosen θ and h parameter values from Table 7.8. For h, it depends on the task to fulfill but the best h value is usually small (1 to 4), and for θ it is better to take a value between 1 and 10. Notice that BPCf still exhibits the best mean rank when its parameter is fixed (results not presented here; see the discussion at the end of this section).

From Table 7.3, it is clear that WK's correlation with the other measures varies a lot depending of the h value. On the other hand, BPC's and BPCf's correlation with the other measures are less dependent of θ. Note that it was expected that those measures should be highly correlated with RWB and EC when θ is small and with SPB when θ is large, as the bag-of-paths betweenness does [145]. However, we observe that this is not the case for a large θ: the criticality measures BPC and BPCf are still more correlated with RWB when $\theta = 10$. This suggests that the proposed measures capture different properties than the bag-of-paths betweenness.

In Figure 7.4, we once more notice that the behavior of WK is strongly dependent of h. It turns out that with small h, its behavior is similar to KLE and ALC. When h is larger, the neighborhood is more and more likely to be close to the whole graph, therefore more and more correlated to EC. As from Table 7.3, BPC's and BPCf's behavior are less sensitive to θ and are close to RWB, SPB and EC.

From visual inspection of Table 7.4, we can identify different clusters of measures:

TABLE 7.8: Number of times each value of the parameters is selected during the disconnection strategies described in Subsection 7.4.2. Note that only WK, BPCf, and BPC need a parameter tuning. Bold values show the maximum per task and per measure.

Measure	Parameter value	100 AB graphs: single ranking	100 ER graphs: single ranking	100 AB graphs: 100 rankings	100 ER graphs: 100 rankings	Sum over the 4 tasks
WK	$h = 1$	28	17	**71**	20	**136**
	$h = 2$	7	**68**	4	7	86
	$h = 3$	**29**	14	9	**26**	78
	$h = 4$	25	0	7	24	56
	$h = 5$	8	1	2	14	25
	$h = 6$	3	0	7	9	19
BPC	$\theta = 10^{-6}$	16	21	7	18	62
	$\theta = 0.001$	6	1	2	4	13
	$\theta = 0.01$	17	3	2	1	23
	$\theta = 0.1$	24	8	14	13	59
	$\theta = 1$	12	**52**	73	**56**	**193**
	$\theta = 10$	**25**	15	2	8	50
BPCf	$\theta = 10^{-6}$	6	24	**39**	13	82
	$\theta = 0.001$	6	0	11	0	17
	$\theta = 0.01$	6	1	11	3	21
	$\theta = 0.1$	8	2	23	17	50
	$\theta = 1$	18	6	13	26	63
	$\theta = 10$	**56**	**67**	3	**41**	**167**

- ▶ WIE, KEM, SHV, and EST seem to form a cluster. This is a bit surprising as these measures are based on different properties of the graph, but still provide relatively similar results. Indeed, WIE is based on shortest paths, KEM is based on random walks, SHV is based on an eigenvalue of $\mathbf{A}$ and EST on paths of different lengths.

- ▶ SPB, RWB, KIR, EC, BPCf and BPC are part of another cluster. The same observation can be made: if RWB, BPCf and BPC are based on random walks, SPB is based on shortest paths and KIR is based on the spectrum of the Laplacian matrix. Notice that SPB, RWB, KIR, BPCf and BPC tend to show good performances on Tables 7.5, 7.6 and 7.7.

- ▶ KLE and ALC look apart, but are correlated to WK when h is small.

- ▶ Finally, notice that the random baseline BL is the last merged measure in the two cases, which looks natural.

Before closing the discussion, let us comment on the presence of parameters. At first sight, it seems unfair to compare measures depending on a parameter (WH, BPC, and BPCf) against measures free of parameter. Recall, however, that the attacker can adapt its behavior to the network structure, so that a

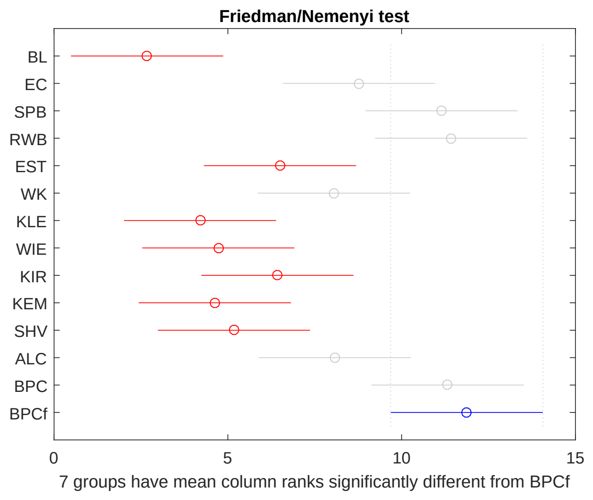


FIGURE 7.4: Mean rank (circles) and critical difference (plain line) of the Friedman/Nemenyi test, for the 20 real-life small social networks datasets. The blue method (the fast version of our proposed algorithm) has the best mean rank and is statistically better than red methods. The critical difference is 4.02.

parameter monitoring the smoothing scale can be considered as an advantage. Moreover, let us recall two facts about the parameter θ of BPC and BPCf. First, measures are not very sensitive to the parameter and, second, its optimal value (according to our experiments) is often close to 1 or 10. Therefore, it seems that we could also just fix this parameter. By the way, we reproduced the experiments by setting $\theta = 1$ and it turns out that BPC had still the best mean rank for three disconnection tasks while the BPCf was the best for the last one (the **single ranking** on AB graphs, experiments not reported here).

7.4.5 Application to real-world small social networks

We now consider some experiments on small real-world social networks. The considered datasets consist in 20 symmetric adjacency matrices, coming from small social networks, with a minimum of 14 nodes and a maximum of 58. These data were selected from the well-known UCINET IV repository [26], and we consider all symmetric datasets, removing antagonistic (negative) networks, tree-shaped networks and network with more than two time views.

Weighted networks were converted to unweighted networks, using $a_{ij} = 0$ if this entry was equal to zero in the original dataset and $a_{ij} = 1$ otherwise.

Then, the same RBCC analysis, after removing nodes according to the different rankings methods, were performed. The results are presented on Figure 7.4 as a Friedman/Nemenyi test (see Appendix A for details). Notice that since those networks are quite small, and also for clarity, only the single ranking experiment was performed. For the parameters, θ and k were set to 1 and 2, respectively.

On this set of networks, the top-7 methods are, in decreasing order: BPCf, RWB, BPC, SPB, EC, ALC, and WK. Those seven methods are not significantly different than BPCf, which has the larger mean rank. Note that these results are quite similar to those reported for the 100 Albert-Barabási's graphs in Table 7.5. Actually the top-3 methods are exactly the same, with a few changes. KIR performs less well on this small set of social networks compared to AB generated graphs. The opposite behavior can be observed for ALC. Finally, WK, SPB and EC are only slightly affected.

Notice that on this set of 20 small social networks and in this configuration, there is no evidence that the three top methods (RWB, BPC and BPCf) significantly outperform EC, which is partly due to the high number of investigated methods, compared to the number of datasets. We therefore decided to remove the other methods (EC, SPB, EST, WK, KLE, WIE, KIR, KEM, SHV, and ALC) to increase the power of the test. Then, both BPC and BPCf become significantly better than EC.

7.5 Conclusion

This chapter investigated centrality/criticality measures on graphs through a node disconnection analysis and introduced a new criticality measure based on a bag-of-paths framework: the bag-of-paths criticality and its fast, approximate, version.

Comparisons based on node disconnection simulations performed on a large number of generated graphs show that those two bag-of-paths criticality measures outperform the other considered centrality/criticality measures. Friedman/Nemenyi tests confirm this fact statistically in most of the cases. Our results are further confirmed on 20 real-life small social networks.

Of course, the node disconnection analysis is only a proxy to determine if our criticalities are able to identify critical nodes. Our future work will mainly focus on testing the proposed measures on other tasks and to consider other strategies, such as disconnecting groups of nodes instead of one single node at each time.

Finally, a simple correlation analysis of those measure allowed to identify coherent groups, namely the WIE, KEM, SHV, and EST versus the SPB, RWB, KIR, EC, BPCf and BPC. It was also shown that the choice of the θ parameter does not impact much the behavior of our two proposed criticality measures.

This study has also some limitations. It would, for instance, be important to confirm the results on larger, real-world, networks and to adapt the algorithm in order to scale on large graphs. Moreover, other criticality, vulnerability, and betweenness measures not considered here should be investigated as well [162, 50].

Chapter 8

Graph-based fraud detection

Nowadays, **e-commerce** becomes more and more important for global trade: sales of goods and services represented more or less 2,000 billion dollars in 2014 and it was estimated that on 7,223 millions peoples on earth, 20 % were e-shoppers [81]. Part of the reasons of this success is easy online credit card transactions and cross-border purchases. Furthermore, most organizations, companies and government agencies have adopted e-commerce to increase their productivity or efficiency in trading products or services [203].

Of course, e-commerce is used by both legitimate users and fraudsters. The Association of Certified Fraud Examiners (ACFE) defines fraud as: “the use of one’s occupation for personal enrichment through the deliberate misuse or misapplication of the employing organization’s resources or assets” [15].

Global card fraud losses amounted to 16.31 Billion US dollar in 2014 and is forecast to continue to increase [124]. This huge number of losses has increased the importance of fraud fighting: in a competitive environment, fraud have a serious business impact if not managed, and prevention (and repression) procedures must be undertaken.

As in many domains, profit-motivated fraudsters interact with the affected business. [193, 19] describes comprehensively this interaction: the fraudster can be internal or external to the business, can either commit fraud as a customer (consumer) or as a supplier (provider), and has different basic profiles. From this description, it comes out that professional fraudsters’ (as opposed to occasional ones) modus operandi changes over time. Therefore, fraud detection system algorithms should also adapt themselve to new behaviors. This is refered as **concept drift**: the constant change in fraudsters behavior.

For those reasons e-commerce and credit card issuers need automated systems that identify incoming fraudulent transactions or transactions that do not correspond to a normal behavior. Data mining and machine learning offer various techniques to find patterns in data; here, the goal is to discriminate between genuine and fraudulent transactions. Such **Fraud Detection Systems**

(FDS) exist and are similar to detection approaches in **Intrusion Detection System** (IDS). FDS use misuse and anomaly based approaches to detect fraud [86].

However, there are issues and challenges that hinder the development of an ideal FDS for e-commerce system [72]; such as,

- **Concept drift**: fraudsters conceive new fraudulent ways/methods over time. Furthermore, normal behavior also varies with time (consumption peak at Christmas for instance).
- **Six-seconds rule** [241]: acceptance check must be processed quickly as the algorithm must decide within six seconds if a transaction can be pursued.
- **Large amount of data**: millions of transactions occur per day whereas have to be analyzed and acceptance must be granted in seconds.
- **Unbalanced data**: frauds represents (hopefully) only less than 1% of transactions but predicting a pattern is harder with unbalanced datasets.

The presence of those challenges leads to high false alert rate, low detection accuracy or slow detection (see [1] for more details). There are many fraud detection domains but internet e-commerce presents a challenging data mining task because it blurs the boundaries between fraud detection systems and network intrusion detection systems.

This work focuses on automatically detecting e-commerce fraudulent transactions using network (or graph) related features. e-commerce refers to a goods (or services) exchange through a computer network, often internet. It contrasts to **face-to-face** transactions, where the buyer and merchant physically meet each other. Furthermore, for a payment, two points of view are possible, since two actors intervene [195]:

- **Issuing**: The issuer (cardholder's bank) confirms the account holder identity, check PIN and verifies that the account balance is sufficient, and then authorizes the transaction.
- **Acquiring**: The acquirer (merchant's bank) checks that the merchant account exists and send an acceptance agreements.

In this chapter, the data we are working on are issuing data. They are therefore encoded from the issuer point of view. More details can be found on Table 8.1 (p. 132).

Figure 8.1 [195] describes the five main steps of a real-world FDS for each new transaction. This description is strongly inspired from [195]. These steps can be summarized as follows :

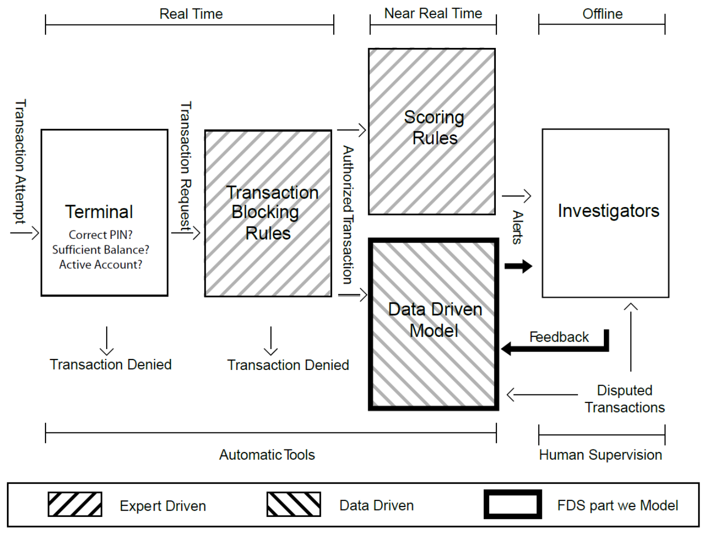


FIGURE 8.1: A scheme illustrating the layers of control in a FDS. From [195], reproduced with permission.

- **Terminal**: As a first step, the terminal queries a server of the card issuing company and controls the PIN code, the number of attempts, the card status (either active or blocked), the balance available and the expenditure limit. For online transactions, these controls have to be executed in **real time** (response in a few milliseconds). The transaction can be denied at this step; if not, the transaction proceeds to transaction-blocking rules.
- **Transaction-blocking rules**: Transaction-blocking rules are a collection of if/then/else statements on the available information when the payment is requested (i.e. without analyzing historical records or cardholder profile). Transactions firing any of these rules are blocked. Transaction-blocking rules are **expert-driven**: they are manually designed by investigators with two constrains: quick to compute (in real-time) and very precise (very few false alarms). All transactions passing blocking rules are authorized. Data are stacked in a feature vector and new, aggregated, features are added (for example, the average number of transaction per day). The following steps are then computed on this new feature set.
- **Scoring rules**: Scoring rules are another set of expert-driven models that are expressed as if/then/else statements. However, these operate on feature vectors and assign a score to each authorized transaction: the

larger the score, the more likely the transaction to be a fraud. Scoring rules can detect only simple fraudulent strategies that have already been discovered by investigators.

- **Data driven model**: This steps is not expert-driven and is purely data driven. It is based on classifiers or another statistical models to estimate the probability for each feature vector being a fraud. It is expected to find frauds according to rules that go beyond investigator experience, and that do not necessarily correspond to interpretable rules. Transactions associated with feature vectors that have either received a large fraud score or an high probability of being a fraud, generate alerts. A limited number of alerted transactions are finally reported to investigators.

- **Investigators**: Investigators are human professionals which are in charge of the expert-driven steps. Investigators also call cardholders and, after verification, assign the label "genuine" or "fraudulent" to the alerted transaction, and return this information as feedbacks to the FDS.

  Any card that is found victim of a fraud is immediately blocked, to prevent further fraudulent activities. Typically, investigators check all the recent transactions from a compromised card, which means that each detected fraud can potentially generate more than one feedback. In a real-world FDS, investigators can only check few alerts per day as this process can be long and tedious. Therefore, the primary goal of the data driven step is to return precise alerts, as investigators might ignore further alerts when too many false alarms are reported.

Our work is based on a recent paper [241] which introduced an automated and field-oriented approach to detect fraudulent patterns in credit card transactions by applying supervised data mining techniques. More precisely, this algorithm uses a collective inference algorithm to spread fraudulent influence through a network by using a limited set of confirmed fraudulent transactions and takes a decision based on risk scores of suspiciousness of transactions, card holder, and merchants. A toy example can be found on Figure 8.2.

In this chapter, several improvements from graph literature and semi-supervised learning are proposed. The resulting fraud detection method is tested on a three-months real-life e-commerce credit card transaction data set obtained from a large credit card issuer in Belgium.

The following questions are addressed:

1. Can we enhance graph-based existing FDS in terms of performance?
2. How can we make graph-based FDS as suitable for real applications as possible?

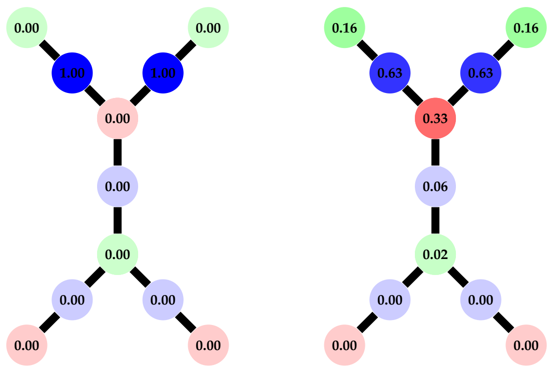


FIGURE 8.2: A small toy graph. Left sub-figure shows five transactions (in blue) between three card holders (in green) and three merchants (in red). Two transactions are known to be frauds (risk score of 1.00). Right sub-figure is obtained after a random walk with restart process. A merchant (risk score of 0.33) is clearly potentialy more fraudulent than the others. Those risk scores are then used as features in a classifier. For this example α was set to 0.5.

3. Is semi-supervised learning [58] or feedback [72] useful for this graph-based FDS?

Our approach takes into account various field/ground realities such as the six-second rule, concept drift, dealing with large datasets and unbalanced data. It also has been conceived in accordance with field experts to guarantee its applicability.

The rest of this chapter is divided as follows: Section 8.1 reviews related work. Section 8.2 details the proposed contributions. Experimental comparisons are presented in Section 8.3 and analyzed in Section 8.4. Finally, Section 8.5 concludes this chapter.

8.1 Related work

Credit-card Fraud detection received a lot of attention, but the number of publications available is limited. Indeed, credit card issuers protect data sources and most algorithms are produced in-house, concealing the model's details [241].

As for any machine learning modeling process, two main approaches can be used: a **supervised** and an **unsupervised** scheme (see Chapter 3). Supervised learning uses labels (the observed prediction of an instance, here the fraud tag) to build the classification model, where unsupervised simply extracts clusters of similar data that are then processed. Common unsupervised techniques are peer group analysis [253] and self-organizing maps [259] while common supervised techniques are logistic regression, artificial neural networks (ANN), random forests, meta-learning, case-based reasoning, Bayesian belief networks, decision trees, hidden Markov models, association rules, support vector machines,... The reader is advised to consult [71] for more detail about credit card fraud detection, and [193] for a wider review on fraud detection.

In [72], the authors address a realistic fraud-detection setting with investigator's feedbacks and delayed labels. We therefore used these characteristics to define our realistic scenario for fraud detection. We were also inspired by [71] and [70].

[20] propose a measure that realistically represents the monetary gains and losses due to fraud detection. Moreover, In [21], the authors propose an example-dependent cost matrix for credit scoring. This example-dependent cost is then developed for several models (naive Bayes model, logistic regression,...).

In [22] and [70] the authors create new sets of features using various transaction aggregation strategies. We did not consider this option as the main goal of this work is to study the usage of graphs for fraud detection. Nevertheless, both approaches (transaction aggregation and graph-based) can easily be combined by concatenating the new obtained features.

According to [241], APATE was the only one to include network knowledge in the prediction models for fraud detection at that time: This model first builds a **tripartite graph** (see below) and then extracts relevant risk scores for each node. [241] shows that this information, added to more conventional ones, increases the performances of the fraud detection system.

In this work, we follow the methodology of APATE [241] (which is described in this section for clarity), and propose several improvements in the next section. In [52], transactions between cards and shops are represented as a **bipartite graph**. This approach finds fully connected subgraphs containing mostly compromised cards, because such bicliques reveal suspicious payment patterns. It then defines new features capturing transaction suspiciousness from those suspicious payment patterns. Other types of graph were also investigated but they did not provide better results and are therefore not presented here.

In particular, APATE starts with a set of time stamped, labeled, transactions. The goal is, of course, to fit a model to infer future fraudulent/genuine transactions. Furthermore, for each transaction of this dataset, the card holder (or user) and merchant (or retailer) is known. APATE thus creates a tripartite adjacency matrix $\mathbf{A}^{\text{tri}}$ (there are three type of node: transactions, card users, and merchants) as follows:

$$\mathbf{A}^{\text{tri}} = \begin{pmatrix} \mathbf{0}_{t\times t} & \mathbf{A}_{t\times c} & \mathbf{A}_{t\times m} \\ \mathbf{A}_{c\times t} & \mathbf{0}_{c\times c} & \mathbf{0}_{c\times m} \\ \mathbf{A}_{m\times t} & \mathbf{0}_{m\times c} & \mathbf{0}_{m\times m} \end{pmatrix} \tag{8.1}$$

where $\mathbf{A}_{t\times c} = (\mathbf{A}_{c\times t})^{\text{T}}$ is an adjacency matrix where transactions are linked with their corresponding card holders, $\mathbf{A}_{t\times m} = (\mathbf{A}_{m\times t})^{\text{T}}$ is an adjacency matrix where transactions are linked with corresponding merchants and $\mathbf{0}_{\dots\times\dots}$ is a correctly sized matrix full of zeros. From $\mathbf{A}^{\text{tri}}$, the transition matrix $\mathbf{P}$ can be derived by defining a random walk on the tripartite graph (see Section 2.4).

This tripartite graph exhibits another interesting behaviors: it follows the scale-free rule. The degree distribution of such a network follows a power law $P(k) \sim k^{-\gamma}$, where $P(k)$ is the probability that a vertex in the network is linked to k other vertices [24]. This kind of network is often observed in natural and human-generated systems, including the world wide web, citation networks, and social networks, with γ between two and three. It is closely related to Albert-Barabási's graphs (see Section 2.8.1). Figure 8.3 shows the distribution $P(k)$ for one of the tripartite graphs used in Section 8.3, with a γ slightly lower than three. Notice that nodes with a degree of two deviate from the curve: by construction, transaction nodes are constrained to have such a degree (one merchant and one card holder per transaction).

A column vector $\mathbf{risk}(0) = [\mathbf{vec}_0^{\text{Trx}}; \mathbf{vec}_0^{\text{CH}}; \mathbf{vec}_0^{\text{Mer}}]$ of length equal to the total number of transactions (hence the superscript Trx), card holders (CH) and merchants (Mer) is also created and initialized. $\mathbf{risk}(t)$ is the **risk vector** after t iteration. Vector $\mathbf{vec}_0^{\text{Trx}}$ is initially full of zeros, except for known fraudulent transactions where it is equal to one (at least at iteration 0, indicated by the corresponding subscript), and $\mathbf{vec}_0^{\text{CH}}$ and $\mathbf{vec}_0^{\text{Mer}}$ are column vector full of zeros. Finally, element k of a vector $\mathbf{risk}(0)$ is noted $[\mathbf{risk}(0)]_k$.

