Università degli Studi "Roma Tre"

Ph.D. Thesis in Mathematics

ML- and graph-based methods for data analysis and prediction

Applications in vehicular traffic, pedestrian dynamics, and computational biology

Candidate:
Elia Onofri

Supervisors:
Dr. Filippo Castiglione
Dr. Emiliano Cristiani
Prof. Marco Pedicini

Coordinator:
Prof. Alessandro Giuliani

Academic Year 2022/2023 – XXXVI cycle

ROMA TRE
UNIVERSITÀ DEGLI STUDI

Matematica
CORSO DI DOTTORATO DI RICERCA IN

XXXVI
CICLO DEL CORSO DI DOTTORATO

ML- and graph-based methods for data analysis and prediction

Applications in vehicular traffic, pedestrian dynamics, and computational biology

Titolo della tesi

Elia Onofri — _____
Nome e Cognome del dottorando — firma

Marco Pedicini — _____
Docente Guida/Tutor: Prof. — firma

Alessandro Giuliani — _____
Coordinatore: Prof. — firma

Elia Onofri

ML- and graph-based methods for data analysis and prediction

Applications in vehicular traffic, pedestrian dynamics, and computational biology

Academic Year 2022/2023

Ph.D. Thesis in Mathematics
Università degli Studi Roma Tre

Supervisors

Dott. Filippo Castiglione
Executive Director Bio-Informatics
Biotech Research Center
Technology Innovation Institute, Abu Dhabi, United Arab Emirates

Dott. Emiliano Cristiani
Research Director
Istituto per le Applicazioni del Calcolo “M. Picone”
Consiglio Nazionale delle Ricerche, Rome, Italy

Prof. Marco Peidicini
Associate Professor, INF/01
Department of Mathematics and Physics
University of Roma Tre, Rome, Italy

Doctorate Coordinator

Prof. Alessandro Giuliani
Full Professor, MAT/07
Department of Mathematics and Physics
University of Roma Tre, Rome, Italy

Dissertation committee

Prof. Vincenzo Bonifaci
Associate Professor, INF/01
Department of Mathematics and Physics
University of Roma Tre, Rome, Italy

Prof.ssa Fosca Giannotti
Full Professor, INF/01
Scuola Normale Superiore, Pisa, Italy

Dott. Roberto Natalini
Institute Director
Istituto per le Applicazioni del Calcolo “M. Picone”
Consiglio Nazionale delle Ricerche, Rome, Italy

Substitute members

Prof. Luciano Teresi
Associate Professor, MAT/07
Department of Mathematics and Physics
University of Roma Tre, Rome, Italy

Dott.ssa Daniela De Canditiis
Senior Researcher
Istituto per le Applicazioni del Calcolo “M. Picone”
Consiglio Nazionale delle Ricerche, Rome, Italy

To my beloved wife Francesca and our cherished, precious, daughter Iulia, whose unwavering love and support illuminate my life's journey with boundless joy and inspiration.

Abstract

This Ph.D. thesis introduces some research topics the candidate worked on during his Ph.D. journey at *"Università degli Studi Roma Tre"*, presenting the candidate's contributions in these fields. The work mainly introduces results in graph algorithms, pedestrian management, vehicular traffic estimation, and drug repurposing. Despite the heterogeneity of these topics, they all share a common foundation in graph theory, with machine learning techniques applied to analyse and predict data patterns.

The present work opens by introducing the basics of graph theory, including the representation and efficient contraction of graphs. It also covers the generation of random graphs and the analysis of random walks over graphs. Additionally, selected topics in machine learning are discussed, such as artificial neural networks and clustering methods.

The candidate's contributions are then presented in individual chapters.

As regards the first contribution, a novel algorithm for contracting graphs equipped with an attribute-based colouring is introduced. The mathematical setting is formalised and theoretical definitions and practical implementations are provided.

The second and largest contribution focuses on pedestrian management in crowded museum environments. A coloured-graph representation of museum-like environments is presented and analysed. A semi-automatic IoT-based tracking system for Lagrangian data is described and the problem of reconstructing trajectories from noisy data is tackled. Different data analysis techniques are discussed. Finally, aided by a digital twin opportunely created, optimisation strategies are considered and proposed. The two world-renowned case studies of the museum of Galleria Borghese in Rome and the Peggy Guggenheim Collection in Venice, validate the approach.

The third contribution integrates machine learning techniques with macroscopic differential models for vehicular traffic estimation and forecast. The corresponding chapter explores congestion event detection and upcoming flux prediction by means of machine learning techniques for time series analysis. These results are used to enhance different aspects of the numerical model, namely helping in the inversion of the fundamental diagram of traffic and providing future boundary conditions that are not available in real-time.