Then, in a convergence procedure similar to the **PageRank** algorithm [187], $\mathbf{risk}(t)$ is updated to spread the fraud label through the tripartite graph. This is known as a random walk with restart procedure (RWWR) [141]:

$$\mathbf{risk}(t) = \alpha \cdot \mathbf{P}^{\text{T}}\mathbf{risk}(t-1) + (1-\alpha) \cdot \mathbf{risk}(0) \tag{8.2}$$

where $\mathbf{risk}(t)$ is the risk vector after t iteration, α is the probability to continue the walk and $(1-\alpha)$ is the probability to restart the walk from a fraudulent transaction. This parameter could be tuned, but was fixed to 0.85 in the experimental comparisons (see [187]). The procedure diffuses the information about the transactions through the network.

Eq. 8.2 is iterated until convergence. Then, from $\mathbf{risk}(tc)$ (where tc stands

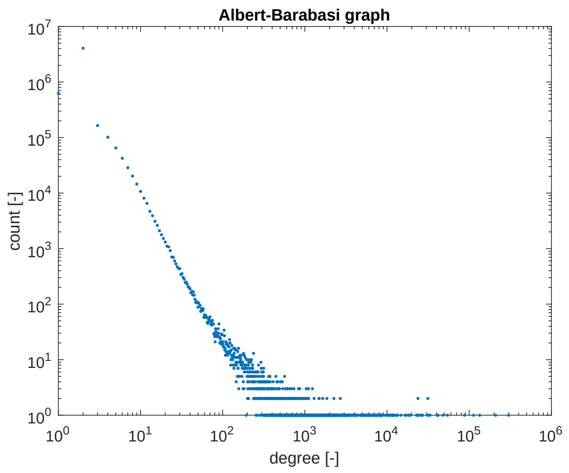


FIGURE 8.3: The distribution $P(k)$ for one of the scale-free tripartite graph used in Section 8.3. γ is slightly lower than three. We use logarithmic scales.

for t at convergence) $\mathbf{vec}_{tc}^{\mathrm{Trx}}$, $\mathbf{vec}_{tc}^{\mathrm{CH}}$ and $\mathbf{vec}_{tc}^{\mathrm{Mer}}$ can be extracted and considered as a **risk measure** (at convergence) for each transaction, card holder, and merchant respectively.

As fraud detection models should adapt dynamically to a changing environment, this procedure is repeated several times, introducing a time decay factor. Intuitively, we want that old transactions matter less than recent one. Each non-zero entry of $\mathbf{A}^{\mathrm{tri}}$ and $\mathbf{risk}(0)$ is modified to characterize transactions based on current and normal customer's past behavior (see [241] for more details):

$$\begin{cases} [\mathbf{A}^{\mathrm{tri}}]_{ij} & \leftarrow & e^{-\gamma \cdot t([\mathbf{A}^{\mathrm{tri}}]_{ij})} & \text{or 0 if no relation} \\ [\mathbf{risk}(0)]_k & \leftarrow & e^{-\gamma \cdot t([\mathbf{risk}(0)]_k)} & \text{or 0 if no fraud} \end{cases} \tag{8.3}$$

where $t(\cdot)$ (a time function, not to be confused with t, the iteration number of Equation (8.2)) is the (scalar) time where transaction between i and j in matrix $\mathbf{A}^{\mathrm{tri}}$ occurred (or k for vector $\mathbf{risk}(0)$), and γ (the exponential decay constant) is a scalar set in such a way that the exponential equals 0.5 after: one day, one week and one month. For instance, if a transaction occured two weeks ago, the corresponding element of $\mathbf{A}^{\mathrm{tri}}$ with week decay is equal to $1/(2^2)$ and is $1/(2^{14})$ with day decay.

Therefore, for each transaction of our starting dataset, we have 12 new features: The **risk score** for transaction, card holder, and merchant, each for four (no decay, day decay, week decay, and month decay) time windows.

However, this procedure cannot be computed in less than a few minutes, which is not suitable with the six-seconds rule. Convergence on a graph with millions of nodes is expensive and is therefore daily re-estimated over night. Transactions appearing during the testing day are evaluated using the model trained on previous night. For card holders and merchants, the graph-based feature values are extracted (looked up) from already the trained model, since they are likely to be part of the previous data.

Naturally, for the new transaction not part of the model, transaction-based risk score have to be estimated, which is done through the formula [241]:

$$\mathrm{sc}(\mathrm{Trx}^{j}_{i,k}) = \frac{1}{\sum_{j=1}^{n} p_{ji} + 1}\mathrm{sc}(\mathrm{Mer}_i) + \frac{1}{\sum_{j=1}^{m} p_{jk} + 1}\mathrm{sc}(CH_k) \tag{8.4}$$

where $\mathrm{sc}(\mathrm{Trx}_{i,k})$ stands for the new transaction risk score between merchant i and card holder k, $\mathrm{sc}(\mathrm{Mer}_i)$ stands for the score of merchant i and $\mathrm{sc}(\mathrm{CH}_k)$ stands for the score of card holder k. It represents the score of a new transaction j after one new iteration of Eq. 8.2.

Equation (8.4) is obtained as follows: Recall that $\mathbf{risk}(tc)$ is the vector obtained after convergence of Equation (8.2). Then we rewrite Equation (8.2) considering that we run a single new iteration after convergence. $[\mathbf{risk}(t+1)]_j$ is then used as $\mathrm{sc}(\mathrm{Trx}^{j}_{i,k})$ [241]:

$$\mathbf{risk}(tc+1) = \alpha \cdot \mathbf{P}^{\mathrm{T}}\mathbf{risk}(tc) + (1-\alpha) \cdot \mathbf{risk}(0) \tag{8.5}$$

If a new transaction $\mathrm{Trx}^{j}_{i,k}$ (we introduce an new node j) between merchant i and card holder k occurs, matrices $\mathbf{A}^{\mathrm{tri}}$ and $\mathbf{P}^{\mathrm{T}}$ should be extended with a new row and a new column (considering new edges $a_{ij} = a_{ji} = a_{ik} = a_{ki} = 1$, since the transaction just occurs). $\mathbf{risk}(tc)$ and $\mathbf{risk}(0)$ should also be extended by a new row, and this new jth score value is unknown at this point for both $\mathbf{risk}(tc)$ and $\mathbf{risk}(0)$. We can assume any value for $[\mathbf{risk}(tc)]_j$: its value needs two step to influence $[\mathbf{risk}(t+1)]_j$ and we limit ourself to only one step. As we have no clue for $[\mathbf{risk}(0)]_j$ when receiving the transaction (it was not present in the train set), we replace it by $[\mathbf{risk}(t+1)]_j$, the updated value we are looking for.

TABLE 8.1: Features used by the random forest classifier. The first group contains demographical features and the second group are graph-based features. Notice that each transaction is linked with a card holder and with a merchant at a certain date: this information is only used to build the tripartite graph.

Variable name	**Description**
inBEL/EURO/OTH	Issuing region: Belgium/Europa/World
TX AMOUNT	Amount of transaction
TX 3D SECURE	Transaction used 3D secure
AGE	Age of card holder
langNED/FRE/OTH	Card holder language: Dutch/French/Other
isMAL/FEM	Card holder is Male/Female
isFoM	Card holder gender unknown
BROKER	Code of card provider
cardMCD/VIS/OTH	Card is a Mastercard/Visa/Other
01 Mer score	Merchant risk score (no time damping)
ST/MT/LT Mer score	Day/week/month decay merchant risk score (3 features)
01 CH score	Card Holder risk score (no time damping)
ST/MT/LT CH score	Day/week/month decay Card Holder risk score (3 features)
01 Trx score	Transaction risk score (no time damping)
ST/MT/LT Trx score	Day/week/month decay Transaction risk score (3 features)
TX FRAUD	Target variable: Fraud/Guenuine

For the particular node $\text{Trx}^j_{i,k}$, Equation (8.5) then writes:

$$\text{sc}(\text{Trx}^j_{i,k}) = \alpha \left(\frac{1}{\sum_{j=1}^{n} p_{ji} + 1} \text{sc}(\text{Mer}_i) + \frac{1}{\sum_{j=1}^{m} p_{jk} + 1} \text{sc}(CH_k) \right) + (1-\alpha)\text{sc}(\text{Trx}^j_{i,k}) \tag{8.6}$$

which leads to Equation (8.5) when α disappears. Notice that if a new transaction involves a new merchant and/or card holder, $\text{sc}(\text{Mer}_i)$ and/or $\text{sc}(\text{CH}_k)$ are also unkown. When this occurs, $\text{sc}(\text{Mer}_i)$ and/or $\text{sc}(\text{CH}_k)$ are set to zero accordingly.

Finally, those 12 new features (plus transaction-related features, see Table 8.1) are fed into a random forest classification model, as this model proved to perform well for the problem at hand, predicting fraudulent transaction [37, 71].

8.2 The Proposed Model

While showing good performance, APATE can be improved in various ways.

8.2.1 Dealing with hubs

From the literature, it is known that presence of hubs in a network can harm the classifiers [198, 199, 114]: **hubs** are nodes having a high degree and are therefore neighbors of a large number of nodes. In our dataset, it corresponds to popular nodes such as popular online shops like Amazon (as an example, the dataset is anonymised). Those nodes tend to accumulate a high value of risk score since they are connected to a lot of transactions. A simple way to counterbalance this accumulation is to divide the risk score by the node degree after convergence. In general, it is possible to divide by any power of the node degree and/or by different powers for the three types of nodes of the tripartite graph (transactions, card holders, and merchants). In practice however, we did not find any combination that significantly beats the simple divide-by-node-degree option (results are not reported here).

Furthermore, it allows us to make a link with the **regularized commute time kernel** which is $\mathbf{K}_{\mathrm{RCT}} = (\mathbf{D} - \alpha\mathbf{A})^{-1}$ (where $\mathbf{D}$ is the degree matrix) : element i, j of this kernel can be interpreted as the discounted cumulated probability of visiting node j when starting from node i (see [265, 91, 168] for details). The (scalar) parameter $\alpha \in]0, 1]$ corresponds to an evaporating or killed random walk where the random walker has a $(1 - \alpha)$ probability of disappearing at each step (therefore it has a similar interpretation as for the RWWR used in APATE, see Section 8.1). It can be shown that it corresponds to the score obtained by a random walk with restart divided by the degree of the nodes, for undirected graphs [90]. This method provided the best results in a recent comparative study on semi-supervised classification [91] and the second best results in the Chapter 5 of this thesis. In practice, the efficient implementation proposed in [168], Equation (22), for semi-supervised classification with the Regularized Commute Time Kernel is used and referred as RCTK.

Equation (8.5) must be divided by two when using the RCTK, as the sum of degree is two for a new transaction (one new edge to the merchant and one new edge to the card holder, by construction).

8.2.2 Introducing a time gap

On the other hand, unlike in [241], the model cannot be based on past few days. Indeed, fraudulent transaction tags (the variable we want to predict) cannot be known with certainty without the human investigators feedback. Moreover, since the fraudsters' modus operandi is known to change over time, it is not acceptable to built our model on old, less reliable (but fully inspected) data. However, it takes several days to inspect all transactions, mainly because it is sometime the card holders that report undetected frauds. Of course, this makes our fraud detection problem harder [70].

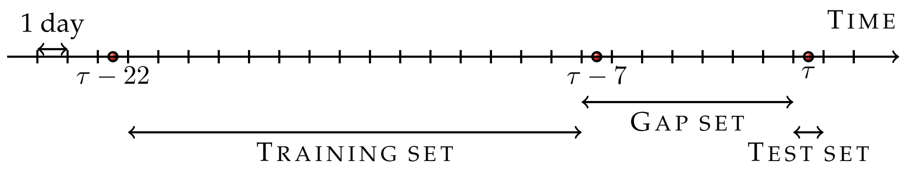


FIGURE 8.4: Real-life FDS scenario with three sets of data. It takes several days to inspect all transactions, mainly because it is sometime the card holder who reports undetected frauds. Hence, in practice, the fraud tags of the Gap set are unknown. This scenario is repeated each day, as the parameter τ is incremented.

In arrangement with field experts, we designed a real-life scenario containing three sets of data:

1. **Training set**: data where the transaction fraud labels can be taken as reliable.
2. **Gap set**: data where the transaction fraud labels are unknown.
3. **Test set**: data of the day on which the algorithm is currently tested.

It corresponds therefore to a **semi-supervised learning scheme** (SSL, see Chapter 3), as training data are partially labeled. If the Gap set is totally left aside, this is an usual **supervised learning** (SL) problem again. Both cases (SL and SSL) were investigated:

- For the SL scheme, only the training set is used to build the graph, and only the Training set is used to train the random forest.
- For the SSL scheme, the training set and the gap set are used to build the graph, and only the Training is used to train the random forest classifier.

Once again, in arrangement with field experts, 15 days of training data and seven days for the gap set were chosen [44, 72]. This scenario is depicted on Fig. 8.4. Notice that on this figure, τ controls the testing day and that models are systematically built (overnight) on the 22 previous days. By changing τ, we get different testing days.

8.2.3 Including investigators feedback

Remember from the introduction that a fraction of previous alerts have been confirmed or overturned by human investigators (typically when a fraud alert occurs, the card is blocked and the card holder is contacted by phone). In our case, we put this number of **feedbacks** per day to 100, in arrangement with field experts. It is a realistic average number of cards that human investigators can check per day, usually by contacting the card holder. It means that, each day, the 100 most probable fraudulent cards (according to the model) are checked and then used as feedback. Therefore, in each of our gap set (except in starting conditions) 100 cards per gap set day have been checked by human investigators. We will take advantage of these investigated cases in order to try to predict more accurately the fraudulent transactions. On average, it corresponds roughly to 1400 transaction feedbacks (two transactions per card) from previous testing day (previous τ's of our model). This option will be referred as +FB and only makes sense in a SSL scheme.

8.2.4 Minors improvements

This variant brings small numeric (as opposed to theoretical) improvements. First it uses priors for missing merchant and card holder scores instead of zero in Equation (8.4). The prior is computed from previous day.

We also balance Equation (8.1) by introducing a balance factor P (a positive scalar), because nothing ensures that transaction-merchant and transaction-card holder edges should be equally weighted:

$$\mathbf{A}^{\text{tri}} = \begin{pmatrix} \mathbf{0}_{t\times t} & \mathbf{A}_{t\times c} & P\mathbf{A}_{t\times m} \\ \mathbf{A}_{c\times t} & \mathbf{0}_{c\times c} & \mathbf{0}_{c\times m} \\ P\mathbf{A}_{m\times t} & \mathbf{0}_{m\times c} & \mathbf{0}_{m\times m} \end{pmatrix} \tag{8.7}$$

This variant was only applied to the best model, namely RCTK SSL +FB see 8.4, and is refered as RCTK SSL +FB P=32, as the best tested value for the parameter P is 32. A plot aiming to optimize this parameter can be found on Figure 8.5.

8.3 Experimental comparisons

In this section, the possible variants of the considered algorithms are compared on a real-life e-commerce credit card transaction data set obtained from a large credit card issuer in Belgium. Those graph-based algorithms compute additional features and were presented in Section 8.1 and 8.2. For practical purposes, considered algorithms are recalled in Table 8.2 and the classifier is always a random forest with 400 trees. Random forest is an ensemble of decision trees (also called bagging), where each tree is trained on different bootstrap sample of the original training set and uses a random features subset [70]. The

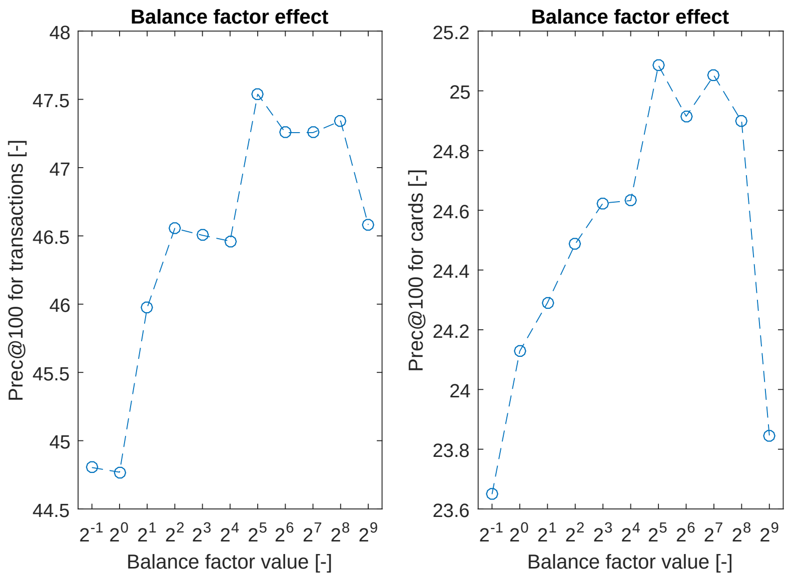


FIGURE 8.5: This plot aims optimizing the balance between transaction-merchant and transaction-card holder edges. The x-axis is logarithmic (base 2). The best balance factor is 32 for both card-based and transaction-based Pr@100.

bootstrap sampling also allows to reweigh the two classes, enhancing the predictions in the case of unbalanced dataset. Diversity in the generated trees is a key factor for good performance with unbalanced data.

The database is composed of 25,445,744 transactions divided in 139 days and fraud ratio is 0.31%. The features list can be found in Table 8.1. From this table, the first group contains mostly socio-demographic features which are taken as-is. The second group contains the graph-based features described in Section 8.1 and 8.2. Notice that each transaction is linked with a card holder and with a merchant at a certain date. Those three pieces of information (card holder, merchant, and date) are used to build the tripartite graph. Finally, this database does not focus on a certain type of card fraud (stolen, card-not-present,...) but contains all reported fraudulent transactions in this time period.

As a performance indicator, Precision@100 (Pr@100) [232] was chosen, in accordance with field experts. It means that the 100 most probable (according to models) fraudulent transactions are checked by human investigators each day (and added as feedback in RWWR SSL +FB and RCTK w/ SSL +FB). Similarly all most probable fraudulent transactions are considered until 100 cards have

TABLE 8.2: Result obtained by the eigth compared models, see Sections 8.1 and 8.2 for acronyms. Considered variations of the APATE Algorithm are tested according to four dimensions: hubs status, learning scheme, feedback status and minor improvements. Precision@100 (see Section 8.3) both for fraudulent card and transaction predictions is also reported (formatted mean ± std). Original features is the baseline with no additional graph features used.

Classifier name	**Risk scores**	**Damp hubs**	**Learning**	**Feedback**	**Card Pr@100**	**Trx Pr@100**
Original features	not used	no	Supervised	no	2.38 ± 2.16	3.91 ± 5.54
RWWR SL = APATE	used	no	Supervised	no	18.05 ± 9.34	35.16 ± 19.07
RWWR SSL	used	no	Semi-supervised	no	14.67 ± 6.92	28.58 ± 14.32
RWWR SSL +FB	used	no	Semi-supervised	yes	17.14 ± 8.40	33.28 ± 15.41
RCTK SL	used	yes	Supervised	no	20.54 ± 10.54	40.37 ± 20.92
RCTK SSL	used	yes	Semi-supervised	no	19.81 ± 8.67	37.59 ± 16.81
RCTK SSL +FB	used	yes	Semi-supervised	yes	23.79 ± 9.63	46.28 ± 17.03
RCTK SSL +FB P=32	used	yes	Semi-supervised	yes	**25.09** ± **9.99**	**47.54** ± **17.26**

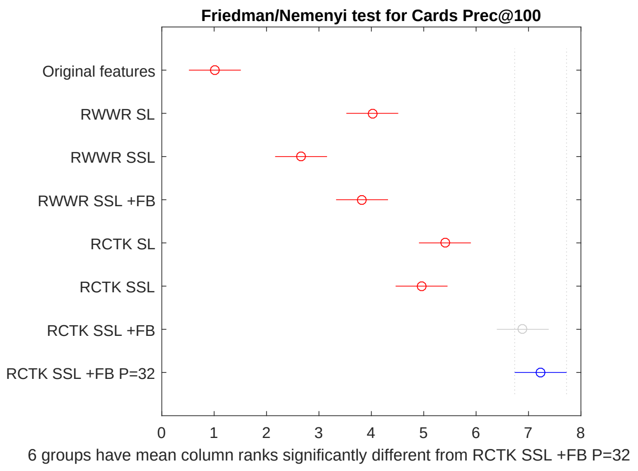


FIGURE 8.6: Mean rank (circles) and critical difference (plain line) of the Friedman/Nemenyi test, obtained on a three-months real-life e-commerce credit card transaction data set. The blue method has the best mean rank and is significantly better than red methods. Critical difference is 0.99. Performance metric is Pr@100 on fraudulent card prediction.

been checked as usually human investigators verify all transactions of a card when they investigate. Precision@100 reports the number of true fraudulent

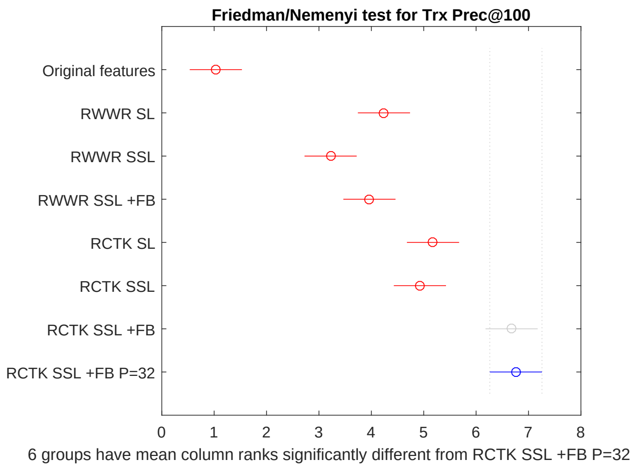


FIGURE 8.7: Friedman/Nemenyi test, obtained on a three-months real-life e-commerce credit card transaction data set. Here, performance metric is Pr@100 on transaction prediction. See Figure 8.6 for details.

transactions or cards among 100 investigated. Notice that card-based precision is more realistic as it is somehow the normal work charge for a human investigators team.

Notice that the data set is build a posteriori, it is therefore possible that a fraud in the test set was previously identified in the train set. In that case, the card would have been blocked at that time, and the transaction in the test set should therefore never occur. In order to suppress this effect, we did not took into account those cases in the computation of Pr@100.

8.4 Results and discussion

Figure 8.6 and 8.7 compare methods described on Table 8.2 through a Friedman/Nemenyi test [74] for cards-based Pr@100 and transactions-based Pr@100, respectively (see Appendix A for a description). To that end, we adopt a sliding window approach: each day (different τ from Fig 8.4) is considered as a different (train-gap-test) dataset. Friedman null hypothesis is rejected with $\alpha = 0.05$ and Nemenyi critical difference is equal to 0.99. A method is considered as significantly better than another if its mean rank is larger by more than this amount.

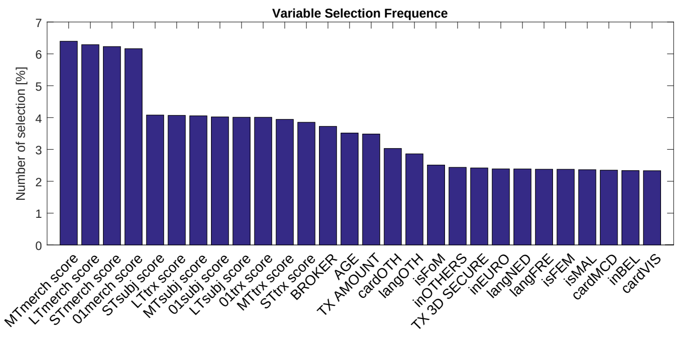


FIGURE 8.8: Variables selected by the random forests for the RCTK SSL +FB model for all days. Our risk features are the most selected features.

Firstly, RCTK always beats RWWR. This superiority indicates that tackling the hubs problem is actually important, as the difference is significant (see Table 8.2). Secondly, SSL (opposed to SL) does not lead to an improvement, but it allows to use the feedback. Thirdly, adding the feedback +FB significantly enhances the performance for RCTK, but not for RWWR. Intuitively, the effect of feedback should always be positive. In practice, however, a certain number of feedbacks is required to improve the performance. This number can be higher than the realistic number of feedback that we allow per day.

Overall, the best combination is RCTK SSL +FB P=32, reaching 25% Pr@100 for fraudulent card detection and nearly 50% in terms of fraudulent transaction. From the original APATE algorithms, we therefore increased the Pr@100 for fraudulent card detection by 39%.

It can be observed that classification based only on the original features provides poor performance. Nevertheless, this does not mean that those features are information-less. Indeed, [70] and [22] used feature engineering to obtain interesting results (from the same data set in the case of [70]).

Figure 8.8 indicates the frequency of selected features by the random forest classifier. The method is RCTK SSL +FB and selects merchant risk scores most often. Then comes the transaction and card user risk scores, followed by features from the original feature set.

Finally, we conclude this section by a time complexity analysis.

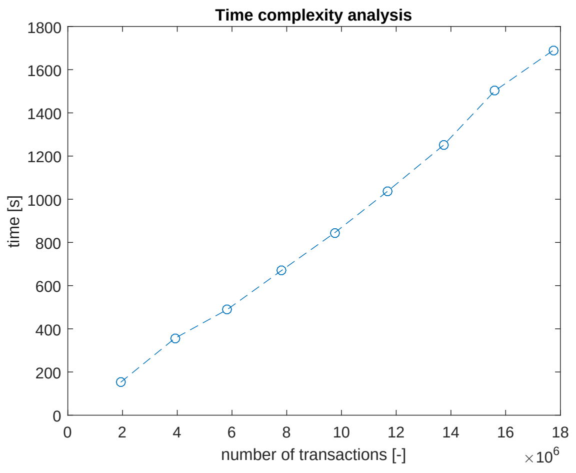


FIGURE 8.9: Required time to build the tripartite graph and extract the 12 risk scores. The corresponding complexity is $O(n)$ with n the number of nodes or transactions.

8.4.1 Time complexity

One of our main objectives through this chapter was to design a scalable fraud detection system. We therefore restrict ourselves to light sparse operations. Apart from the classifier itself, the bottleneck is a matrix-vector sparse multiplication (see Equation (8.2)).

The corresponding time-complexity is $O(n_{\text{Trx}})$ in our case with n_{Trx} being the number of nodes or transactions. Indeed, the total number of node (of three types) and the number of transactions are linearly related. Experimentally, the time required to build the tripartite graph and extract the 12 risk scores versus both transactions and total number of node almost follow a regression line. This relation is reported for time versus transactions on Figure 8.9. Finally, the space complexity is also $\mathcal{O}(n_{\text{Trx}})$.

8.5 Conclusion

In this chapter, we start from an existing fraud detection systems (FDS) APATE and bring several improvements which have a large impact on performances:

We first damp the hurtful effect of hub nodes, using the regularized commute time kernel (RCTK). We then take into account real-life constrains such as the Gap set (introducing semi-supervised learning learning, SSL) and using

Pr@100 as a metric. Feedback information from human investigators is also introduced (+FB).

The resulting fraud detection system (FDS) is tested on a three-months real-life e-commerce credit card transactions data set obtained from a large credit card issuer in Belgium. Our extensions improve significantly the Pr@100, both on fraudulent cards or transactions prediction (for acronyms, see Section 8.2).

The scalability of this method is also demonstrated, as the time-complexity to extract the risk scores is $O(n)$. It allows to tackles millions of transactions in minutes on a standard laptop.

An envisaged further work is to introduce semi-supervised learning not only at the level of graph analysis but also for the main classifier.

Chapter 9

Constrained randomized shortest path problems

The present chapter aims to study randomized shortest path problems with **equality constraints** on the transition probabilities from a subset of nodes, in the context of a single source and a single destination. This allows to set a priori some transition probabilities and then finding the optimal policy compatible with these probabilities. As shown in Section 9.2, the randomized shortest path framework is really close to the bag-of-paths framework.

It therefore extends previous work dedicated to randomized shortest paths (RSP, [208, 257, 144]), and initially inspired by stochastic traffic assignment studied in transportation science [7]. The model can be described informally as follows. Suppose we have to find the optimal policy, minimizing expected cost for reaching a destination node from a source node in a network, where costs are associated to local decisions/actions. Usually, deterministic and stochastic shortest path algorithms provide pure deterministic policies: when standing in a given state k, we just choose the best action u leading to the minimal expected cost.

In the present work, we investigate the possibility of optimally **randomizing** the policy while fixing a subset of transition probabilities: the agents can choose several actions, according to some probability distribution. Randomizing the policy thus introduces a continual exploration of the network. As in the standard RSP framework, the degree of randomness can be controlled by a **temperature parameter** allowing to interpolate between the least-cost solution given by the optimal shortest path algorithm and a random behavior provided by a predefined, reference, random policy.

In practice, randomization corresponds to the association of a probability distribution on the set of admissible actions in each node ([208], choice randomization or mixed strategy). If no randomization is present, only the best policy is exploited. Randomization appears when this probability distribution is no more peaked on the best choice: the agent is willing to sacrifice efficiency for exploration. Note that randomized choices are common in a variety of fields [208]; for instance game theory (mixed strategies; see for instance [186]), computer sciences [178], Markov games [160], decision sciences [200], ...

A comprehensive related work and a detailed discussion of the reasons for randomizing the policy can be found in [208], and is summarized here:

- If the environment is changing over time (non-stationary), the system could benefit from randomization by performing continual exploration.
- It is sometimes of interest to explore the environment, such as in reinforcement learning.
- A deterministic policy would lead to a totally predictable behavior; on the contrary, randomness introduces unpredictability and therefore renders interception more difficult. Randomization (mixed strategies) has proven useful for exactly this reason in game theory.
- A randomized policy spreads the traffic over multiple paths, therefore reducing the danger of congestion.
- In some applications, like in social networks analysis, computing a distance accounting for all paths – and thus integrating the concept of high connectivity – could provide better results than relying on the optimal, shortest, paths only.
- In computer gaming, it is often desirable to be able to adapt the strength of the digital opponent, which can easily be done with the introduced model.

In this context, the randomness associated to paths connecting the initial node and the goal node is quantified by the relative entropy, or Kullback-Leibler divergence (see, e.g., [67]), between the probability distribution on the paths and their likelihood according to a reference random walk on the graph, usually a uniform distribution on the set of available actions. This relative entropy captures the degree of randomness of the system.

The optimal, randomized, policy (assigning a probability distribution on the set of actions available on each state) is then obtained by minimizing the free energy (the expected cost plus the Kullback-Leibler divergence), subject to some equality constraints on the transition probabilities provided by the environment – the probabilities of jumping to a given state after having chosen an action – which have to be verified exactly. Based on this formalism, a first, generic, algorithm for solving constrained randomized shortest paths problems is developed by exploiting Lagrange duality.

Then, as an application example, this framework is used in order to solve Markov decision problems for providing mixed, randomized, policies.

Markov decision problems [194, 197, 224, 225, 235], also called **stochastic shortest path problems** [33, 34], are currently used in a wide range of application areas including transportation networks, medical imaging, wide-area network routing, artificial intelligence, to name a few (see, e.g., [197, 224, 254, 255, 256]).

A simple, easy-to-implement, **soft value iteration** algorithm solving the problem is derived and its convergence to a fixed point is proved. Interestingly, this "soft" value iteration algorithm is closely related to dynamic policy programming [18] as well as Kullback-Leibler and path integral control [237, 207, 137, 234, 233], and similar to the exploration strategy recently introduced in [13, 14].

This therefore shows that the recently proposed exploration strategy in [13, 14], developed in parallel to this work, is globally optimal in our setting in the following sense: it minimizes expected cost subject to constant relative entropy of paths probabilities when the goal state is absorbing and reachable from any other state. Interestingly, as in [95, 94], the softmax value iteration algorithm extends the Bellman-Ford value iteration algorithm by simply replacing the minimum operator by a soft minimum operator.

Note that still another way of solving the problem was developed in [31], but this algorithm is not included here because it is less generic. For a comprehensive related work about randomized policies, see [208, 3].

In brief, this work has four contribution:

- It extends randomized shortest paths to problems with constrained transition probabilities on a subset of nodes.
- The constrained randomized shortest paths framework is applied to standard Markov decision problems in order to provide optimal randomized policies.
- A simple, easy-to-implement, soft value iteration algorithm is derived for obtaining optimal randomized policies.
- Simulations on concrete problems show that the algorithms behave as expected.

This chapter is organized as follows: Before going further, we start with a small toy example in Section 9.1. Then, Section 9.2 introduces the randomized shortest paths framework. Section 9.3 considers randomized shortest path problems with constraints on transition probabilities. In Section 9.4, the standard Markov decision problem is recast as a constrained randomized shortest path problem on a bipartite graph and an algorithm is proposed for solving it. Moreover a simple more efficient soft value iteration algorithm is also developed. Section 9.5 shows some simulation examples and Section 9.6 is the conclusion.

9.1 Constrained randomized shortest path example

Let us consider a car driver leaving his house and going to his workplace. His GPS can guide him using the shortest path between those two places.

Tired of wasting his time in traffic jams, or because he wants to explore his city a bit further, he decides to drive his way using a non-deterministic policy. At each crossroad, he can choose to go left, to go right, or to continue ahead (each crossroad having its specificity that can be random because of traffic lights, detour roads and traffic cops). He will never take highways, so that the number of crossroads in his neighborhood is finite.

The idea behind the constrained randomized shortest path is to add more and more randomization (using the temperature parameter) in a path between some fixed start and end points. In our example (and in Markov decision processes, as we will see), a part of transitions are constrained by the environment (here, each crossroad leads to somewhere else), and others are unconstrained (here, the driver can choose to go left, right or ahead).

The randomization will therefore occur on unconstrained choices, and, once the targeted level of randomness is reached, those choices (here, the sequence of left, right or ahead) will be used as the optimized, randomized, policy.

9.2 The randomized shortest path framework

Our formulation of the problem is based on the **randomized shortest path** (RSP) framework (see Section 9.2). It defines a dissimilarity measure interpolating between the shortest path distance and the commute-time distance in a graph [257, 208, 144]. This formalism is based on full paths instead of standard "local" flows [6].

We start by providing the necessary background and notation in Section 9.2.1. Then, we proceed with a short summary of the randomized shortest path formalism in Section 9.2.2, before introducing, in Section 9.3, randomized shortest paths with constraints on the transition probabilities.

9.2.1 Some background and notation

We consider a weighted directed graph or network, G, with a set of n nodes $\mathcal{V}$ (or vertices) and a set of arcs $\mathcal{E}$ (or edges). The graph is represented by its $n \times n$ adjacency matrix $\mathbf{A}$ containing binary values if the graph is unweighted or non-negative affinities between nodes in the case of a weighted graph. To each edge linking node i to node j, we also associate a non-negative number

c_{ij} representing the immediate cost of following this edge. The costs should be non-negative and are gathered in matrix **C**.

Let us also recall the **reference random walk** on G, which was defined in Chapter 2. The choice to follow an edge from node i is made according to a probability distribution (transition probabilities) defined on the set $\mathcal{S}ucc(i)$ of successor nodes of i. These transition probabilities, defined on each node i, will be denoted as

$$p_{ij}^{\text{ref}} = p^{\text{ref}}(s(t+1) = j | s(t) = i) = \frac{a_{ij}}{\sum_{k \in \mathcal{S}ucc(i)} a_{ik}} \tag{9.1}$$

where a_{ij} is element i, j of the adjacency matrix and $s(t)$ is a random variable containing the node visited by the random walker at time t. Furthermore, $\mathbf{P}_{\text{ref}}$ will be the matrix containing the transition probabilities p_{ij}^{ref} as elements. For consistency, if there is no edge between i and j, we consider that c_{ij} takes a large value, denoted by ∞; in this case, the corresponding transition probability is equal to zero, $p_{ij} = 0$.

Finally, in this work, we consider that there is a unique absorbing, killing, node which is the last node n – the goal node. Any transition from this node is forbidden, that is, $p_{nj}^{\text{ref}} = 0$ for all j.

9.2.2 The standard randomized shortest path formalism

The main idea is the following. We consider the set of all **hitting** paths, or walks, $\wp \in \mathcal{P}$ from node 1 to the (unique) absorbing, or hitting, node n on G. We further assume than this absorbing node n can be reached in a finite number of steps from each other node in the graph. Each path $\wp$ consists in a sequence of connected nodes starting in node 1 and ending in n. Then, we assign a probability distribution $\mathrm{P}(\cdot)$ on the set of paths $\mathcal{P}$ by minimizing the generalized **free energy**[1] of statistical physics [130, 190, 202],

$$\left| \begin{array}{ll} \underset{\{\mathrm{P}(\wp)\}_{\wp \in \mathcal{P}}}{\text{minimize}} & \phi(\mathrm{P}) = \sum_{\wp \in \mathcal{P}} \mathrm{P}(\wp) \tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}} \mathrm{P}(\wp) \log \left(\dfrac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)} \right) \\ \text{subject to} & \sum_{\wp \in \mathcal{P}} \mathrm{P}(\wp) = 1 \end{array} \right. \tag{9.2}$$

where $\tilde{c}(\wp) = \sum_{\tau=1}^{t} c_{s(\tau-1)s(\tau)}$ is the total cumulated cost along path $\wp$ when visiting the sequence of nodes, or states, $(s(\tau))_{\tau=0}^{t}$ and t is the length of path

[1]Alternatively, we can adopt a maximum entropy point of view [126, 131]. Moreover, the free energy could also be defined as $\phi(\mathrm{P}) = \sum_{\wp \in \mathcal{P}} \mathrm{P}(\wp)(\tilde{c}(\wp) - c^*) + T \sum_{\wp \in \mathcal{P}} \mathrm{P}(\wp) \log \left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)} \right)$ where c^* is the shortest path cost from starting node 1 to destination node n. In this case, costs are computed relatively to the shortest path cost. This choice leads exactly to the same probability distribution over paths (Equation (9.3)).

$\wp$. Furthermore, $\tilde{\pi}(\wp) = \prod_{\tau=1}^{t} p_{s(\tau-1)s(\tau)}^{\mathrm{ref}}$ is the product of the transition probabilities (see Equation (9.1)) along path $\wp$ – the likelihood of path $\wp$. It defines a **reference** probability distribution over paths as $\sum_{\wp \in \mathcal{P}} \tilde{\pi}(\wp) = 1$ [95, 94].

The objective function in Equation (9.2) is a mixture of two dissimilarity terms with the temperature T balancing the trade-off between them. The first term is the expected cost for reaching destination node from source node (favoring shorter paths – **exploitation**). The second term corresponds to the relative entropy, or Kullback-Leibler divergence, between the path probability distribution and the path likelihood distribution (introducing randomness – **exploration**). When the temperature T is low, shorter paths are favored while when T is large, paths are chosen according to their likelihood in the random walk on the graph G (i.e. the product of the reference transition probabilities). Note that we should normally add non-negativity constraints on the path probabilities, but this is not necessary as the resulting quantities s automatically non-negative (the relative entropy forbids negative values). Note that, instead of minimizing free energy, it is equivalent to minimize expected cost subject to a fixed relative entropy constraint [95, 90, 94].

This argument, akin to maximum entropy [126, 131, 130, 138], leads to a **Gibbs- Boltzmann distribution** on the set of paths (see, e.g., [95, 94] for a detailed derivation),

$$\mathrm{P}^*(\wp) = \frac{\tilde{\pi}(\wp)\exp[-\theta\tilde{c}(\wp)]}{\sum\limits_{\wp' \in \mathcal{P}} \tilde{\pi}(\wp')\exp[-\theta\tilde{c}(\wp')]} = \frac{\tilde{\pi}(\wp)\exp[-\theta\tilde{c}(\wp)]}{\mathcal{Z}} \tag{9.3}$$

where $\theta = 1/T$ is the inverse of the temperature and the denominator $\mathcal{Z} = \sum_{\wp \in \mathcal{P}} \tilde{\pi}(\wp)\exp[-\theta\tilde{c}(\wp)]$ is the **partition function** of the system. This defines the **optimal mixed policy** in terms of **probabilities of choosing a path**, $\mathrm{P}^*(\wp)$. It can be shown that this set of path probabilities is exactly equivalent to the ones generated by a Markov chain with biased transition probabilities favoring shorter paths, depending on the temperature T [208].

9.2.3 The minimum free energy

Interestingly, if we replace the probability distribution $\mathrm{P}(\cdot)$ by the optimal distribution $\mathrm{P}^*(\cdot)$ provided by Equation (9.3) in the objective function (9.2), we obtain for the minimum **free energy** between node 1 and node n,

$$\begin{aligned}\phi_1^*(T) = \phi(\mathrm{P}^*) &= \sum_{\wp \in \mathcal{P}} \mathrm{P}^*(\wp)\tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}} \mathrm{P}^*(\wp) \log \left(\frac{\mathrm{P}^*(\wp)}{\tilde{\pi}(\wp)} \right) \\ &= \sum_{\wp \in \mathcal{P}} \mathrm{P}^*(\wp)\tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}} \mathrm{P}^*(\wp) \left(-\frac{1}{T}\tilde{c}(\wp) - \log \mathcal{Z} \right) \\ &= -T \log \mathcal{Z} \end{aligned} \tag{9.4}$$

9.2.4 Computing interesting quantities from the partition function

Moreover, several quantities of interest can be computed by taking the partial derivative of the optimal free energy (9.4) [257, 208, 144, 95, 94, 90].

Expected number of visits to edges and nodes. For instance, from Equation (9.4), for the expected number of passages through edge (i, j) at temperature $T = 1/\theta$, that is, the flow in (i, j),

$$\begin{aligned}\frac{\partial \phi(\mathrm{P}^*)}{\partial c_{ij}} &= -\frac{1}{\theta \mathcal{Z}} \frac{\partial \mathcal{Z}}{\partial c_{ij}} = -\frac{1}{\theta \mathcal{Z}} \sum_{\wp \in \mathcal{P}} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)](-\theta) \frac{\partial \tilde{c}(\wp)}{\partial c_{ij}} \\ &= \sum_{\wp \in \mathcal{P}} \frac{\tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\mathcal{Z}} \frac{\partial \tilde{c}(\wp)}{\partial c_{ij}} \\ &= \sum_{\wp \in \mathcal{P}} \mathrm{P}^*(\wp)\, \eta\big((i,j) \in \wp\big) \\ &= \bar{n}_{ij}(T) \end{aligned} \tag{9.5}$$

where we used $\partial \tilde{c}(\wp)/\partial c_{ij} = \eta\big((i,j) \in \wp\big)$, with $\eta\big((i,j) \in \wp\big)$ being the number of times edge (i, j) appears on path $\wp$ at temperature T. Therefore, we have for the flow in (i, j),

$$\bar{n}_{ij}(T) = -T \frac{\partial \log \mathcal{Z}}{\partial c_{ij}} \tag{9.6}$$

Now, it has already been shown that the partition function can be computed in closed form (see, e.g., [208, 145, 90] for details). Let us first introduce the **fundamental matrix** of the RSP system,

$$\mathbf{Z} = \mathbf{I} + \mathbf{W} + \mathbf{W}^2 + \cdots = (\mathbf{I} - \mathbf{W})^{-1}, \quad \text{with } \mathbf{W} = \mathbf{P}_{\text{ref}} \circ \exp[-\theta \mathbf{C}] \tag{9.7}$$

where $\mathbf{C}$ is the cost matrix and $\circ$ is the elementwise (Hadamard) product. Elementwise, the entries of the $\mathbf{W}$ matrix are $w_{ij} = [\mathbf{W}]_{ij} = p_{ij}^{\text{ref}} \exp[-\theta c_{ij}]$.

Note that this matrix is sub-stochastic because the costs are non-negative and node n is absorbing and killing (row n contains only 0 values).

Then, the partition function is simply $\mathcal{Z} = [\mathbf{Z}]_{1n} = z_{1n}$ (see [257, 208, 144, 95, 94]). More generally [101], we find that

$$z_{1k} = \sum_{\wp \in \mathcal{P}_{1k}} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)] \quad \text{and} \quad z_{kn} = \sum_{\wp \in \mathcal{P}_{kn}} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)] \tag{9.8}$$

with $z_{nn} = 1$ and where $\mathcal{P}_{1k}$ is the set of hitting paths starting in node 1 and ending in node k. Symmetrically, $\mathcal{P}_{kn}$ is the set of hitting paths starting in node k and ending in node n. The z_{1k} quantities are called the **forward variables** whereas the z_{kn} are the **backward variables**. These variables can be interpreted as probabilities of surviving during the killed random walk (see, e.g., [95, 94] for details).

Moreover, the flow in (i, j) can be obtained from (9.7) by

$$\bar{n}_{ij}(T) = -\frac{1}{\theta} \frac{\partial \log \mathcal{Z}}{\partial c_{ij}} = \frac{z_{1i} p_{ij}^{\text{ref}} \exp[-\theta c_{ij}] z_{jn}}{z_{1n}} = \frac{z_{1i} w_{ij} z_{jn}}{z_{1n}} \tag{9.9}$$

and because only the first row and the last column of $\mathbf{Z}$ are needed, two systems of linear equations can be solved instead of matrix inversion in Equation (9.7).