The fourth and final contribution presents a novel approach for drug repurposing based on a recommender system. A drug-disease knowledge graph is constructed using gene similarity scores, and recommendations are generated using biased random walks. The approach is validated through a case study on rheumatoid arthritis and classical benchmark approaches.

Closes the thesis a summary of the introduced works and an extended list of the candidate's publications.

Keywords: Graph analysis · machine learning · colour contraction · supervised learning · clustering

Preface

The present thesis explores the candidate's work and contribution, which spanned over the course of three years rich in collaborations. Throughout this period of time, the candidate had the opportunity to confront himself with novel and captivating challenges, interact with remarkable researchers from diverse fields, and engage with various research groups.

Beyond stimulating the candidate's intellectual curiosity across a wide range of topics, this journey enabled the candidate to dive in and apply a significant portion of the knowledge acquired throughout his entire academic path, spanning a bachelor's degree in mathematics and a master's degree in computational sciences.

The following pages delve into four research themes, which, despite their apparent differences, share a common extended mathematical foundation that will hopefully be elucidated throughout the course of this work. The primary focus of this work revolves around the topics of graph theory and machine learning, employed to analyse and predict data patterns within heterogeneous data.

The interdisciplinary nature of the work is further emphasised by the presence of three supervisors, each representing different aspects and thematics of the research landscape the candidate is involved in. Their expertise spans theoretical and computational modelling, computational biology and immunology, and theoretical and applied graph theory, collectively enriching the candidate's research journey and ensuring a comprehensive exploration of the presented works.

Road map

Chapter 1 provides the basics of graph theory, laying out the notation that will be used in this work. In particular, Section 1.2 introduces all the basics about (simple, undirected) graphs, Section 1.3 considers some useful generalisation including digraphs and multi-graphs, and Section 1.4 delves into the problem of efficient representation of graphs. Later, Section 1.5 introduces the problem of graph contraction that will be later analysed in Chapter 3 and Section 1.6 dis-

cusses the problem of generating random graphs. Finally, Section 1.7 describes random walks over graphs and provides some basic definitions and properties of Markov chains that will be later used in Chapter 6.

Chapter 2 covers some selected arguments in machine learning. In particular, Section 2.1 opens the chapter by discussing the deep entanglement between mathematics and machine learning and presenting a broad overview of its different fields. Section 2.2 introduces more in detail the setting of supervised learning and settles the ground for Section 2.3 which provides an overview of artificial neural networks by focusing on feed-forward and recurrent neural networks. Section 2.4 closes the chapter by delving into the topic of clustering, mainly focusing on centroid-based (Section 2.4.1) approaches and hierarchical analysis (Section 2.4.2).

Chapter 3 presents the first result of the candidate by introducing a novel algorithm for tackling the contraction of coloured graphs which is performance-oriented. In particular, the concepts of γ-contraction and β-contraction are formally defined and details are discus regarding the theoretical aspect of contracting over a colouring γ. The novel algorithm is presented both in abstract and implementation terms and suitable data structures are introduced. Benchmarks over random graphs and an application example are further analysed. The chapter closes with a final discussion.

Chapter 4 describes an all-around study on crowd tracking, modelling, and simulating performed in the setting of cultural heritage. In particular, a novel coloured graph-based approach is introduced for representing museum-like environments. A semi-automatic tracking system is presented, and both Lagrangian and Eulerian data are analysed. The problem of Lagrangian trajectory reconstruction from noisy data is tackled in-depth, requiring the introduction of a novel approach based on cascaded localisers resembling the graph structure of the environment. Multiple data analysis techniques are exploited, including a hierarchical clustering approach based on a custom metric obtained by the graph structure. Later, a technique for digital twin creation based on a time-varying Markov model is introduced. Finally, the digital twin is used to improve the quality of the museum fruition. Two case studies – the museums of Galleria Borghese in Rome and the Peggy Guggenheim Collection in Venice – further validate the approach presented here.

Chapter 5 introduces novel methods that integrate machine learning techniques with macroscopic differential models for vehicular traffic estimation and forecast. The chapter discusses traffic data of flux and velocity, gathered from fixed sensors along a highway network. In particular, two main tasks are considered: congestion events detection and expected upcoming flux prediction. Such tasks fit well in the context of time series analysis and, apart from being interesting *per se*, can enrich the data that are later exploited by the differential model for traffic state estimation. In particular, congestion detection is used to propose a novel density-based approach (relying on the inversion of the fundamental diagram) for injecting flux data into the model, hence overcoming numerical issues that might lead to unfeasible situations. Conversely, the expected upcoming flux

is used in place of boundary conditions (missing in real-time estimation) for forecasting the traffic state. The chapter closes with a description of simulation examples with both real and synthetic data.