From the last equation and $z_{in} = \sum_{j=1}^{n} w_{ij} z_{jn} + \delta_{in}$ (the elementwise form of $(\mathbf{I} - \mathbf{W})\mathbf{Z} = \mathbf{I}$), the expected number of visits to a node j can be computed from

$$\bar{n}_i(T) = \sum_{j=1}^{n} \bar{n}_{ij}(T) + \delta_{in} = \frac{z_{1i} z_{in}}{z_{1n}} \quad \text{for } i \neq n \tag{9.10}$$

where we assume $i \neq n$ for the last equality because we already know that $\bar{n}_n(T) = 1$ at the ending node, which is absorbing and killing.

Note that it can easily be shown that the relative entropy of the paths in Equation (9.2) can be rewritten in function of the expected number of visits and the **local relative entropy** as $\sum_{i=1}^{n-1} \bar{n}_i(T) \sum_{j \in \mathcal{S}ucc(i)} p_{ij} \log \frac{p_{ij}^*}{p_{ij}^{\text{ref}}}$ where the p_{ij}^* are the transition probabilities defining the current local policy of the random walker (probability of following an edge) [208]. In fact, all the quantities of interest can be computed from the expected number of visits (see [101], Equation (11)).

Randomized mixed policy. Furthermore, the optimal transition probabilities of following an edge (i, j) with $i \neq n$ are

$$p^*_{ij}(T) = \frac{\bar{n}_{ij}(T)}{\bar{n}_i(T)} = p^{\text{ref}}_{ij} \exp[-\theta c_{ij}] \frac{z_{jn}}{z_{in}} = \frac{w_{ij} z_{jn}}{z_{in}} = \frac{w_{ij} z_{jn}}{\sum_{j'=\mathcal{S}ucc(i)} w_{ij'} z_{j'n}} \quad (9.11)$$

as $p^{\text{ref}}_{ij} \exp[-\theta c_{ij}] = w_{ij}$ and $z_{in} = \sum_{j=\mathcal{S}ucc(i)} w_{ij} z_{jn}$ for all $i \neq n$ (the elementwise form of $(\mathbf{I} - \mathbf{W})\mathbf{Z} = \mathbf{I}$, coming from Equation (9.7)). This expression defines a biased random walk on G – the random walker is "attracted" by the destination node n. These transition probabilities do not depend on the source node and correspond to the optimal randomized strategy, or policy, minimizing free energy. This randomized policy is also called a **mixed policy** as in game theory (see, e.g., [186]).

Expected cost until destination. In addition, the expected cost until reaching hitting node n is [208, 145, 90]

$$\langle \tilde{c} \rangle = \sum_{\wp \in \mathcal{P}} \mathrm{P}^*(\wp) \tilde{c}(\wp) = \sum_{\wp \in \mathcal{P}} \frac{\tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\mathcal{Z}} \tilde{c}(\wp) \quad (9.12)$$

By defining the matrix containing the expected number of passages through the edges by $\mathbf{N}$ with $[\mathbf{N}]_{ij} = \bar{n}_{ij}(T)$, the expected cost spread in the network is

$$\langle \tilde{c} \rangle = -\frac{\partial \log \mathcal{Z}}{\partial \theta} = \mathbf{e}^{\mathrm{T}} (\mathbf{N} \circ \mathbf{C}) \mathbf{e} \quad (9.13)$$

where $\mathbf{e}$ is a column vector of 1s. This is just the cumulative sum of the expected number of passages through each edge times the cost of following the edge, $\sum_{i=1}^{n-1} \sum_{j \in \mathcal{S}ucc(i)} \bar{n}_{ij}(T) c_{ij}$ [101].

Entropy of the paths. In Equation (9.2), the relative entropy of the set of paths was defined as

$$J(\mathrm{P}^* | \tilde{\pi}) = \sum_{\wp \in \mathcal{P}} \mathrm{P}^*(\wp) \log \left(\frac{\mathrm{P}^*(\wp)}{\tilde{\pi}(\wp)} \right) \quad (9.14)$$

and, from Equations (9.2) and (9.4), it can be computed thanks to

$$J(\mathrm{P}^* | \tilde{\pi}) = -(\log \mathcal{Z} + \tfrac{1}{T} \langle \tilde{c} \rangle) \quad (9.15)$$

where the partition function $\mathcal{Z} = [\mathbf{Z}]_{1n} = z_{1n}$.

Free energy distance. It was already shown that the minimal free energy (9.4) at temperature T is provided by $\phi_1^*(T) = \phi(\mathrm{P}^*) = -T \log \mathcal{Z} = -\frac{1}{\theta} \log z_{1n}$. In [95, 94], it was proved that the free energy from any starting node i to absorbing node n, $\phi_i^*(T) = -\frac{1}{\theta} \log z_{in}$, can be computed thanks to the following recurrence formula to be iterated until convergence

$$\phi_i^*(T) = -\tfrac{1}{\theta} \log z_{in} = \begin{cases} -\frac{1}{\theta} \log \left[\displaystyle\sum_{j \in \mathcal{S}ucc(i)} p_{ij}^{\mathrm{ref}} \exp[-\theta(c_{ij} + \phi_j^*(T))] \right] & \text{if } i \neq n \\ 0 & \text{if } i = n \end{cases} \tag{9.16}$$

This equation is an extension of Bellman-Ford's formula for computing the shortest path distance in a graph (see, e.g., [34, 62, 66, 135, 201, 215]). Moreover, the recurrence expression (9.16) is also a generalization of the distributed consensus algorithm developed in [227], considering binary costs only.

It was shown [95, 94] that this minimal free energy interpolates between the least cost ($T = \theta^{-1} \to \infty$; $\phi_i^*(\infty) = \min_{j \in \mathcal{S}ucc(i)} \{c_{ij} + \phi_j^*(\infty)\}$ and $\phi_n^*(\infty) = 0$) and the expected cost before absorption ($T = \theta^{-1} \to 0$; $\phi_i^*(0) = \sum_{j \in \mathcal{S}ucc(i)} p_{ij}^{\mathrm{ref}} (c_{ij} + \phi_j^*(0))$ and $\phi_n^*(0) = 0$) [144, 95, 94]. In addition [144, 95, 94], this quantity defines a **directed distance** between any node and absorbing node n.

In fact, as discussed in [95, 94], this last expression is obtained by simply replacing the $\min$ operator by a weighted version of the **softmin operator** ([64]; also called the Log-Sum-Exp function [46, 227]) in the standard Bellman-Ford recurrence formula,

$$\mathrm{softmin}_{\mathbf{q},\theta}(\mathbf{x}) = -\tfrac{1}{\theta} \log \left(\sum_{j=1}^{n} q_j \exp[-\theta x_j] \right) \text{ with all } q_j \geq 0 \text{ and } \textstyle\sum_{j=1}^{n} q_j = 1 \tag{9.17}$$

which is a smooth approximation of the min operator and interpolates between weighted average and minimum operators, depending on the parameter θ [64, 227]. It also appeared in control [237] and exploration strategies for which an additional Kullback-Leibler cost term is incorporated in the immediate cost [207, 137, 234, 233, 18]. Moreover, this function was recently proposed as an operator guiding exploration in reinforcement learning, and more specifically for the SARSA algorithm [13, 14].

Note that the optimal mixed policy derived in Equation (9.11) can be rewritten in function of the free energy as

$$p^*_{ij}(T) = \frac{p^{\text{ref}}_{ij} \exp[-\theta c_{ij}] z_{jn}}{\sum_{j'=1}^{n} p^{\text{ref}}_{ij'} \exp[-\theta c_{ij'}] z_{j'n}} = \frac{p^{\text{ref}}_{ij} \exp[-\theta(c_{ij} + \phi^*_j(T))]}{\sum_{j'=1}^{n} p^{\text{ref}}_{ij'} \exp[-\theta(c_{ij'} + \phi^*_{j'}(T))]} \tag{9.18}$$

because $z_{in} = \exp[-\theta\phi^*_i(T)]$ and $z_{in} = \sum_{j=1}^{n} w_{ij} z_{jn} = \sum_{j=1}^{n} p^{\text{ref}}_{ij} \exp[-\theta c_{ij}] z_{jn}$ for all $i \neq n$. This corresponds to a multinomial logistic function.

9.3 Randomized shortest paths with constrained transition probabilities

Interestingly, the randomized shortest path formulation can easily be extended to account for some types of constraints. The goal here is to determine the best mixed policy for reaching destination node n from source node 1 subject to **equality constraints** on some transition probabilities, given by Equation (9.18). We proceed as in previous section, but we now constrain the transition probabilities associated to some nodes to be equal to predefined values provided by the user. In other words, we constrain the relative flow passing through the edges starting from some nodes belonging to a set of nodes $\mathcal{C}$ (the constrained nodes).

9.3.1 The Lagrange function

More precisely, the considered constraints on the nodes $i \in \mathcal{C}$ are

$$p^*_{ij}(T) = \frac{\bar{n}_{ij}(T)}{\bar{n}_i(T)} = \frac{\bar{n}_{ij}(T)}{\sum_{j \in \mathcal{S}ucc(i)} \bar{n}_{ij}(T)} = q_{ij} \text{ for the edges starting in nodes } i \in \mathcal{C} \tag{9.19}$$

which are independent of T. These transition probabilities q_{ij} are specified by the user for all the nodes in $\mathcal{C}$. We assume that these constraints are feasible. In particular, we must have $\sum_{j \in \mathcal{S}ucc(i)} q_{ij} = 1$ for all $i \in \mathcal{C}$. Moreover, the RSP model (see Equation (9.2)) implies that we should recover a pure random walk behavior, with reference probabilities provided by Equation (9.1), when $T \to \infty$. Therefore, to be consistent, the reference probabilities must be chosen such that $p^{\text{ref}}_{ij} = p^*_{ij}(T = \infty) = q_{ij}$ for nodes $i \in \mathcal{C}$. We assume that this is the case in the sequel.

We consider the following Lagrange function

$$\mathscr{L}(\mathrm{P},\boldsymbol{\lambda}) = \underbrace{\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\tilde{c}(\wp) + T\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right)}_{\text{free energy, } \phi(\mathrm{P})} + \mu\Big(\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp) - 1\Big)$$
$$+ \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{S}ucc(i)} \lambda_{ij}\Bigg[\underbrace{\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\,\eta\big((i,j)\in\wp\big)}_{\bar{n}_{ij}(T)} - q_{ij}\underbrace{\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\,\eta(i\in\wp)}_{\bar{n}_i(T)}\Bigg] \tag{9.20}$$

where, as before, $\mathcal{P}$ is the set of paths from node 1 to node n and with $\eta\big((i,j)\in\wp\big)$ being the number of times edge (i,j) appears on path $\wp$. In a similar way, $\eta(i\in\wp)$ is the number of times node i is visited on path $\wp$.

The Lagrange function can be rearranged as

$$\mathscr{L}(\mathrm{P},\boldsymbol{\lambda}) \tag{9.21}$$
$$= \sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\underbrace{\Bigg[\tilde{c}(\wp) + \sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{S}(i)} \lambda_{ij}\,\eta\big((i,j)\in\wp\big) - \sum_{i\in\mathcal{C}} \eta(i\in\wp)\sum_{j'\in\mathcal{S}(i)} q_{ij'}\,\lambda_{ij'}\Bigg]}_{\tilde{c}'(\wp)}$$
$$+ T\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right) + \mu\Big(\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp) - 1\Big)$$
$$= \underbrace{\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\tilde{c}'(\wp) + T\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right)}_{\text{free energy } \phi'(\mathrm{P})} + \mu\Big(\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp) - 1\Big) \tag{9.22}$$

where $\mathcal{S}(i)$ has been used as a shortcut for $\mathcal{S}ucc(i)$ and, by using $\eta(i\in\wp) = \sum_{j\in\mathcal{S}(i)} \eta\big((i,j)\in\wp\big)$, the local costs c_{ij} are redefined as

$$c'_{ij} = \begin{cases} c_{ij} + \underbrace{\lambda_{ij} - \sum_{j'\in\mathcal{S}ucc(i)} q_{ij'}\lambda_{ij'}}_{\text{cost update } \varDelta_{ij}} & \text{when node } i\in\mathcal{C} \\ c_{ij} & \text{otherwise} \end{cases} \tag{9.23}$$

and $\mathbf{C}'$ will be the matrix containing these new costs $c'_{ij} = c_{ij} + \varDelta_{ij}$, which are called the **augmented** costs. We observe that Equation (9.22) is exactly a randomized shortest paths problem (see Equation (9.2)) containing augmented costs instead of the initial costs.

We further observe that the weighted (by transition probabilities) mean of the cost updates $\Delta_{ij} = \lambda_{ij} - \sum_{j' \in \mathcal{S}ucc(i)} q_{ij'} \lambda_{ij'}$ is equal to zero on each node $i \in \mathcal{C}$: $\sum_{j \in \mathcal{S}ucc(i)} q_{ij} \Delta_{ij} = 0$. This implies that the weighted average of the augmented costs is equal to the weighted average of the original costs on each constrained node i, $\sum_{j \in \mathcal{S}ucc(i)} q_{ij} c'_{ij} = \sum_{j \in \mathcal{S}ucc(i)} q_{ij} c_{ij}$. Therefore, this choice ensures that the expected augmented cost is equal to the real expected cost provided by Equation (9.13), $\langle \tilde{c}' \rangle = \mathbf{e}^{\mathrm{T}}(\mathbf{N} \circ \mathbf{C}')\mathbf{e} = \langle \tilde{c} \rangle$. In this case, the perceived cost when visiting any node using the augmented costs ($\mathbf{C}'$) is exactly the same in average as the perceived real cost ($\mathbf{C}$) when no constraint is introduced.

Thus, in Equation (9.22), everything happens as if the costs have been redefined by taking into account the Lagrange parameters. These Lagrange parameters can therefore be interpreted as additional costs necessary to satisfy the equality constraints. We now find the λ_i by using Lagrangian duality.

Let $\phi'(\mathrm{P})$ be the free energy obtained from these augmented costs. We now have to find the λ_{ij} by Lagrangian duality.

9.3.2 Exploiting Lagrangian duality

In this section, as in [126, 131], we will take advantage of the fact that, in our formulation of the problem, the Lagrange dual function and its gradient are easy to compute.

As the objective function is convex and all the equality constraints are linear, there is only one global minimum and the duality gap is zero [35, 69, 112]. The optimum is a saddle point of the Lagrange function and a common optimization procedure (often called Arrow-Hurwicz-Uzawa [12]) consists in sequentially (i) solving the primal while considering the Lagrange parameters as fixed and then (ii) solving the dual (which is concave) while considering the variables to be optimized (the paths probabilities) as fixed, until convergence.

In our context, this provides the two following steps which are iterated

$$\begin{cases} \mathscr{L}(\mathrm{P}^*, \boldsymbol{\lambda}) = \underset{\{\mathrm{P}(\wp)\}_{\wp \in \mathcal{P}}}{\text{minimize}} \, \mathscr{L}(\mathrm{P}, \boldsymbol{\lambda}) & \text{(compute the dual function)} \\ \mathscr{L}(\mathrm{P}^*, \boldsymbol{\lambda}^*) = \underset{\boldsymbol{\lambda}}{\text{maximize}} \, \mathscr{L}(\mathrm{P}^*, \boldsymbol{\lambda}) & \text{(maximize the dual function)} \end{cases} \tag{9.24}$$

and this is the procedure that is followed, where the dual function is maximized through a simple coordinate ascend on Lagrange multipliers.

9.3.3 Computing the dual function

We already know from (9.3) that the first step in Equation (9.24) leads to

$$\mathrm{P}^*(\wp) = \frac{\tilde{\pi}(\wp)\exp[-\theta\tilde{c}'(\wp)]}{\sum\limits_{\wp'\in\mathcal{P}} \tilde{\pi}(\wp')\exp[-\theta\tilde{c}'(\wp')]} = \frac{\tilde{\pi}(\wp)\exp[-\theta\tilde{c}'(\wp)]}{\mathcal{Z}'} \tag{9.25}$$

where $\tilde{c}'(\wp)$ is the augmented cost of a path. Notice that once the optimal probability distribution on paths has been computed, the expected number of transitions through any edge can be deduced from Equation (9.5) and the transition probabilities from (9.11). The partition function $\mathcal{Z}'$, and the z'_{in} in general, can then be obtained from Equation (9.11).

Then, from Equations (9.4) and (9.22), the dual function can easily be computed in function of the partition function defined from the augmented costs,

$$\mathscr{L}(\mathrm{P}^*, \boldsymbol{\lambda}) = -T\log\mathcal{Z}' \tag{9.26}$$

and has to be maximized with respect to the $\{\lambda_{ij}\}$ with $i \in \mathcal{C}$ and $j \in \mathcal{S}ucc(i)$.

9.3.4 Maximizing the dual function

Let us now try to maximize the dual function. Because $\bar{n}_i(T) = \sum_{j\in\mathcal{S}ucc(i)} \bar{n}_{ij}(T)$, by following the reasoning of previous subsection, we obtain

$$\begin{aligned}\frac{\partial\mathscr{L}(\mathrm{P}^*, \boldsymbol{\lambda})}{\partial\lambda_{ij}} &= \frac{\partial(-T\log\mathcal{Z}')}{\partial\lambda_{ij}} \\ &= \sum_{j'\in\mathcal{S}ucc(i)} \frac{\partial(-T\log\mathcal{Z}')}{\partial c'_{ij'}}\frac{\partial c'_{ij'}}{\partial\lambda_{ij}} \\ &= \sum_{j'\in\mathcal{S}ucc(i)} \bar{n}_{ij'}(T)(\delta_{jj'} - q_{ij}) \\ &= \bar{n}_{ij}(T) - q_{ij}\bar{n}_i(T)\end{aligned} \tag{9.27}$$

Quite naturally, setting the result to zero provides the constraints for nodes $i \in \mathcal{C}$,

$$\frac{\bar{n}_{ij}(T)}{\bar{n}_i(T)} = q_{ij} \tag{9.28}$$

and we now have to solve this equation in terms of the Lagrange parameter, λ_{ij}.

9.3.5 Computing the Lagrange parameters

Recalling that $\bar{n}_i(T) = \sum_{j \in \mathcal{S}ucc(i)} \bar{n}_{ij}(T)$ and Equations (9.9) and (9.11), we obtain, for each $i \in \mathcal{C}$ and $j \in \mathcal{S}ucc(i)$,

$$\frac{p_{ij}^{\mathrm{ref}} \exp[-\theta c'_{ij}] z_{jn}}{z_{in}} = q_{ij} \tag{9.29}$$

The objective of this subsection is to compute each augmented cost (and thus the Lagrange parameter λ_{ij}, see Equation (9.23)) corresponding to nodes $i \in \mathcal{C}$ from this equation by isolating c'_{ij}, given that the backward variables z_{in} are fixed once we know the optimal path distribution.

Recall (see the discussion following Equation (9.19)) that it was assumed that the imposed q_{ij} and the reference transition probabilities p_{ij}^{ref} are consistent on constrained nodes, that is, $p_{ij}^{\mathrm{ref}} = q_{ij}$ for all $i \in \mathcal{C}$. We thus obtain, for $i \in \mathcal{C}$,

$$\frac{\exp[-\theta c'_{ij}] z_{jn}}{z_{in}} = \frac{\exp[-\theta c'_{ij}] z_{jn}}{\sum_{j' \in \mathcal{S}ucc(i)} p_{ij'}^{\mathrm{ref}} \exp[-\theta c'_{ij'}] z_{j'n}} = 1 \tag{9.30}$$

as $z_{in} = \sum_{j=1}^{n} p_{ij}^{\mathrm{ref}} \exp[-\theta c_{ij}] z_{jn}$ for all $i \neq n$ (see Equation (9.18)).

This shows that the augmented costs related to a constrained node are only defined up to the addition of a constant term (a translation). This gives us the opportunity to constrain the weighted average of the augmented costs $c'_{ij} = c_{ij} + \Delta_{ij}$ to be equal to the weighted average of the original costs, as required – see the discussion following Equation (9.23). The Δ_{ij} have to be computed from this previous equation while satisfying this constraint, which reduces to $\Delta_{ij} = \lambda_{ij} - \sum_{j' \in \mathcal{S}ucc(i)} q_{ij'} \lambda_{ij'}$, as shown before.

Let us isolate Δ_{ij} in Equation (9.30):

$$\exp[\theta \Delta_{ij}] = \frac{\exp[-\theta c_{ij}] z_{jn}}{z_{in}} \tag{9.31}$$

Then, by taking $\frac{1}{\theta} \log$ of both sides,

$$\begin{aligned} \Delta_{ij} &= \tfrac{1}{\theta} \log \left(\exp[-\theta c_{ij}] z_{jn} \right) - \tfrac{1}{\theta} \log z_{in} \\ &= \underbrace{\tfrac{1}{\theta} \log(z_{jn}) - c_{ij}}_{\lambda_{ij}} - \tfrac{1}{\theta} \log z_{in} \end{aligned} \tag{9.32}$$

where we defined $\lambda_{ij} = \frac{1}{\theta}\log(z_{jn}) - c_{ij}$. Multiplying both sides by q_{ij} and summing over j provides

$$\tfrac{1}{\theta}\log z_{in} = \sum_{j \in \mathcal{S}ucc(i)} q_{ij}\left(\tfrac{1}{\theta}\log(z_{jn}) - c_{ij}\right) \tag{9.33}$$

which, from (9.32), just aims to center the λ_{ij}, as required.

This suggests the following sequential steps to be applied on each node $i \in \mathcal{C}$ in turn

$$\begin{cases} \lambda_{ij} \leftarrow \frac{1}{\theta}\log(z_{jn}) - c_{ij} & \text{for all } j \in \mathcal{S}ucc(i) \\ \varDelta_{ij} \leftarrow \lambda_{ij} - \sum_{j' \in \mathcal{S}ucc(i)} q_{ij'}\lambda_{ij'} & \text{for all } j \in \mathcal{S}ucc(i) \\ c'_{ij} \leftarrow c_{ij} + \varDelta_{ij} & \text{for all } j \in \mathcal{S}ucc(i) \end{cases} \tag{9.34}$$

until convergence. Recall also that, for consistency, $q_{ij} = p^{\text{ref}}_{ij}$ on constrained nodes.

9.3.6 The final procedure

Therefore, after initializing the Lagrange parameters to 0, the final procedure iterates the following steps:

- The elements of the fundamental matrix are computed thanks to Equation (9.7) from the augmented costs c'_{ij} (defined in Equation (9.23)) and from the transition matrix of the natural random walk on G (Equation (9.1)), where destination node n is made absorbing and killing.
- The augmented costs are updated for all edges incident to constrained nodes (in $\mathcal{C}$) thanks to the three cases detailed in Equation (9.34).

After convergence of the two previous steps, the optimal policy is computed thanks to Equation (9.11) by using the augmented costs c'_{ij} instead of c_{ij}. This provides optimal transition probabilities $p^*_{ij}(T)$. We now apply this procedure in order to solve simple Markov decision problems.

9.4 Markov decision processes as a constrained randomized shortest path on a bipartite graph

The previous sections described all the needed tools for computing an optimal mixed policy on a Markov decision process (MDP). Recall that, as in [34], we assume that there is a special cost-free **destination** or **goal** node n; once the system has reached that node, it simply disappears (killing node). Thus, node

n has no outgoing edge. As in [208], we also consider a problem structure such that termination is inevitable. Thus, the horizon is in effect finite, but its length is random and it depends on the policy being used. The conditions for which this is true are, basically, related to the fact that the destination node can be reached in a finite number of steps from any potential initial node; for a rigorous treatment, see e.g. [34, 32].

The main objective is thus to design a **randomized mixed policy** minimizing the expected cost-to-go subject to an entropy constraint controlling the total randomness spread in the network, and therefore the exploration rate. In other words, we are looking for an optimal policy or, in our case, an optimal transition probabilities matrix $\mathbf{P}^*$ of a finite, first-order, Markov chain minimizing the expected cost needed to reach the destination state from the initial state, while fixing the entropy spread in the chain as well as the transition probabilities provided by the environment.

Therefore, the solution is obtained by the algorithm described in Subsection 9.3.6 – the randomized shortest paths with constraints on transition probabilities – applied to a bipartite graph, as described now.

9.4.1 Basic framework and procedure

The Markov decision process is now viewed as a (constrained) randomized shortest path on a bipartite graph (see Figure 9.1). First, we examine how the reference transition probabilities defining the natural random walk on this graph are defined. Then, the structure of the bipartite graph and the way to compute the optimal policy are described. Finally, the precise form of the matrices needed to run the constrained RSP is detailed.

Pure random walk: reference probabilities. More precisely, in the case of a **pure random walk** (the reference transition probabilities (Equation 9.1), corresponding to $T \to \infty$ in Equation (9.2)), we consider that agents are sent from an initial state 1 and that, at each state $s = k$ (n states in total, $\mathcal{S}$), they choose an action $a_k = u$ with a probability mass p_{ku}^{ref}, $k \in \mathcal{S} \setminus n$ and $u \in \mathcal{U}(k)$, the set of actions available in state k. When no prior information on the system is available, these are usually set to $p_{ku}^{\mathrm{ref}} = 1/|\mathcal{U}(k)|$ (a uniform distribution). In our bipartite graph, $\mathcal{U}(k)$ is nothing else than the successors of node k, $\mathcal{U}(k) = \mathcal{S}ucc(k)$. State n is still the absorbing, killing, state (the goal state).

Agents then perform the chosen action u, and incur a finite cost c_{ku} associated to the execution of action u in state k. They then jump to the next state $s = l$ with a reference transition probability p_{ul}^{ref} provided by the environment, where $l \in \mathcal{S}$ and $u \in \mathcal{A}$, depending on the action. This behavior corresponds to a pure exploration, according to the Markov chain with transition probabilities p_{ul}^{ref} and, for consistency, these transition probabilities must be equal to the

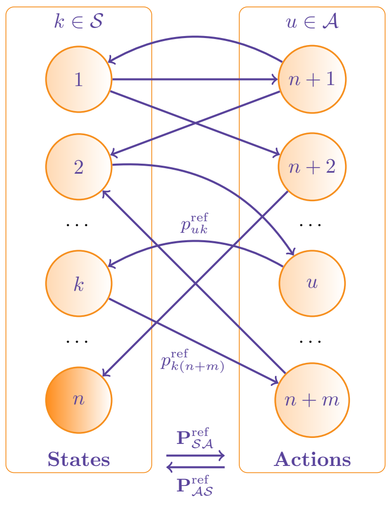

$k \in \mathcal{S}$
$u \in \mathcal{A}$
1
2
k
n
$n+1$
$n+2$
u
$n+m$
p^{ref}_{uk}
$p^{\text{ref}}_{k(n+m)}$
States
Actions
$\mathbf{P}^{\text{ref}}_{\mathcal{SA}}$
$\mathbf{P}^{\text{ref}}_{\mathcal{AS}}$

FIGURE 9.1: The bipartite graph with states on the left side ($\mathcal{S}$) and control actions on the right ($\mathcal{A}$). Node 1 is the initial state while node n is the absorbing, destination, state of the system. The transition probabilities from states to actions p^{ref}_{ku} are gathered in matrix $\mathbf{P}^{\text{ref}}_{\mathcal{SA}}$ and the transition probabilities from actions to states p^{ref}_{ku}, provided by the environment, are gathered in matrix $\mathbf{P}^{\text{ref}}_{\mathcal{AS}}$.

constrained transition probabilities, q_{ul}, as discussed in the previous section. Notice that an additional cost could also be associated to the transition to state l, after action u is performed, as, e.g., in [225], but in this work we adopt the simpler setting where the cost is a function of the action u in state k only [194, 235].

Definition of the bipartite graph. Therefore, the process can be modeled as a directed **bipartite graph** G_{b} in which the left nodes are the original states $\mathcal{S}$ and the right nodes are the possible actions associated to the states, $\mathcal{A} = \mathcal{U}(1) \cup \mathcal{U}(2) \cup \ldots \cup \mathcal{U}(n-1)$ (n is absorbing and has no associated action). We have $n = |\mathcal{S}|$ and $m = |\mathcal{A}|$. Note that each action associated to a state is a node of G_{b}, even if the same action is also available in some other nodes – actions are duplicated for each node in which they appear. Therefore, the number of such right nodes is $|\mathcal{A}| = |\mathcal{U}(1)| + |\mathcal{U}(2)| + \cdots + |\mathcal{U}(n-1)| = m$.

Moreover, it is assumed that, in this bipartite graph G_{b}, the nodes corresponding to states $\mathcal{S}$ are numbered first (from 1 to n) and actions $\mathcal{A}$ are following (from $n+1$ to $n+m$). Therefore, the set of available actions in any state k is nothing else that the successor nodes of $k, \mathcal{U}(k) = \mathcal{S}ucc(k)$.

The optimal mixed policy. When the temperature T decreases, the agents are more and more exploiting good policies while still exploring the environment – they interpolate between a purely random behavior (guided by the reference probabilities) and the best, deterministic, policy solving the Markov decision process, provided, e.g., by the value iteration algorithm [197, 224, 33, 34]. The control actions $u \in \mathcal{A}$ are then chosen according to an optimal stochastic **policy** Π^*, mapping every state k to the set $\mathcal{U}(k)$ of available actions with a probability mass $p^*_{ku}(T)$ provided by the randomized shortest path model applied on graph G_{b} (see Equation (9.11)). The policy $\Pi \equiv \{p^*_{ku}(T), k \in \mathcal{S} \setminus n \wedge u \in \mathcal{U}(k)\}$ defines, for each state k, an optimal probability distribution on the set $U(k)$ of actions available in this state, provided by Equation (9.11), and gradually biasing the walk towards the optimal, deterministic, policy. Recall that the policy is optimal in the sense that it minimizes expected cost for a given degree of relative entropy (see Equation (9.2)).

For instance, if the indexes of the available actions in some state k are $\mathcal{U}(k) = \{5, 6, 7\}$, the probability mass $p^*_{ku}(T)$ specifies three probabilities $p^*_{k5}(T)$, $p^*_{k6}(T)$, and $p^*_{k7}(T)$, summing to one. These optimal transition probabilities are provided by the policy obtained from the constrained randomized shortest paths model applied on the bipartite graph G_{b}, that is, by Equation (9.11).

As discussed in [208], such random choices are common in a variety of fields, for instance decision sciences [200] or game theory, where they are called mixed strategies (see, e.g., [186]). Thus, the problem tackled in this section simply corresponds to a constrained **randomized shortest path problem** (RSP) on G_{b}.

Useful matrices. Recall that, in the randomized shortest path framework, the procedure for computing the optimal policy takes three quantities in input: the **reference transition probabilities**, the **cost matrix** and the **set of constrained nodes**.

The $n \times m$ reference transition probabilities matrix $\mathbf{P}^{\mathrm{ref}}_{\mathcal{SA}}$ (states-actions) contains the p^{ref}_{ku} for the transitions from the left nodes (states in $\mathcal{S}$) and the right nodes (actions in $\mathcal{A}$) of G_{b}. For node n (the killing absorbing node), the corresponding row of $\mathbf{P}^{\mathrm{ref}}_{\mathcal{SA}}$ (the last row) is set equal to 0. Indeed, there are no actions associated to this node.

In a symmetric way, once an action in $\mathcal{A}$ has been chosen, the agent is in the corresponding action node in the bipartite graph G_{b}, say node $u \in \mathcal{A}$. Then, he jumps to a state $k \in \mathcal{S}$ with probability p^{ref}_{uk} (actions-states), predefined by the environment. It reflects the probability of ending in state $s = k$ once action $a = u$ has been chosen. The corresponding $m \times n$ transition probabilities matrix from actions to states is $\mathbf{P}^{\text{ref}}_{\mathcal{AS}}$ and its elements are equal to the constrained transition probabilities, $p^{\text{ref}}_{uk} = q_{uk}$. The set of action nodes is therefore the set of constrained nodes.

Consequently, the transition matrix of the bipartite graph G_{b} is

$$\mathbf{P}^{\text{ref}} = \begin{array}{c|cc} & \mathcal{S} & \mathcal{A} \\ \hline \mathcal{S} & \mathbf{O} & \mathbf{P}^{\text{ref}}_{\mathcal{SA}} \\ \mathcal{A} & \mathbf{P}^{\text{ref}}_{\mathcal{AS}} & \mathbf{O} \end{array} \tag{9.35}$$

where $\mathbf{O}$ is a 0 matrix of the appropriate size. Its elements are p^{ref}_{ij}.

Moreover, the cost matrix for the bipartite graph G_{b} is

$$\mathbf{C}_{\text{b}} = \begin{array}{c|cc} & \mathcal{S} & \mathcal{A} \\ \hline \mathcal{S} & \mathbf{O} & \mathbf{C} \\ \mathcal{A} & \mathbf{O} & \mathbf{O} \end{array} \tag{9.36}$$

that is, as in [34, 194], costs are only defined on state-action edges, c_{ku} with $k \in \mathcal{S} \setminus n$ and $u \in \mathcal{A}$. The other costs are equal to zero (no incurred cost for action-state transitions). Extensions to costs defined on action-state edges are possible, but are omitted to keep things simple. Row n of this matrix, corresponding to the killing absorbing node, is set to 0.

Finally, the set of constrained nodes is simply the set of action nodes $\mathcal{A}$ from which the transition probabilities are provided by the environment, $\mathbf{P}^{\text{ref}}_{\mathcal{AS}}$.

Computing the optimal policy. Once these matrices are computed, the optimal randomized policy interpolating between the optimal deterministic policy of the standard MDP and a pure random walk defined by the reference transition matrix $\mathbf{P}^{\text{ref}}$ is obtained by applying the randomized shortest path procedure on the graph G_{b} (see Subsection 9.3.6). The optimal mixed policy is obtained from Equation (9.11).

We now describe a simplified way for obtaining the optimal mixed policy, inspired by the value iteration algorithm.

9.4.2 A simple soft value iteration algorithm

Interestingly and surprisingly, we will now show that replacing the minimum operator by a softmin operator in the standard value iteration algorithm provides exactly the same results as the constrained RSP. This property leads to a simple algorithm, extending the standard value iteration algorithm, for computing randomized policies.

This shows that the proposition of using the softmin function for exploration in reinforcement learning [18, 14, 207, 137, 234, 233] is also globally optimal in that it minimizes expected path cost subject to a fixed relative entropy of paths equality constraint (see Equation (9.2)), at least in our setting of a absorbing, goal, node n reachable from any other node of the graph.

The value iteration algorithm

Let us first recall the **standard value iteration** procedure, computing the expected cost until absorption by the goal state n [197, 224, 33, 34] when starting from a node $k \in \mathcal{S}$, denoted by v_{kn}, and based on the following recurrence formula

$$v_{kn} = \begin{cases} \min\limits_{u \in \mathcal{U}(k)} \left\{ c_{ku} + \sum\limits_{l \in \mathcal{S}ucc(u)} p^{\text{ref}}_{ul} v_{ln} \right\} & \text{if } k \neq n \\ 0 & \text{if } k = n \end{cases} \tag{9.37}$$

where p^{ref}_{ul} is element u, l (with $u \in \mathcal{A}$ and $l \in \mathcal{S}$) of matrix $\mathbf{P}_{\text{ref}}$ of the reference random walk on the bipartite graph (see Equation (9.35)). This expression is iterated until convergence, which is guaranteed under some mild conditions, for any set of nonnegative initial values (see, e.g., [194, 197, 224, 33, 34] for details).

The soft value iteration algorithm

We start from the softmin form of the free energy ([95, 94, 90]; see also Equation (9.16)), which corresponds to a Bellman-Ford algorithm for computing the shortest path distance in which the min operator has been replaced by the softmin operator defined in Equation (9.17). Substituting in the same way the softmin operator for the min in the value iteration update formula provides a randomized equivalent of the Bellman-Ford optimality conditions,

$$\phi^{\mathcal{S}}_k(T) = \begin{cases} -\frac{1}{\theta} \log \left[\sum\limits_{u \in \mathcal{U}(k)} p^{\text{ref}}_{ku} \exp \left[-\theta \Big(c_{ku} + \sum\limits_{l \in \mathcal{S}ucc(u)} p^{\text{ref}}_{ul} \phi^{\mathcal{S}}_l(T) \Big) \right] \right] & \text{if } k \neq n \\ 0 & \text{if } k = n \end{cases} \tag{9.38}$$

which is similar to the expression computing the free energy in Equation (9.16) where the update is replaced by an equality. The quantity $\phi^{\mathcal{S}}_k(T) = -T \log z_{kn} = -\frac{1}{\theta} \log z_{kn}$ (see Appendix B for details), where z_{kn} is the backward variable introduced in Equation (9.8), will be called the **free energy** of the

Markov decision process, associated to the different states $k \in \mathcal{S}$. This equation states the necessary optimality conditions and will be called the **Bellman-Ford optimality conditions** for the randomized Markov decision process.

Note that it can be shown (see the appendix of [95, 94]) that this recurrence formula reduces to the standard optimality conditions for Markov decision processes (Equation (9.37)) when $\theta \to \infty$. Conversely, when $\theta \to 0$, it reduces to the expression allowing to compute the expected cost until absorption by the goal state n, also called the average first-passage cost [141, 185], $\phi_k^{\mathcal{S}}(T) = \sum_{u \in \mathcal{U}(k)} p_{ku}^{\text{ref}} (c_{ku} + \sum_{l \in \mathcal{S}ucc(u)} p_{ul}^{\text{ref}} \phi_l^{\mathcal{S}}(T))$. Furthermore, it can be shown that the backward variables z_{in} can be interpreted as the probabilities of reaching the goal state n, and thus of surviving, during a killed random walk on G_{b} with transition probabilities $w'_{ij} = p_{ij}^{\text{ref}} \exp[-\theta c'_{ij}]$, for $i \in \mathcal{S}$, $j \in \mathcal{A}$, and vice-versa [95, 94, 90]. The quantity $\phi_k^{\mathcal{S}}(T)$ is the global cost (free energy) associated to the logarithmh of this probability of surviving.

This suggests the use of the following soft form of value iteration for computing the solution of the randomized Markov decision system by replacing the equality by an update in (9.38),

$$\phi_k^{\mathcal{S}}(T) \leftarrow \begin{cases} -\frac{1}{\theta} \log \left[\sum\limits_{u \in \mathcal{U}(k)} p_{ku}^{\text{ref}} \exp \left[-\theta \Big(c_{ku} + \sum\limits_{l \in \mathcal{S}ucc(u)} p_{ul}^{\text{ref}} \phi_l^{\mathcal{S}}(T) \Big) \right] \right] & \text{if } k \neq n \\ 0 & \text{if } k = n \end{cases} \tag{9.39}$$

which has to be iterated over all states until convergence.

After convergence of the values to a fixed point, optimality conditions (9.38) should be verified. Then, the optimal policy for each node $k \in \mathcal{S} \setminus n$ and $k \neq n$ is computed thanks to

$$p_{ku}^{*}(T) = \frac{p_{ku}^{\text{ref}} \exp \left[-\theta \Big(c_{ku} + \sum\limits_{l \in \mathcal{S}ucc(u)} p_{ul}^{\text{ref}} \phi_l^{\mathcal{S}}(T) \Big) \right]}{\sum\limits_{u' \in \mathcal{U}(k)} p_{ku'}^{\text{ref}} \exp \left[-\theta \Big(c_{ku'} + \sum\limits_{l \in \mathcal{S}ucc(u')} p_{u'l}^{\text{ref}} \phi_l^{\mathcal{S}}(T) \Big) \right]} \quad \text{for } k \neq n \tag{9.40}$$

which provides the probability of choosing action u within state k.

This procedure, involving the iteration of Equation (9.38) and the computation of the optimal policy from Equation (9.40), will be called the **soft value iteration** algorithm.

As for the free energy of the bag-of-paths system [95, 94], we derive – and thus justify theoretically – this iterative algorithm from the randomized shortest paths framework. The derivation is provided in Appendix B.

We observed empirically in all our experiments that both techniques (the soft value iteration and the constrained randomized shortest path procedures)

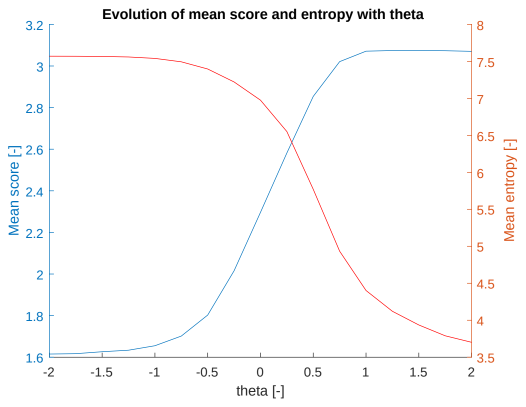


FIGURE 9.2: For an increasing θ (in logarithmic scale), all mixed strategies based on the soft value iteration (see Equation (9.39)) are comared and results are reported. The blue curve depicts the evolution of the average (over $10e6$ turns) score at the end of each turn (reward, larger is better) for an "artificial" player, whose policy is provided by the constrained randomized shortest path model, in terms of the strategies obtained with different values of the θ parameter. Conversely, the red curve indicates the average entropy of the different strategies. The largest entropy is achieved when θ is smallest, and is minimum when θ is largest.

converge and provide exactly the same policies.

9.5 Simulations and discussion: application to the 421 dice game

This section describes an experiment illustrating the application of constrained randomized shortest paths to Markov decision problems. Several different simulations have been run on four different problems but, in order to save space, we decided to report only one application, the 421 game.

9.5.1 Rules of the 421 game

The 421 dice game (quatre-cent-vingt-et-un in french [133]) is popular in France and Belgium and is played with three six-faced dice and 11 tokens (representing penalties). A player looses the game when he gets all tokens – the goal is thus to get rid of our tokens. The game is composed of two phases:

TABLE 9.1: Results of the soft value iteration method (SVI with a high θ) performed on the 421 game. The first column represents the states of the set of three dices (dices drawings are sorted in increasing order). The second column provides the corresponding reward scores defined in the game. The third and fourth columns show the optimal strategies provided by the Markov decision process, respectively for the first re-roll and the second one (it indicates which dice has to be re-rolled, formatted as three booleans).

DDD	**Score**	**SVI reroll 1**	**SVI reroll 2**
111	7	0 0 0	0 0 0
211	2	1 0 0	1 0 0
221	1	0 1 0	0 1 0
222	3	0 0 0	0 0 0
311	3	1 0 0	1 0 0
321	2	1 0 0	1 0 0
322	1	1 1 1	1 1 1
331	1	1 1 0	1 1 0
332	1	1 1 1	1 1 1
333	3	0 0 0	0 0 0
411	4	1 0 0	1 0 0
421	8	0 0 0	0 0 0
422	1	0 0 1	0 0 1
431	1	0 1 0	0 1 0
432	2	0 1 0	0 1 0
433	1	0 1 1	0 1 1
441	1	0 1 0	0 1 0
442	1	0 1 0	0 1 0
443	1	1 0 1	1 0 1
444	3	0 0 0	0 0 0
511	5	1 0 0	0 0 0
521	1	1 0 0	1 0 0
522	1	1 1 1	1 1 1
531	1	1 1 0	1 1 0
532	1	1 1 1	1 1 1
533	1	1 1 1	1 1 1
541	1	1 0 0	1 0 0
542	1	1 0 0	1 0 0
543	2	1 0 1	0 0 0
544	1	1 0 1	1 0 1
551	1	1 1 0	1 1 0
552	1	1 1 1	0 0 1
553	1	1 1 1	0 0 1
554	1	1 1 0	1 1 0
555	3	0 0 0	0 0 0
611	6	0 0 0	0 0 0
621	1	1 0 0	1 0 0
622	1	1 1 1	1 1 1
631	1	1 1 0	1 1 0
632	1	1 1 1	1 1 1
633	1	1 1 1	1 1 1
641	1	1 0 0	1 0 0
642	1	1 0 0	1 0 0
643	1	1 0 1	1 0 1
644	1	1 0 1	1 0 1
651	1	1 1 0	1 1 0
652	1	1 1 1	1 1 1
653	1	1 1 1	1 1 1
654	2	1 1 0	0 0 0
655	1	1 1 1	1 0 0
661	1	1 1 0	1 1 0
662	1	0 0 1	0 0 1
663	1	0 0 1	0 0 1
664	1	0 0 1	0 0 1
665	1	0 0 1	0 0 1
666	3	0 0 0	0 0 0

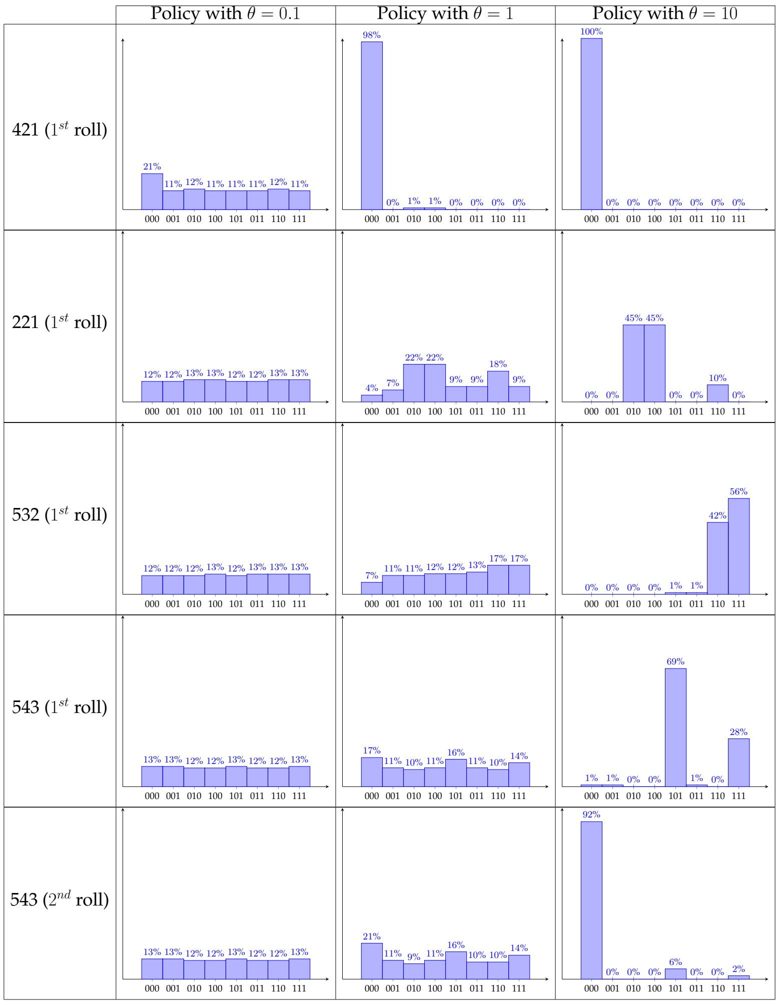


FIGURE 9.3: Mixed (randomized) policy provided by the soft value iteration (SVI, Equations (9.39)-(9.40)), for some interesting cases. The resulting policies interpolate between a uniform distribution (equal to the reference transition probabilities, left) when $\theta \to 0$ and the deterministic, optimal, policy obtained by the value iteration algorithm (sum of Kronecker delta, right), when $\theta \to \infty$.

- During the first phase, each player rolls all three dices (no re-rolls are allowed) and reward scores according to the second column on Table 9.1. Observe that after each full turn, the player with the **lowest resulting score** (according to Table 9.1) must take a certain number of tokens from the pot. This number is equal to the **highest resulting score** among the players for this turn. The game continues until the pot is empty. Therefore, the goal of the first phase is to distribute the 11 tokens among players. Thereafter, the second phase can start.

- During the second phase, the players also roll all three dices a first time, but are now allowed to further **re-roll them two times** (all or part of the three dices). Thus, after having rolled all three dices once, the player has the choice to re-roll dice number 1, number 2 or number 3, two of these, or even re-roll them all, and this two times. This means that he has two chances to enhance his score obtained after the first drawing. After that, he passes his turn to the next player. Therefore, in this game, the **decision actions** are how many dices to re-roll (0, 1, 2 or 3) and, in each of these cases, which dices to re-roll. Moreover, there are two sequential decisions since the player has the opportunity to re-roll the dices two times. Note that the **state** for the player corresponds to the combined state of the three dices (column 1 of Figure 9.3). Based on this state, the player has to decide which action has to be performed. Then after re-rolling (or not) the dices, we end up in a new state and the same applies after the (potential) second re-roll.

  Furthermore, after each full turn, the player with the **lowest score** (the loser for this turn) must take a given number of tokens from the player that scored the **highest** (the winner for this turn). Here again, this number is equal to the **largest obtained score** during this turn. The game goes on until someone gets all the tokens (11). In this case, he looses the game (and must pay a beer to the players).

Thus, the Markov decision process (MPD) must only be run during the second phase. In this example, we do not take into account the interactions among players, that is, the number of tokens of each player. The goal is to maximize the reward scores of the considered player, knowing that zero, one or two re-roll(s) are allowed and we can choose each time which dice to re-draw. There are 3×56 states for this game (the 56 states of column 1 of Figure 9.3 must be encoded three times: once for after the first roll, once for after the first re-roll and once for the second re-roll) plus a (virtual) starting state. Meanwhile, only 56 states from the $6 \times 6 \times 6$ permutations of the three dices are considered, as the order of the dices is not taken into account. In this 421 game, the **reward** for the MDP at the end of each turn is defined as the number of tokens you get rid of (transferred to the looser) if you are the winner.

9.5.2 Simulation results

The optimal policies obtained using the constrained randomized shortest paths framework (see Section 9.3.6) or using the soft value iteration algorithm (SVI, Equations (9.39)-(9.40)) are reported for a high θ on Table 9.1. We of course verified experimentally that the two procedures converge to exactly the same values and that both converge to the optimal, deterministic, policy when θ increases. Note that the states are denoted as follows. DDD in Table 9.1, column 1, indicates the results of the three dices ordered from the highest value to the lowest.

The optimal policy is then reported in columns 3 and 4 for the first and the second re-roll. Note that the strategy can be different during the first re-roll or the second re-roll (see, e.g., DDD = 543). Re-rolls are coded using three booleans (for instance, 011 means "re-roll dice 2 and 3 but not dice 1". The mean **reward**, defined as the obtained score after a turn, is 3.06 when playing the optimal policy. Conversely, when playing randomly and choosing the policy according to the reference probabilities, the mean reward is much lower, 1.61. It is important to have these values in mind when analyzing the results.

Interesting cases deserving some comments can be identified on Table 9.1:

- 421-like cases: Those are obvious cases (421, 111, 611, ...). If the player gets those high score combinations, the policy is 000; keep this high score, that is, do not re-roll.
- 532-like cases: The opposite of 421-like cases. If you get a bad combination (522, 322, 553, ...), the policy is 111, that is, just re-roll all dices. The expected value of re-rolling all dice is better than trying to keep interesting dices.
- 221-like cases: Those are the situations where it is advisable to keep some of the dices to get interesting combinations. A good example is 211: just roll the dice 2. Then,
  - If the dice scores 3, 4, 5, 6 or even 1, the number of points increases.
  - If the dice scores 2 the number of points stay the same.
- 543-like cases: Those are similar to the previous case but with a different strategy during the first and the second reroll. The expected score is 1.61 without re-rolls and 3.10 if re-rolls are allowed. 543 has a score of 2. During the first re-roll, the expected score is therefore higher (3.10), but become lower during the second (1.61). This difference explains why 543 has different mixed strategies according to the number of re-rolls.

Moreover, the optimal mixed policy obtained by the soft value iteration with intermediate values of θ is reported for the previous interesting cases on Figure 9.3. For cases 421, 532, 221-like, the strategy is very similar during the first or second re-roll. Notice that the increase of each bin is monotonic with the increase of θ and that the function interpolates between a uniform distribution (equal to priors, when θ is small) and the quasi-deterministic (sum of Kronecker delta, when θ is high) optimal policy.

Finally, Figure 9.2 represents the evolution of the average number of final tokens after the game (the higher, the better, reported as mean **reward** averaged over $10e6$ runs of whole turns) and the average entropy in function of θ. The largest expected reward and minimal entropy are achieved when θ is large and the opposite is true when θ is small. The resulting functions are both logistic-shaped between two bounds:

- When θ is small, entropy is maximum as each action has a $1/8$ probability to be chosen. The mean score is minimum and is the same as if we consider a random walk for this MDP. In this particular case, the expected score is the same as if no re-roll is allowed.
- When θ is large, entropy is minimum and the mixed policy converges to the optimal, deterministic, policy provided by the standard value iteration algorithm. The mean score is maximum and the corresponding policy induces re-rolls.

This example clearly shows that using a mixed strategy allows to balance the strength of the player.

9.6 Conclusion

This work presented two procedures for solving constrained randomized shortest path problems, together with an application to randomized Markov decision processes, where the problem is viewed as a bipartite graph. The main objective is to reach a destination node from an initial node in a graph while minimizing expected cost subject to a relative entropy equality constraint and transition probabilities constraints on some edges. This model provides a randomized policy encouraging exploration, balancing exploitation and exploration. The amount of exploration is monitored by an inverse temperature parameter.

The problem is expressed in terms of full paths connecting the initial node to the destination node and can easily be solved. The solution is a Gibbs-Boltzmann probability distribution on the set of paths with sone virtual costs associated to the constrained edges.

The first algorithm solves the problem by exploiting Lagrange duality and requires to solve iteratively the standard randomized shortest paths problem until convergence of the virtual costs.

The second algorithm is reminiscent of Bellman-Ford's algorithm for solving the shortest path distance problem. It simply aims to replace the min operator by a softmin operator in Bellman-Ford's recurrence relation for updating the unconstrained nodes.

The usefulness of these algorithms is then illustrated on standard Markov decision problems. Indeed, a standard Markov decision process can be reinterpreted as a randomized shortest paths problem on a bipartite graph. Standard Markov decision problems are thus easily solved by the two introduced algorithms: they provide a randomized policy minimizing expected cost under equality constraints.

This shows that the exploration strategy using the softmin instead of the min in the value iteration algorithm is optimal in the predefined sense. It therefore justifies from another, RSP, point of view the previous work [13, 14, 17, 18, 92, 137, 207, 234, 233, 237].

Future work will focus on extending the randomized shortest paths model in order to deal with other types of constraints. In particular we will work on inequality constraints on transition probabilities, as well as flow equality and inequality constraints, both on node flows and edge flows. Another interesting extension of the RSP model is the multi-sources multi-destinations randomized optimal transport on a graph generalizing the deterministic problem optimal transport on a graph problem.

Chapter 10

Conclusion

This thesis was devoted to network analysis, based on a framework called the bag-of-paths, which is introduced in Chapter 4 (Chapter 8 is not based on this framework, but implement concepts from Chapter 2, 3, and 5). We discuss different applications of the framework such as semi-supervised classification, criticality measures, and randomized policies for Markov decision processes, as the title of this work suggests.

This last chapter provides a summary of the present thesis and is structured as follows: Section 10.1 is a quick overview of the different chapters, Section 10.2 aims to summarize the contributions of these chapters, and Section 10.3 summarizes the limitations of the work. Finally, a last discussion in Section 10.4 concludes this chapter and the thesis.

10.1 Overview

Let us start by an overview, conveniently organized per chapter:

- **Chapter 1** is a general introduction followed by the plan of the work.
- **Chapter 2** and **Chapter 3** review the underlying graph and semi-supervised learning concepts, respectively.
- Then we introduce the bag-of-paths framework in **Chapter 4**, emphasizing on its mathematical derivation and its graph-based interpretation.
- In **Chapter 5**, a classifier based on a group betweenness defined within the bag-of-paths framework is derived for graph-based semi-supervised classification. This approach outperforms seven state-of-the-art graph-based classifiers, based on 14 datasets.
- In **Chapter 6**, 16 semi-supervised classification methods are investigated to compare the feature-based approach, the graph structure based approach, and a dual approach combining both information sources. From those 16 classifiers, two are based on the bag-of-paths framework.

- In **Chapter 7**, a graph criticality measure based on the bag-of-paths framework (and a faster approximate version) is derived and compared to 12 competing measures. Here again, our approach is competitive in comparison with the other investigated state-of-the-art methods.
- In **Chapter 8**, a graph-based fraud detection system, called APATE, is improved using the concepts presented in Chapters 2, 3, and 5. It allows to efficiently identify frauds among a real-life database consisting of millions of credit-card transactions.
- In **Chapter 9**, two methods tackling Markov decision problems are presented and compared through one example. The originality comes from the fact that the output of those methods (based on the bag-of-paths framework) are mixed, randomized, strategies.

10.2 Contributions

Contributions for the original chapters (5-8) are summarized below.

- In **Chapter 5**, a graph-based semi-supervised classification algorithm based on a group betweenness measure is developed. All computational steps are computable in closed form. Furthermore, 13 classifications datasets are identified and referenced for further research and all the Matlab code is made available on internet.
- In **Chapter 6**, the problem of multiple data sources (graph and plain features) is addressed. This problem is complex because nothing tells a priori how this information should be handled. The chapter investigates multiple interesting questions: How to combine both information sources ? Is one of of the information source better than the other? Is it always the case ? What about dimensionality reduction (a large topic in machine learning) ? How should we proceed in practice ? We used an extensive study to draw some general conclusions and advices to answer those complex questions. Here again, 10 classification datasets are identified and referenced for further research and all the Matlab code is available from internet.
- In **Chapter 7**, the criticality measure of a graph, or the identification of the most critical nodes is investigated. We develop a new criticality measure based on the bag-of-paths framework. This measure (and its faster approximation) is compared to 12 competing measures. Finally, an objective criterion to assess the efficiency of criticality measures is investigated, based on the generation of a large number of synthetic and real graphs.

- In **Chapter 8**, we proposed three major improvements to an existing algorithm named APATE [241]. First, the highly connected nodes (hubs) harmful effect is damped. Second, semi-supervised learning is introduced to increase the performance of the model. Finally, human feedback is used to further improve the results.
- **Chapter 9**, This work extends the randomized shortest path framework (RSP, close to the bag-of-paths framework) in two directions. First, it shows how to deal with equality constraints on transition probabilities and derives a generic algorithm for solving the problem. Second, it derives a simple algorithm to compute the optimal, mixed, policy solving Markov decision processes (MDP) by considering a constrained RSP on a bipartite graph. The resulting algorithm allows to balance exploitation and exploration in an optimal way by interpolating between a pure random behavior and the optimal, deterministic, policy. It is also shown that the mixed policy can be obtained by iterating the Bellman's value iteration equations in which the minimum operator is replaced by a soft minimum.

10.3 Limitations

The limitations of the original chapters are summarized below. Here again, Chapters 1, 2, 3, 4, and 10 are not present in this section.

- In **Chapter 5**, the biggest drawback of the BoP classifier is that it is not computationally tractable for large graphs. Investigations have been carried in that direction but no convincing solution has been found. The limiting step is the computation of matrix $\mathbf{Z}$ (the fundamental matrix), which requires a full matrix inversion.
- In **Chapter 6**, a key issue is to determine a priori if a given dataset is graph-driven or features-driven, since this information is important in order to choose the most efficient algorithm combining graph and plain features information. Spatial autocorrelation coefficients can help to tackle this issue but cannot be used in practice as they assume that all class labels are known. How can we relax this hypothesis ? We still have no answer at this point. Furthermore, no scalability effect has been studied.
- In **Chapter 7**, this study should be confirmed on larger networks. Here again, the limiting step is the computation of matrix $\mathbf{Z}$ (the fundamental matrix), which require a full matrix inversion. At that point, no convincing solution has been found.

- In **Chapter 8**, the proposed system is a very domain-specific application and the algorithm could certainly be modified to tackle other scenarios. By the way, the main point is that semi-supervised learning, hubs damping in graphs and utilization of feedback can improve performance of generic algorithms. In this chapter, the problem of scalability has been a major issue, but at the end it is possible to deal with millions of transactions in a few minutes.
- **Chapter 9**, presented two procedures for solving randomized Markov decision processes (MDP):
  - The first one main drawback is that it is computationally demanding: it relies on iterative algorithms that must solve a linear system of equations at each iteration.
  - The second procedure is more time-efficient as it is based on a Bellman-Ford-like algorithm: the soft value iteration. It could scale on large graphs by using sparse matrices.

10.4 Final discussion

As stated in Chapter 1, the presented work of this thesis is at the center of a technological revolution. It aims at proposing and developing applications of the bag-of-paths framework in the context of networks analysis. In this framework, the Boltzmann sampling distribution depends on a parameter, θ, gradually biasing the distribution towards shorter paths when θ is large. Then, only little exploration is performed and only the shortest paths are considered. On the other hand, when θ is small (close to 0+), longer paths are considered and are sampled according to the product of the random walk transition probabilities along the path.

In this thesis, the bag-of-paths framework was applied in many problems, but many more are possible. By the way, it seems more adapted to cases where data can be expressed (or gathered) as graphs, and when the shortest path and random walk approaches fail to produce consistently good results. In that case, using the bag-of-paths, and its underlying interpolation, can actually boost the performance.

However, this interpolation comes with an increasing computational cost (discussed in Section 10.3). In some cases, this cost leads to intractability (as in Chapter 8).

Shortest paths (and to a lesser extent random walks) lead to great, efficient implementation that the bag-of-paths cannot compete with in terms of running time.

We are certain that the bag-of-paths framework can be the basis for many other techniques, as the various experimental results have already shown.

Appendix A

The Friedman/Nemenyi test

Despite widely used to compare classifiers, the paired t-test suffers from three weaknesses [74]: variables must be (i) commensurable, (ii) normally distributed and (iii) the test is affected by outliers. The Friedman test (and Nemenyi post-hoc test) does not exhibits those weaknesses.

Therefore, the Friedman/Nemenyi test is used throughout this thesis to compare multiple classifiers (or more generally, models) across multiple datasets. The Friedman test provide evidence that the results of the different classifiers are significantly different (see Section A.1). Then the Nemenyi Post-hoc test assesses significant superiority/inferiority of each classifiers among the others (see Section A.2). This Appendix is widely inspired from the excellent work of [74]. In this context, the classifiers are called methods and a performance criterion must be chosen (for instance the accuracy, AUC,...)

A.1 The Friedman test

The Friedman test [98, 97] is a non-parametric equivalent of the repeated-measures ANOVA [88]. It ranks the methods for each dataset separately, best algorithm getting rank m, the second best rank $m-1$, etc (In case of ties, average ranks are assigned).

Let r_{ij} be the rank for the ith dataset (out of N) and the jth method (out of k). The Friedman test first computes the average rank of each algorithm, $R_j = \frac{1}{N}\sum_{i=1}^{N} r_{ij}$. The null hypothesis, states that all the algorithms are equivalent and so that their average ranks R_j should be equal. The alternative hypothesis states the opposite. Under the null hypothesis, the Friedman statistics

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_{j=1}^{N} R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{A.1}$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom, when N and k are big enough (a rule of a thumb is N > 10 and k > 5, which is always the case in this thesis). Notice that in this thesis all Friedman null hypothesis were (by far) rejected with p-value< 0.05.

A.2 The Nemenyi post-hoc test

Once Friedman null hypothesis is rejected, the (non parametric) post-hoc Nemenyi test can be computed. The Nemenyi test [99] is similar to the Tukey test for ANOVA [240] and is used when all classifiers must be compared to each other.

The performance of two classifiers is significantly different (p-value$= 0.05$) if the corresponding average ranks differ by at least the critical difference CD:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{A.2}$$

where critical value q_α is based on the Studentized range statistic divided by $\sqrt{2}$. Notice that the Nemenyi post-hoc test is not very powerful due to the square root of N, which means that many different datasets are required when the number of methods increases.

In this thesis, we therefore report the result of this test with a plot with the mean rank of each method plus and minus the corresponding critical difference CD.

Appendix B

Proof of main results of Chapter 9

This chapter compiles mathematical proofs that were moved here to facilitate the reading of Chapter 9. In particular, it contains the derivation of the soft value iteration algorithm.

In order to compute the optimal policy p^*_{ku}, we observe from Equation (9.11) that we need to find the backward variable z_{un} starting from an action $u \in \mathcal{A}$,

$$p^*_{ku} \propto p^{\text{ref}}_{ku} \exp[-\theta c_{ku}] z_{un}$$

in our bipartite graph. The quantity p^*_{ku} then needs to be normalized so that $\sum_{u \in \mathcal{U}(k)} p^*_{ku} = 1$.

B.1 Computation of the backward variable on action nodes

Now, from the definition of the backward variable (Equation (9.8), but including the augmented costs), we obtain by decomposing the paths $u \rightsquigarrow n (u \in \mathcal{A})$ into the first step $u \to l$, and then the remaining steps $l \rightsquigarrow n$ (see [101] for a related computation)

$$
\begin{aligned}
z_{un} &= \sum_{\wp_{un} \in \mathcal{P}_{un}} \tilde{\pi}(\wp_{un}) \exp[-\theta \tilde{c}'(\wp_{un})] \\
&= \sum_{l \in \mathcal{S}ucc(u)} \sum_{\wp_{ln} \in \mathcal{P}_{ln}} p^{\text{ref}}_{ul} \tilde{\pi}(\wp_{ln}) \exp[-\theta(c'_{ul} + \tilde{c}(\wp_{ln}))] \\
&= \sum_{l \in \mathcal{S}ucc(u)} p^{\text{ref}}_{ul} \exp[-\theta c'_{ul}] \underbrace{\sum_{\wp_{ln} \in \mathcal{P}_{ln}} \tilde{\pi}(\wp_{ln}) \exp[-\theta \tilde{c}(\wp_{ln})]}_{z_{ln}} \\
&= \sum_{l \in \mathcal{S}ucc(u)} p^{\text{ref}}_{ul} \exp[-\theta c'_{ul}] z_{ln} \qquad \text{(B.1)}
\end{aligned}
$$

where $\wp_{un}$ is a path starting in action state u and ending in the killing absorbing state n. Note that we have to use the augmented costs c'_{ul} in order to ensure that the flow in the edge (u, l) is equal to the predefined transition probability p^{ref}_{ul} provided by the environment. The value of these augmented costs can be found in Equation (9.34) and will be adapted to our bipartite MDP graph later. This provides

$$
c'_{ul} = \tfrac{1}{\theta} \log z_{ln} - \tfrac{1}{\theta} \sum_{l' \in \mathcal{S}ucc(u)} p^{\text{ref}}_{ul'} \log z_{l'n} + \sum_{l' \in \mathcal{S}ucc(u)} p^{\text{ref}}_{ul'} c_{ul'} \qquad \text{(B.2)}
$$

Injecting this result in Equation (B.1) yields

$$
\begin{aligned}
&z_{un} \qquad \text{(B.3)} \\
&= \sum_{l \in \mathcal{S}(u)} p^{\text{ref}}_{ul} \exp[-\theta c'_{ul}] z_{ln} \\
&= \sum_{l \in \mathcal{S}(u)} p^{\text{ref}}_{ul} \exp[-\log z_{ln}] \exp\Bigg[\sum_{l' \in \mathcal{S}(u)} p^{\text{ref}}_{ul'} \log z_{l'n}\Bigg] \exp\Bigg[-\theta \sum_{l' \in \mathcal{S}(u)} p^{\text{ref}}_{ul'} c_{ul'}\Bigg] z_{ln} \\
&= \sum_{l \in \mathcal{S}(u)} p^{\text{ref}}_{ul} \exp\Bigg[\sum_{l' \in \mathcal{S}(u)} p^{\text{ref}}_{ul'} \log z_{l'n}\Bigg] \exp\Bigg[-\theta \sum_{l' \in \mathcal{S}(u)} p^{\text{ref}}_{ul'} c_{ul'}\Bigg] \\
&= \exp\Bigg[\sum_{l \in \mathcal{S}(u)} p^{\text{ref}}_{ul} \log z_{ln}\Bigg] \exp\Bigg[-\theta \sum_{l' \in \mathcal{S}(u)} p^{\text{ref}}_{ul'} c_{ul'}\Bigg] \qquad \text{(B.4)}
\end{aligned}
$$

where $\mathcal{S}(u)$ has been used as a shortcut for $\mathcal{S}ucc(u)$.

Now, in the case of our bipartite MDP graph, the original costs c_{ul} are equal to zero for transitions between action nodes and state nodes, as specified in

Equation (9.36). Therefore,

$$z_{un} = \exp\left[\sum_{l \in \mathcal{S}ucc(u)} p_{ul}^{\text{ref}} \log z_{ln}\right] \tag{B.5}$$

which is the expression for computing the backward variables associated to actions. This expression depends on z_{ln}, which will be derived in Subsection B.3.

B.2 Computation of the optimal policy

Let us now replace the value of z_{un}, just obtained in the previous section, in the equation providing the optimal policy (Equation (B.1)),

$$\begin{aligned}
p_{ku}^{*} &\propto p_{ku}^{\text{ref}} \exp[-\theta c_{ku}] z_{un} \\
&= p_{ku}^{\text{ref}} \exp[-\theta c_{ku}] \exp\left[\sum_{l \in \mathcal{S}ucc(u)} p_{ul}^{\text{ref}} \log z_{ln}\right] \\
&= p_{ku}^{\text{ref}} \exp\left[-\theta\Big(c_{ku} + \sum_{l \in \mathcal{S}ucc(u)} p_{ul}^{\text{ref}} (-\tfrac{1}{\theta} \log z_{ln})\Big)\right] \\
&= p_{ku}^{\text{ref}} \exp\left[-\theta\Big(c_{ku} + \sum_{l \in \mathcal{S}ucc(u)} p_{ul}^{\text{ref}} \phi_{l}^{\mathcal{S}}(T)\Big)\right] \qquad \text{(B.6)}
\end{aligned}$$

which justifies Equation (9.40) after normalization. In the last step, we introduced the free energy defined in Subsection 9.4.2, $\phi_{k}^{\mathcal{S}}(T) = -T \log z_{kn} = -\frac{1}{\theta} \log z_{kn}$. Because the free energy, and thus the optimal policy, depends on the backward variable defined on states, z_{kn} with $k \in \mathcal{S}$, let us now turn to the computation of this quantity.

B.3 Computation of the backward variable on states

By proceeding as for the derivation of the recurrence relation of Equation (B.1), we obtain

$$\begin{aligned}
z_{kn} &= \sum_{\wp_{kn} \in \mathcal{P}_{kn}} \tilde{\pi}(\wp_{kn}) \exp[-\theta \tilde{c}(\wp_{kn})] \\
&= \sum_{u \in \mathcal{U}(k)} p_{ku}^{\text{ref}} \exp[-\theta c_{ku}] z_{un} \qquad \text{(B.7)}
\end{aligned}$$

and this last result only depends on the original costs c_{ku} (and not the augmented costs c'_{ku}) because there is no augmented cost associated to the transitions from a state to an action node, as these transitions are not part of the set of constrained transitions (see Equation (9.36)).

B.4 The soft value iteration recurrence equation

Taking $-\frac{1}{\theta}\log$ of each side of this Equation (B.7), using $\phi_k^{\mathcal{S}}(T) = -\frac{1}{\theta}\log z_{kn}$, and replacing z_{un} by its value provided in Equation (B.5) gives

$$\begin{aligned}
\phi_k^{\mathcal{S}}(T) &= -\tfrac{1}{\theta}\log\Bigg[\sum_{u\in\mathcal{U}(k)} p_{ku}^{\text{ref}}\exp[-\theta c_{ku}]z_{un}\Bigg] \\
&= -\tfrac{1}{\theta}\log\Bigg[\sum_{u\in\mathcal{U}(k)} p_{ku}^{\text{ref}}\exp[-\theta c_{ku}]\;\exp\Big[\sum_{l\in\mathcal{S}ucc(u)} p_{ul}^{\text{ref}}\log z_{ln}\Big]\Bigg] \\
&= -\tfrac{1}{\theta}\log\Bigg[\sum_{u\in\mathcal{U}(k)} p_{ku}^{\text{ref}}\exp\Big[-\theta\Big(c_{ku} + \sum_{l\in\mathcal{S}ucc(u)} p_{ul}^{\text{ref}}(-\tfrac{1}{\theta}\log z_{ln})\Big)\Big]\Bigg] \\
&= -\tfrac{1}{\theta}\log\Bigg[\sum_{u\in\mathcal{U}(k)} p_{ku}^{\text{ref}}\exp\Big[-\theta\Big(c_{ku} + \sum_{l\in\mathcal{S}ucc(u)} p_{ul}^{\text{ref}}\phi_l^{\mathcal{S}}(T)\Big)\Big]\Bigg]
\end{aligned} \tag{B.8}$$

which is exactly the sought result of Equation (9.38), together with the fact that when $k = n$ (absorbing, killing, state), $z_{nn} = 1$ (see Equation (9.8)) and thus $\phi_n^{\mathcal{S}}(T) = 0$. Notice that the $\phi_k^{\mathcal{S}}(T)$ are necessarily non-negative. This equation is a smooth approximation of the standard value iteration algorithm through the softmin operator (Equation (9.17)) – the soft value iteration – that has to be iterated on all nodes until convergence.

Bibliography

[1] A. Abdallah, M. Maarof, and A. Zainal. "Fraud detection system". In: *Journal of Network and Computer Applications* 68 (2016), pp. 90–113.

[2] S. Abney. *Semisupervised Learning for Computational Linguistics*. Chapman and Hall/CRC, 2008.

[3] Y. Achbany et al. "Optimal Tuning of Continual Exploration in Reinforcement Learning". In: *Proceedings of the 16th International Conference on Artificial Neural Networks (ICANN 06). Lecture notes in Computer Science* LNCS 4131 (2006), pp. 734–749.

[4] G. Agnarsson and R. Greenlaw. *Graph theory: Modeling, Applications, and Algorithms*. Pearson, 2007.

[5] A. Agrawala. "Learning with a probabilistic teacher". In: *IEEE Transactions on Information Theory* 16.4 (July 1970), pp. 373–379.

[6] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[7] T. Akamatsu. "Cyclic flows, Markov process and stochastic traffic assignment". In: *Transportation Research B* 30.5 (1996), pp. 369–386.

[8] M. Alamgir and U. von Luxburg. "Phase transition in the family of p-resistances". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 379–387.

[9] R. Albert, H. Jeong, and A. Barabasi. "Error and attack tolerance of complex networks". In: *Nature* 406.6794 (2000), pp. 378–382.

[10] L. Anselin. *Spatial Econometrics: Methods and Models*. Kluwer academic press, 1988.

[11] A. Arenas et al. "Motif-based communities in complex networks". In: *Journal of Physics A: Mathematical and Theoretical* 41.22 (2008), p. 224001.

[12] K. Arrow, L. Hurwicz, and H. Uzawa. *Studies in Linear and Non-linear Programming*. Stanford University Press, 1958.

[13] K. Asadi and M. Littman. "A new softmax operator for reinforcement learning". In: *ArXiv preprint arXiv:1612.05628* (2016).

[14] K. Asadi and M. Littman. “A new softmax operator for reinforcement learning”. In: *Proceedings of the International Conference on Machine Learning (ICML2017)*. 2017, pp. 243–252.

[15] Association of Certified Fraud Examiners. *Report to the nation*. 2002. URL: `http://www.acfe.com/uploadedfiles/acfe_website/content/documents/2002rttn.pdf` (visited on 06/02/2016).

[16] N. Augustin, M. Mugglestone, and S. Buckland. “The role of simulation in modelling spatially correlated data”. In: *Environmetrics* 9.2 (1998), pp. 175–196.

[17] M. Azar, V. Gómez, and B. Kappen. “Dynamic policy programming with function approximation”. In: *International Conference on Artificial Intelligence and Statistics*. 2011, pp. 119–127.

[18] M. Azar, V. Gómez, and B. Kappen. “Dynamic policy programming”. In: *Journal of Machine Learning Research* 13.Nov (2012), pp. 3207–3245.

[19] B. Baesens, V. Van Vlasselaer, and W. Verbeke. *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection*. Wiley Publishing, 2015.

[20] A. Bahnsen et al. “Cost Sensitive Credit Card Fraud Detection Using Bayes Minimum Risk”. In: *12th International Conference on Machine Learning and Applications, ICMLA 2013, Miami, FL, USA, December 4-7, 2013, Volume 1*. 2013, pp. 333–338.

[21] A. Bahnsen et al. “Example-Dependent Cost-Sensitive Logistic Regression for Credit Scoring”. In: *13th International Conference on Machine Learning and Applications, ICMLA 2014, Detroit, MI, USA, December 3-6, 2014*. 2014, pp. 263–269.

[22] A. Bahnsen et al. “Feature engineering strategies for credit card fraud detection”. In: *Expert Systystem with Applications* 51 (2016), pp. 134–142.

[23] R. Bapat. *Graphs and Matrices*. Springer-Verlag, 2010.

[24] A. Barabasi and R. Albert. “Emergence of scaling in random networks”. In: *Science* 286.5439 (1999), pp. 509–512.

[25] A. Barabásiand and M.Posfai. *Network Science*. Cambridge University Press, 2016.

[26] Vladimir Batagelj and Andrej Mrvar. *Pajek datasets*. 2006. URL: `http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/ucidata.htm` (visited on 09/07/2017).

[27] M. Belkin, I. Matveeva, and P. Niyogi. “Regularization and semi-supervised learning on large graphs”. In: *Proceedings of the International Conference on Learning Theory (COLT 2004)*. 2004, pp. 624–638.

[28] M. Belkin, P. Niyogi, and V. Sindhwani. "Manifold regularization: a geometric framework for learning from examples". In: *Journal of Machine Learning Research* 7 (2006), pp. 2399–2434.

[29] H. Benali and B. Escofier. "Analyse factorielle lissee et analyse des differences locales". In: *Revue de Statistique Appliquee* 38.2 (1990), pp. 55–76.

[30] Y. Bengio, O. Delalleau, and N. Le Roux. "Label propagation and quadratic criterion". In: *Semi-supervised learning, O. Chapelle, B. Scholkopf and A. Zien (editors)*. MIT Press, 2006, pp. 193–216.

[31] B. Benjamin. "Randomized Markov decision processes: a study of two new algorithms". Promotor: Prof. Marco Saerens. MA thesis. Universite de Louvain, 2013.

[32] D. Bertsekas. *Neuro-dynamic Programming*. Athena Scientific, 1996.

[33] D. Bertsekas. *Network Optimization: Continuous and Discrete models*. Athena Scientific, 1998.

[34] D. Bertsekas. *Dynamic Programming and Optimal Control, 2nd ed*. Athena Scientific, 2000.

[35] Dimitri P. Bertsekas. *Nonlinear Programming, 2nd ed.* Athena Scientific, 1999.

[36] J. Besag. "Nearest-neighbour systems and the auto-logistic model for binary data". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 34.1 (1972), pp. 75–83.

[37] S. Bhattacharyya et al. "Data Mining for Credit Card Fraud: A Comparative Study". In: *Decision Support Systems* 50.3 (Feb. 2011), pp. 602–613.

[38] L. Biggs, E. Lloyd, and J. Wilson. *Graph Theory 1736-1936*. Oxford University Press, 1976.

[39] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[40] R. Bivand, E. Pebesma, and V. Gomez-Rubio. *Applied Spatial Data Analysis with R*. Springer, 2008.

[41] A. Blum and T. Mitchell. "Combining Labeled and Unlabeled Data with Co-training". In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. COLT' 98. New York, NY, USA: ACM, 1998, pp. 92–100.

[42] B. Bollobas. *Modern Graph Theory*. Springer, 1998.

[43] B. Bollobas. *Random Graphs*. Cambridge University Press, 2001.

[44] R. Bolton and D. Hand. "Unsupervised Profiling Methods for Fraud Detection". In: *Proceedings of the Credit Scoring and Credit Control VII Conference*. 2001, 235–255.

[45] D. Borcard and P. Legendre. "All-scale spatial analysis of ecological data by means of principal coordinates of neighbour matrices". In: *Ecological Modelling* 153.1-2 (2002), pp. 51 –68.

[46] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[47] E. Bozzo and M. Franceschet. "Resistance distance, closeness, and betweenness". In: *Social Networks* 35.3 (2013), pp. 460–469.

[48] M. Brand and K. Huang. "A unifying theorem for spectral embedding and clustering". In: *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*. Key West, FL, 2003.

[49] U. Brandes. "A faster algorithm for betweenness centrality". In: *The Journal of Mathematical Sociology* 25.2 (2001), pp. 163–177.

[50] U. Brandes and T. Erlebach. *Network Analysis: Methodological Foundations*. Springer-Verlag, 2005.

[51] U. Brandes and D. Fleischer. "Centrality Measures Based on Current Flow". In: *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*. 2005, pp. 533–544.

[52] F. Braun et al. "Improving Card Fraud Detection through Suspicious Pattern Discovery". In: *Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. 2017, pp. 181–190.

[53] P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer-Verlag, 1999.

[54] J. Callut and P. Dupont. "Learning hidden Markov models from first passage times". In: *Proceedings of the European Machine Learning conference (ECML). Lecture notes in Artificial Intelligence, Springer* (2007).

[55] J. Callut et al. "Semi-supervised classification from discriminative randow walks". In: *Proceedings of the European Machine Learning conference (ECML 2008). Lecture notes in Artificial Intelligence, Springer* 5211 (2008), pp. 162–177.

[56] S. Chakrabarti, B. Dom, and P. Indyk. "Enhanced hypertext categorization using hyperlinks". In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD 1998)*. 1998, pp. 307–318.

[57] A. Chandra et al. "The Electrical Resistance of a Graph Captures its Commute and Cover Times". In: *Annual ACM Symposium on Theory of Computing* (1989), pp. 574–586.

[58] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[59] O. Chapelle, J. Weston, and B. Scholkopf. "Cluster kernels for semi-super-vised learning". In: *NIPS, 2002* (2002), pp. 585–592.

[60] P. Chebotarev. "A class of graph-geodetic distances generalizing the shortest-path and the resistance distances". In: *Discrete Applied Mathematics* 159.5 (2011), pp. 295–302.

[61] D. Chen and X. Cheng. "An asymptotic analysis of some expert fusion methods". In: *Pattern Recognition Letters* 22 (2001), pp. 901–904.

[62] N. Christofides. *Graph Theory: An Algorithmic Approach*. Academic Press, 1975.

[63] F. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[64] J. Cook. "Basic properties of the soft maximum". Unpublished manuscript available from www.johndcook.com/blog/2010/01/13/soft-maximum. 2011.

[65] Roger Cooke. *Experts in Uncertainty*. Oxford University Press, 1991.

[66] T. Cormen et al. *Introduction to Algorithms, 3th Edition*. The MIT Press, 2009.

[67] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory, 2nd edition*. John Wiley & Sons, 2006.

[68] N. Cressie. *Statistics for Spatial Data, revised edition*. Wiley, 1993.

[69] J. Culioli. *Introduction a l'Aptimisation*. Ellipses, 2012.

[70] A. Dal Pozzolo. "Adaptive Machine Learning for Credit Card Fraud Detection". PhD thesis. Universite Libre de Bruxelles, 2015.

[71] A. Dal Pozzolo et al. "Learned lessons in credit card fraud detection from a practitioner perspective". In: *Expert System with Applications* 10.41 (2014), pp. 4915–4928.

[72] A. Dal Pozzolo et al. "Credit card fraud detection and concept-drift adaptation with delayed supervised information." In: *Proceedings of the International Joint Conference on Neural Networks*. IEEE, 2015, pp. 1–8.

[73] A. Dempster, N. Laird, and D. Rubin. "Maximum likelihood from incomplete data via the EM algorithm (with discussion)". In: *Journal of the Royal Statistical Society B* 39.1 (1977), pp. 1–38.

[74] J. Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *Journal pf Machine Learning Research* 7 (Dec. 2006), pp. 1–30.

[75] R. Devooght et al. "Random Walks Based Modularity: Application to Semi-supervised Learning". In: *Proceedings of the 23rd International Conference on World Wide Web*. WWW '14 (2014), pp. 213–224.

[76] S. Dorogovtsev. *Lectures on Complex Networks*. New York, NY, USA: Oxford University Press, Inc., 2010.

[77] P. Doyle. "The Kemeny constant of a Markov chain". unpublished manu-script. 2009. URL: `http://www.math.dartmouth.edu/~doyle`.

[78] P. Doyle and L. Snell. *Random Walks and Electric Networks*. The Mathematical Association of America, 1984.

[79] S. Dray, P. Legendre, and P. Peres-Neto. "Spatial modelling: a comprehensive framework for principal coordinate analysis of neighbour matrices (PCNM)". In: *Ecological Modelling* 196.3-4 (2006), pp. 483 –493.

[80] D. Dubois et al. "Assessing the value of a candidate: Comparing belief function and possibility theories". In: *Proceedings of the Fifteenth international conference on Uncertainty in Artificial Intelligence* (1999), pp. 170–177.

[81] E-commerce Europe. *Global B2C E-commerce LIGHT Report 2015*. 2014. URL: `https://www.ecommerce-europe.eu/facts-figures/free-light-reports` (visited on 05/25/2016).

[82] E. Estrada. "Network robustness to targeted attacks. The interplay of expansibility and degree distribution". In: *The European Physical Journal B - Condensed Matter and Complex Systems* 52.4 (2006), pp. 563–574.

[83] E. Estrada, D. Higham, and N. Hatano. "Communicability betweenness in complex networks". In: *Physica A: Statistical Mechanics and its Applications* 388.5 (2009), pp. 764 –774.

[84] L. Euler. "Solutio Problematis ad Geometriam Situs Pertinentis". In: *Opera Omnia* 7 (1736), pp. 128–140.

[85] R. Fan et al. "LIBLINEAR: A Library for Large Linear Classification". In: *Journal of Machine Learning Research* 9 (2008), pp. 1871–1874.

[86] T. Fawcett and F. Provost. "Adaptive Fraud Detection". In: *Data Mining and Knowledge Discovery* 1 (1997), pp. 291–316.

[87] T. Fawcett and F. Provost. "Activity Monitoring: Noticing Interesting Changes in Behavior". In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '99. San Diego, California, USA: ACM, 1999, pp. 53–62.

[88] R. Fisher. *Statistical Methods and Scientific Inference*. 2nd edition. Hafner Publishing Co., New York, 1959.

[89] F. Fouss and M. Saerens. "Yet another method for combining classifiers outputs: A maximum entropy approach". In: *Proceedings of the 5th International Workshop on Multiple Classifier Systems (MCS 2004), Lecture Notes in Computer Science, Vol. 3077, Springer-Verlag* (2004), pp. 82–91.

[90] F. Fouss, M. Saerens, and M. Shimbo. *Algorithms and Models for Network Data and Link Analysis*. Cambridge University Press, 2016.

[91] F. Fouss et al. "An experimental investigation of kernels on a graph on collaborative recommendation and semisupervised classification". In: *Neural Networks* 31 (2012), pp. 53–72.

[92] R. Fox, A. Pakman, and N. Tishby. "G-learning: taming the noise in reinforcement learning via soft updates". In: *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI 2016)*. 2001, pp. 202–211.

[93] S. Fralick. "Learning to recognize patterns without a teacher". In: *IEEE Transactions on Information Theory* 13.1 (Jan. 1967), pp. 57–64.

[94] K. Françoisse et al. "A bag-of-paths framework for network data analysis". In: *Neural Networks* 90 (2017), pp. 90–111.

[95] K. Françoisse et al. "A bag-of-paths framework for network data analysis". In: *ArXiv preprint arXiv:1302.6766* (2013).

[96] L. Freeman. "A set of measures of centrality based on betweenness". In: *Sociometry* 40.1 (1977), pp. 35–41.

[97] M. Friedman. "A comparison of alternative tests of significance for the problem of m rankings". In: *Annals of Mathematical Statistics* 11 (1940), pp. 86–92.

[98] M. Friedman. "The use of ranks to avoid the assumption of normality implicit in the analysis of variance". In: *Journal of the American Statistical Association* 32 (1940), pp. 675–701.

[99] P. Friedman. "Distribution-free Multiple Comparisons". PhD thesis. Princeton University, 1963.

[100] A. Gammerman, V. Vapnik, and V. Vowk. "Learning by tranduction". In: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. Wisconsin, 1998, pp. 273–297.

[101] Si. García-Díez et al. "A sum-over-paths extension of edit distances accounting for all sequence alignments". In: *Pattern Recognition* 44.6 (2011), pp. 1172–1182.

[102] Thomas Gartner. *Kernels for Structured Data*. World Scientific Publishing, 2008.

[103] R. Geary. "The contiguity ratio and statistical mapping". In: *The Incorporated Statistician* 5.3 (1954), pp. 115–146.

[104] L. Getoor and B. Taskar, eds. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

[105] L. Getoor and B. Taskar, eds. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

[106] D. Gleich. *MatlabBGL package*. 2008. URL: `https://www.mathworks.com/matlabcentral/fileexchange/10922-matlabbgl` (visited on 10/24/2017).

[107] D. Gleich. *Pearson higher eduction*. 2009. URL: `https://media.pearsoncmg.com/bc/abp/engineering-resources/products/product.html#product,isbn=0131471392` (visited on 11/28/2017).

[108] F. Gobel and A. Jagers. "Random walks on graphs". In: *Stochastic Processes and their Applications* 2 (1974), pp. 311–336.

[109] G. Golub and C. Van Loan. *Matrix Computations, 3th Ed.* The Johns Hopkins University Press, 1996.

[110] M. Gondran and Michel M. Minoux. *Graphs and Algorithms*. John Wiley & Sons, Inc., 1984.

[111] C. Grinstead and J. Snell. *Introduction to Probability, 2nd ed.* The Mathematical Association of America, 1997.

[112] I. Griva, S. Nash, and A. Sofer. *Linear and Nonlinear Optimization*. 2nd. SIAM, 2008.

[113] R. Haining. *Spatial Data Analysis*. Cambridge University Press, 2003.

[114] K. Hara et al. "Localized Centering: Reducing Hubness in Large-Sample Data." In: *Proceedings of the Association for the Advancement of Artificial Intelligence Conference*. 2015, pp. 2645–2651.

[115] F. Harary. *Graph Theory*. Addison-Wesley, 1969.

[116] D. Hardoon, S. Szedmak, and J.Shawe-taylor. "Canonical Correlation Analysis: An Overview with Application to Learning Methods". In: *Neural Comput.* 16.12 (Dec. 2004), pp. 2639–2664.

[117] S. Haykin. *Neural Networks and Learning Machines*. Third edition. Pearson Education, 2009.

[118] X. He. "Laplacian regularized D-optimal design for active learning and its application to image retrieval". In: *IEEE Transactions on Image Processing* 19.1 (2010), pp. 254–263.

[119] M. Herbster and G. Lever. "Predicting the Labelling of a Graph via Minimum p-Seminorm Interpolation". In: *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*. 2009.

[120] M. Herbster, M. Pontil, and S. Rojas-Galeano. "Fast prediction on tree". In: *Proceedings of the 22th Neural Information Processing Conference NIPS 2008* (2008), pp. 657–664.

[121] S. Hill, F. Provost, and C. Volinsky. "Network-Based Marketing: Identifying Likely Adopters via Consumer Networks". In: *Statistical Science* 21.2 (2006), pp. 256–276.

[122] T. Hofmann, B. Schölkopf, and A. Smola. "Kernel Methods in Machine Learning". In: *The Annals of Statistics* 36.3 (2008), pp. 1171–1220.

[123] P. Holme et al. "Attack vulnerability of complex networks". In: *Physic Review E* 65.5 (May 2002), p. 056109.

[124] HSN Consultants, Inc. *The Nilson Report*. 2015. URL: `https://www.nilsonreport.com/upload/issues/1085_1008.pdf` (visited on 08/09/2016).

[125] C. Hsu and C. Lin. "A Comparison of Methods for Multiclass Support Vector Machines". In: *Transaction on Neural Network* 13.2 (Mar. 2002), pp. 415–425.

[126] T. Jaakkola, M. Meila, and T. Jebara. "Maximum entropy discrimination". In: *Advances in Neural Information Processing Systems 16 (NIPS 2000)*. MIT Press, 2000, pp. 470–476.

[127] Robert Jacobs. "Methods For Combining Experts' Probability Assessments". In: *Neural Computation* 7 (1995), pp. 867–888.

[128] P. Jakma. *Barabasi-albert preferential attachment and the internet*. 2013. URL: `https://paul.jakma.org/2013/12/02` (visited on 10/24/2017).

[129] A. Jamakovic and S.Uhlig. "On the relationship between the algebraic connectivity and graph's robustness to node and link failures". In: *Proceedings of the 3rd IEEE Conference on Next Generation Internet Networks (EuroNGI '07)*. IEEE, 2007, pp. 96–102.

[130] E. Jaynes. "Information theory and statistical mechanics". In: *Physical Review* 106 (1957), pp. 620–630.

[131] T. Jebara. *Machine Learning, Discriminative and Generative*. Kluwer Academic Publishers, 2004.

[132] T. Joachims. "Transductive Learning via Spectral Graph Partitioning". In: *Proceedings of the 20th International Conference on Machine Learning (ICDM 2003)*. Washington DC, 2003, pp. 290–297.

[133] J. Jones. *421 ... A French dicegame. Materials. Rules.* 2014. URL: `https://plazilla.com/page/4295153846/421-a-french-dicegame-materials-rules` (visited on 02/07/2018).

[134] P. de Jong, C. Sprenger, and F. van Veen. "On Extreme Values of Moran's I and Geary's c". In: *Geographical Analysis* 16.1 (1984), pp. 17–24.

[135] D. Jungnickel. *Graphs, Networks, and Algorithms, 2th ed.* Algorithms and Computation in Mathematics, 2005.

[136] A. Kapoor et al. "Hyperparameter annd kernel learning for graph based semi-supervised classification". In: *Advances in Neural Information Processiong Systems (NIPS 2005)* (2002), pp. 627–634.

[137] H. Kappen, V. Gómez, and M. Opper. "Optimal control as a graphical model inference problem". In: *Machine learning* 87.2 (2012), pp. 159–182.

[138] J. Kapur and H. Kesavan. *The Generalized Maximum Entropy Principle (with Applications)*. Sandford Educational Press, 1987.

[139] J. Kapur and H. Kesavan. *Entropy Optimization Principles with Applications*. Academic Press, 1992.

[140] T. Kato, H. Kashima, and M. Sugiyama. "Robust label propagation on multiple networks". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 35–44.

[141] J. Kemeny and L. Snell. *Finite Markov Chains*. Springer-Verlag, 1976.

[142] J. Kemeny, L. Snell, and A. Knapp. *Denumerable Markov chains*. Springer-Verlag, 1976.

[143] J. Kittler and F. Alkoot. "Sum versus vote fusion in multiple classifier systems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.1 (2003), pp. 110–115.

[144] I. Kivimäki, M. Shimbo, and M. Saerens. "Developments in the theory of randomized shortest paths with a comparison of graph node distances". In: *Physica A: Statistical Mechanics and its Applications* 393 (2014), pp. 600–616.

[145] I. Kivimäki et al. "Two betweenness centrality measures based on randomized shortest paths". In: *Scientific Reports* 6 (2016), srep19668.

[146] D. Klein. "Centrality measure in graphs". In: *Journal of Mathematical Chemistry* 47 (4 2010), pp. 1209–1223.

[147] D. Klein and M. Randic. "Resistance distance". In: *Journal of Mathematical Chemistry* 12.1 (1993), pp. 81–95.

[148] G. Klir and T. Folger. *Fuzzy Sets, Uncertainty, and Information*. Prentice-Hall, 1988.

[149] E. Kolaczyk. *Statistical analysis of network data: methods and models*. Springer, 2009.

[150] D. König. *Theorie der Endlichen und Unendlichen Graphen*. Akademische Verlagsgesellschaft, Leipzig, 1936.

[151] L. Kuncheva. *Combining pattern classifiers: methods and algorithms*. Wiley, 2004.

[152] F. Lad. *Operational subjective statistical methods*. John Wiley & Sons, 1996.

[153] K. Lang. "Newsweeder: Learning to filter netnews". In: *Proceedings of the Twelfth International Conference on Machine Learning*. 1995, pp. 331–339.

[154] L. Lebart. "Contiguity analysis and classification". In: *Data Analysis*. Ed. by Wolfgang Gaul, Otto Opitz, and Martin Schader. Studies in classification, data analysis, and knowledge organization. Springer, 2000, pp. 233–243.

[155] L. Lebart, A. Morineau, and 4th ed. M. Piron. *Statistique Exploratoire Multidimensionnelle*. Dunod, 1995.

[156] B. Lebichot et al. "Semi-Supervised Classification through the Bag-of-Paths Group Betweenness". In: *IEEE Transactions on Neural Networks and Learning Systems* 25 (6 2014), pp. 1173–1186.

[157] J. LeSage and R. Pace. *Introduction to Spatial Econometrics*. Chapman & Hall, 2009.

[158] W. Levy and H. Delic. "Maximum Entropy Aggregation of Individual Opinions". In: *IEEE Transactions on Systems, Man and Cybernetics* 24.4 (1994), pp. 606–613.

[159] D. Liben-Nowell and J. Kleinberg. "The link-prediction problem for social networks". In: *Journal of the American Society for Information Science and Technology* 58.7 (2007), pp. 1019–1031.

[160] M. Littman. "Markov Games as a Framework for Multi-agent Reinforcement Learning". In: *Proceedings of the 11th International Conference on Machine Learning (ICML-94)*. 1994, pp. 157–163.

[161] J. Liu and J. Han. "Spectral clustering". In: *Data clustering: Algorithms and applications*. Ed. by C. Aggarwal and C. Reddy. CRC Press, 2014, pp. 177–199.

[162] L. Lü et al. "Vital nodes identification in complex networks". In: *Physics Reports* 650 (2016), pp. 1–63.

[163] Q. Lu and L. Getoor. "Link-based classification". In: *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*. 2001, pp. 496–503.

[164] D. Luenberger. *Introduction to Dynamic Systems: Theory, Models, and Applications*. John Wiley & Sons, 1979.

[165] Sofus A. Macskassy and Foster Provost. "Classification in networked data: a toolkit and a univariate case study". In: *Journal of Machine Learning Research* 8 (2007), pp. 935–983.

[166] A. Mantrach. "Novel Measures on Directed Graphs and Applications to Large-scale Within-network Classification". PhD thesis. Universite Libre de Bruxelles, 2010.

[167] A. Mantrach et al. "The sum-over-paths covariance kernel: a novel covariance between nodes of a directed graph". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.6 (2010), pp. 1112–1126.

[168] Amin Mantrach et al. "Semi-supervised classification and betweenness computation on large, sparse, directed graphs". In: *Pattern Recognition* 44.6 (2011), pp. 1212 –1224.

[169] K. Mardia, J. Kent, and J. Bibby. *Multivariate Analysis*. Academic Press, 1979.

[170] J. McAuley and J. Leskovec. "Learning to Discover Social Circles in Ego Networks". In: *Advances in Neural Information Processing Systems (NIPS)* (2012).

[171] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions, 2nd ed*. Wiley, 2008.

[172] A. Meot, D. Chessel, and R. Sabatier. "Operateurs de voisinage et analyse des donnees spatio-temporelles (in french)". In: *Biometrie et environnement*. Ed. by D. Lebreton and B. Asselain. Masson, 1993, pp. 45–72.

[173] C. Merz. "Using correspondence analysis to combine classifiers". In: *Machine Learning* 36 (1999), pp. 226–239.

[174] C. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM Books, 2000.

[175] R. Milo et al. "Network Motifs: Simple Building Blocks of Complex Networks". In: *Science* 298 (2002), pp. 824–827.

[176] P. Moran. "Random associations on a lattice". In: *Proceedings of the Cambridge Philosophical Society*. 1947, pp. 321–328.

[177] P. Moran. "The Interpretation of statistical maps". In: *Journal of the Royal Statistical Society, Series B, 10* (1948), pp. 243–251.

[178] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[179] N. Augustin andM. Mugglestone and S. Buckland. "An Autologistic Model for the Spatial Distribution of Wildlife". In: *Journal of Applied Ecology* 33.2 (1996), pp. 339–347.

[180] I. Myung, S. Ramamoorti, and Jr A. Bailey. "Maximum Entropy Aggregation of Expert Predictions". In: *Management Science* 42.10 (1996), pp. 1420–1436.

[181] M. Newman. "A measure of betweenness centrality based on random walks". In: *Social Networks* 27.1 (2005), pp. 39–54.

[182] M. Newman. "Modularity and community structure in networks". In: *Proceedings of the National Academy of Sciences (USA)* 103 (2006), pp. 8577–8582.

[183] M. Newman. *Networks: an Introduction*. Oxford University Press, 2010.

[184] M. Newman and M. Girvan. "Finding and evaluating community structure in networks". In: *Physical Review E* 69 (2004), p. 026113.

[185] J. Norris. *Markov Chains*. Cambridge University Press, 1997.

[186] Martin J. Osborne. *An introduction to Game Theory*. Oxford University Press, 2004.

[187] L. Page et al. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab, 1999.

[188] J.-Y. Pan et al. "Automatic multimedia cross-modal correlation discovery". In: *Proceedings of the 10th ACM SIGKDD international conference on Knowledge Discovery and Data Mining (KDD 2004)* (2004), pp. 653–658.

[189] Y. Pawitan. *In all Likelihood: Statistical Modelling and Inference using Likelihood*. Oxford University Press, 2001.

[190] L. Peliti. *Statistical Mechanics in a Nutshell*. Princeton University Press, 2011.

[191] I. Petreska et al. "Application of modal analysis in assessing attack vulnerability of complex networks". In: *Communications in Nonlinear Science and Numerical Simulation* 15.4 (2010), pp. 1008–1018.

[192] D. Pfeiffer et al. *Spatial Analysis in Epidemiology*. Oxford University Press, 2008.

[193] C. Phua et al. "A Comprehensive Survey of Data Mining-based Fraud Detection Research". In: *Computing Research Repository* abs/1009.6119 (2010).

[194] Warren Powell. *Approximate Dynamic Programming, 2nd ed*. John Wiley and Sons, 2011.

[195] A. Dal Pozzolo et al. "Credit Card Fraud Detection: a Realistic Modeling and a Novel Learning Strategy". In: *IEEE Transactions on Neural Networks and Learning Systems* PP (99 2017), pp. 1–14.

[196] S. Prithviraj et al. "Collective Classification in Network Data". In: *AI Magazine* 29.3 (2008), pp. 93–106.

[197] M. Puterman. *Markov Decision Processes: Discrete Stochastic Programming*. John Wiley & Sons, 1994.

[198] M. Radovanović, A. Nanopoulos, and M. Ivanović. “Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data”. In: *Journal of Machine Learning Research* 11 (2010), pp. 2487–2531.

[199] M. Radovanović, A. Nanopoulos, and M. Ivanović. “On the Existence of Obstinate Results in Vector Space Models”. In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’10. ACM, 2010, pp. 186–193.

[200] Howard Raiffa. *Decision Analysis*. Addison-Wesley, 1970.

[201] R. Rardin. *Optimization in Operations Research*. Prentice Hall, 1998.

[202] L. Reichl. *A Modern Course in Statistical Physics, 2nd ed*. Wiley, 1998.

[203] Bolton RJ and DJ Hand. “Statistical fraud detection: A review”. In: *Statistical science* 17 (2002), pp. 235–249.

[204] K. Rosen. *Discrete Mathematics and Its Applications*. 2nd. McGraw-Hill, Inc., 1991.

[205] S. Ross. *Introduction to Probability Models, 10th Ed.* Academic Press, 2010.

[206] V. Roth. “Probabilistic discriminative kernel classifiers for multi-class problems”. In: *Pattern Recognition: Proceedings of the 23rd DAGM Symposium*. Vol. 2191. Lecture Notes in Computer Science. Springer, 2001, pp. 246–253.

[207] J. Rubin, O. Shamir, and N. Tishby. “Trading Value and Information in MDPs”. In: *Decision Making with Imperfect Decision Makers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 57–74.

[208] M. Saerens et al. “Randomized shortest-path problems: Two related models”. In: *Neural Computation* 21.8 (2009), pp. 2363–2404.

[209] A. Santiago and R. M. Benito. “Robustness of heterogeneous complex networks”. In: *Physica A: Statistical Mechanics and its Applications* 338 (2009), pp. 2234–2242.

[210] P. Sarkar and A. Moore. “A tractable approach to finding closest truncated-commute-time neighbors in large graphs”. In: *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)* (2007).

[211] B. Scholkopf and A. Smola. *Learning with Kernels*. The MIT Press, 2002.

[212] K. Schwab. *The Fourth Industrial Revolution*. World Economic Forum, 2016.

[213] H. Scudder. “Probability of error of some adaptive pattern-recognition machines”. In: *Information Theory, IEEE Transactions on* 11.3 (July 1965), pp. 363–371.

[214] G. Seber. *A matrix handbook for statisticians*. Wiley, 2008.

[215] Robert Sedgewick. *Algorithms, 4th ed*. Pearson, 2011.

[216] Stephen B. Seidman. “Network structure and minimum degree”. In: *Social Networks* 5.3 (1983), pp. 269–287.

[217] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[218] T. Silva and L. Zhao. *Machine Learning in Complex Networks*. Springer, 2016.

[219] V. Sindhwani, M. Belkin, and P. Niyogi. “The geometric basis of semi-supervised learning”. In: *Semi-supervised learning, O. Chapelle, B. Scholkopf and A. Zien (editors)*. MIT Press, 2006, pp. 217–235.

[220] M. Steele. *Stochastic Calculus and Financial Application*. Springer-Verlag, 2001.

[221] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, 1986.

[222] A. Subramanya and P. Pratim Talukdar. *Graph-based Semi-supervised Learning*. Morgan & Claypool Publishers, 2014.

[223] S. Sun. “A survey of multi-view machine learning”. In: *Neural Computing & Applications* 23 (7/8 Nov. 2013), pp. 2031–2038.

[224] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[225] R. Sutton and A. Barto. *Reinforcement Learning: an Introduction, 2nd ed. Draft manuscript in progress.* The MIT Press, 2017.

[226] M. Szummer and T. Jaakkola. “Partially labeled classification with Markov random walks”. In: *Advances in Neural Information Processiong Systems*. Ed. by T. Dietterich, S. Becker, and Z. Ghahramani. Vol. 14. Vancouver, Canada: MIT Press, 2001.

[227] A. Tahbaz and A. Jadbabaie. “A one-parameter family of distributed consensus algorithms with boundary: from shortest paths to mean hitting times”. In: *Proceedings of the IEEE Conference on Decision and Control (CDC)*. 2006, pp. 4664–4669.

[228] L. Tang and H. Liu. “Relational learning via latent social dimensions”. In: *Proceedings of the ACM conference on Knowledge Discovery and Data Mining (KDD 2009)*. 2009, pp. 817–826.

[229] L. Tang and H. Liu. “Scalable learning of collective behavior based on sparse social dimensions”. In: *Proceedings of the ACM conference on Information and Knowledge Management (CIKM 2009)*. 2009, pp. 1107–1116.

[230] L. Tang and H. Liu. “Toward predicting collective behavior via social dimension extraction”. In: *IEEE Intelligent Systems* 25.4 (2010), pp. 19–25.

[231] H. Taylor and S. Karlin. *An Introduction to Stochastic Modeling, 3th Ed.* Academic Press, 1998.

[232] S. Theodoridis and K. Koutroumbas. *Pattern recognition, 4th ed.* Academic Press, 2009.

[233] E. Theodorou, D. Krishnamurthy, and E. Todorov. “From information theoretic dualities to path integral and kullback-leibler control: Continuous and discrete time formulations”. In: *The Sixteenth Yale Workshop on Adaptive and Learning Systems*. 2013.

[234] E. Theodorou and E. Todorov. “Relative entropy and free energy dualities: Connections to path integral and kl control”. In: *Proceedings of the 51st IEEE Conference on Decision and Control (CDC 2012)*. IEEE. 2012, pp. 1466–1473.

[235] H. Tijms. *A First Course in Stochastic Models*. John Wiley & Sons, 2003.

[236] A. Tizghadam and A. Leon-Garcia. “Betweenness centrality and resistance distance in communication networks”. In: *IEEE Network* 24.6 (2010), pp. 10–16.

[237] E. Todorov. “Linearly-solvable Markov decision problems”. In: *Advances in Neural Information Processing Systems 19 (NIPS 2006)*. MIT Press, 2006, pp. 1369–1375.

[238] H. Tong, C. Faloutsos, and J.-Y. Pan. “Random walk with restart: fast solutions and applications”. In: *Knowledge and Information Systems* 14.3 (2008), pp. 327–346.

[239] H. Tong et al. “On the Vulnerability of Large Graphs”. In: *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM '10)*. 2010, pp. 1091–1096.

[240] J. Tukey. “Comparing individual means in the analysis of variance”. In: *Biometrics* 5 (June 1949), pp. 99–114.

[241] V. Van Vlasselaer et al. “APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions.” In: *Decision Support Systems* 75 (2015), pp. 38–48.

[242] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[243] U. von Luxburg. “A Tutorial on Spectral Clustering”. In: *Statistics and Computing* 17.4 (2007), pp. 395–416.

[244] U. von Luxburg, A. Radl, and M. Hein. "Getting lost in space: large sample analysis of the commute distance". In: *Proceedings of the 23th Neural Information Processing Systems conference (NIPS 2010)* (2010), pp. 2622–2630.

[245] K. Wagstaff et al. "Constrained K-means Clustering with Background Knowledge". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. Morgan Kaufmann Publishers Inc., 2001, pp. 577–584.

[246] T. Waldhor. "Moran's spatial autocorrelation coefficient". In: *Encyclopedia of Statistical Sciences, 2nd ed. (S. Kotz, N. Balakrishnana, C. Read, B. Vidakovic and N. Johnson, editors)*. Vol. 12. Wiley, 2006, pp. 7875–7878.

[247] L. Waller and C. Gotway. *Applied Spatial Statistics for Public Health Data*. Wiley, 2004.

[248] J. Wang et al. "Linear neighborhood propagation and its applications". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.9 (2009), pp. 1600–1615.

[249] J. Ward. "Hierarchical grouping to optimize an objective function". In: *Journal of American Statistical Association* 58 (1963), pp. 236–244.

[250] S. Wasserman and K. Faust. *Social network analysis: methods and applications*. Cambridge University Press, 1994.

[251] K. Wehmuth and A. Ziviani. "Distributed location of the critical nodes to network robustness based on spectral analysis". In: *Proceedings of the 7th Latin American Network Operations and Management Symposium (LANOMS)*. 2011, pp. 1–8.

[252] D. West. *Introduction to Graph Theory*. 2nd. Prentice Hall, 2001.

[253] D. Weston et al. "Plastic card fraud detection using peer group analysis". In: *Advances in Data Analysis and Classification* 2.1 (2008), pp. 45–62.

[254] D. White. "Real applications of Markov decision processes". In: *Interfaces* 15.6 (1985), pp. 73–83.

[255] Douglas White. "Further real applications of Markov decision processes". In: *Interfaces* 18.5 (1988), pp. 55–61.

[256] Douglas J White. "A survey of applications of Markov decision processes". In: *Journal of the Operational Research Society* 44.11 (1993), pp. 1073–1096.

[257] L. Yen et al. "A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances". In: *Proceedings of the 14th SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008)*. 2008, pp. 785–793.

[258] L. Yen et al. “Graph nodes clustering with sigmoid commute-time kernel : A comparative study”. In: *Data & Knowledge Engineering* 68 (2009), pp. 338–361.

[259] V. Zaslavsky and A. Strizhak. “Credit card fraud detection using self-organizing maps”. In: *Information and Security* 18 (2006), p. 48.

[260] D. Zhang and R. Mao. “Classifying networked entities with modularity kernels”. In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008)*. ACM. 2008, pp. 113–122.

[261] J. Zhao et al. “Multi-view Learning Overview: Recent progress and new challenges”. In: *Information Fusion* 38.C (Nov. 2017), pp. 43–54.

[262] D. Zhou, J. Huang, and B. Scholkopf. “Learning from Labeled and Unlabeled Data on a Directed Graph”. In: *Proceedings of the 22nd International Conference on Machine Learning*. 2005, pp. 1041–1048.

[263] D. Zhou and B. Scholkopf. “Learning from labeled and unlabeled data using random walks”. In: *Proceedings of the 26th DAGM Symposium*. 2004, pp. 237–244.

[264] D. Zhou and B. Scholkopf. “Discrete regularization”. In: *Semi-supervised learning, O. Chapelle, B. Scholkopf and A. Zien (editors)*. MIT Press, 2006, pp. 237–249.

[265] D. Zhou et al. “Learning with local and global consistency”. In: *Proceedings of the Neural Information Processing Systems Conference (NIPS 2003)*. 2003, pp. 237–244.

[266] X. Zhu. “Semi-supervised learning literature survey”. unpublished ma-nuscript. 2008. URL: `pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf`.

[267] X. Zhu, Z. Ghahramani, and J. Lafferty. “Semi-supervised learning using gaussian fields and harmonic functions”. In: *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*. 2003, pp. 912–919.

[268] X. Zhu and A. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan & Claypool Publishers, 2009.