Chapter 6 describes the last contribution of the candidate presented in this thesis, a novel approach for drug repurposing based on a recommender system hinged on gene similarity scores and biased random walks. In particular, different sources of data are considered for the creation of a drug-disease knowledge graph obtained as the joining of two similarity-weighted complete graphs. Recommendations are then obtained as paths of short length, highlighted by biased random walks. The recommendation mechanism is made *explainable* by a sage usage of the Markov chain underlying the random walk process. The stability of the model is further discussed, based on the Ergodicity of the process. A case study – rheumatoid arthritis – is used, along with classical performance evaluation techniques, to validate the novel methodology.

How to read this thesis

The present work covers four different research topics by introducing novel contents in coloured-based contraction, pedestrian management, vehicular traffic estimation and drug repurposing. Despite being very dissimilar one from the other, all of the approaches here proposed deepen their roots in a common background which is primary-based on graph theory and, secondary, on machine learning approaches (which are typically built on top of some clever graph-based representation of the problem).

Part I introduces a common ground useful for understanding the contributions presented in this thesis. However, Chapters 1 and 2 do not introduce any novel result and can be safely skipped by readers familiar with graph theory and machine learning. We solely remind the reader that we refer to undirected simple graphs (unless differently specified) and that with the term colouring, we refer to a typological mapping (with values in $C \subsetneq \mathbb{N}, |C| < \infty$) rather than a proper colouring, as understood in the well-known colouring problems.

The Chapters 3–6 from Parts II–IV hold the candidate's contributions and are thought of as self-contained chapters, only requiring the basics from Part I. Hence, they can be read independently one from the other and in any order. For this reason, each chapter is provided with (i) an abstract summarising the content, (ii) a literature discussion on the specific topic, and (iii) a *conclusions* section which also discusses possible further developments.

Each chapter of this work introduces specific own naming, acronyms, and symbols, which are consistent[1] throughout the text. In particular, Chapter 1 (especially Section 1.2) is devoted to introducing the entire graph-related notation. Section Mathematical notation (page xxxvii) introduces, for the sake of completeness, a few well-

[1] Consistent at the best of the author's knowledge

known mathematical notations that are assumed known. Section List of acronyms and symbols (page xxix) provides a list of chapter-wise organised abbreviations and symbols which can be helpful while navigating along the present work.

Candidate's contributions

This thesis mainly presents contributions that the candidate published as journal papers or conference proceedings, Chapter 5 being the exception since the corresponding work is under review at the time of writing. A few results presented here are novel also *w.r.t.* the published articles (see below) since they belong to articles that are currently work-in-progress: it is the case, *e.g.*, of the mathematical formalisation of γ-contraction in Chapter 3 and of results achieved in the Peggy Guggenheim Collection from Chapter 4.

Works here are not only collected but also harmonised together and extended in order to create a single and cohesive text.

In what follows, we report the direct contributions of the candidate to the specific parts of this work and we highlight the contribution that are not present in the original articles.

Chapter 3 – Graph contraction on attribute-based colouring The candidate focused mainly on the implementation of the code and on the literature review of the problem. In particular, the first serial version of the code was written by the candidate in multiple languages before the agreement on using `C` and the candidate also carried out the performance campaign over random graphs. The mathematical characterisation of the problem, which is formalised here for the first time, is entirely due to the candidate, as well as the proof of Theorem 3.1. The chapter heavily extends upon the published articles with regard to the context of mathematical formalisation of the problem and of the algorithm and improves the description of the implementation by means of (previously unpublished) `C` pseudo-codes.

Chapter 4 – Managing crowded museums The main areas of contribution of the candidate concern the novel-introduced graph-based approach, the machine learning development and the realisation of the technical/implementation part of the project. More in detail, the candidate

- handled the technical and implementation part of the project, developing, managing, and maintaining the data collection systems (IoT and Eulerian).
- proposed, described, and formalised the idea of representing museums as coloured graphs and elaborated the way the metric could arise from such a representation.
- designed, developed, and validated trajectory refinement methods and the creation of corresponding datasets.
- was involved in preliminary analysis of the trajectories
- carried out the trajectory clustering

- contributed to the tuning of the digital twin and the drafting of recommendations that were then taken into account by the curators of the museums adopted as case studies.

The main novelty of the chapter *w.r.t.* the published work consists of all the details, descriptions, and results regards the Peggy Guggenheim Collection.

Chapter 5 – Hybrid approach for traffic state estimation and forecast The principal area of contribution of the candidate concerns the machine learning development, implementation and application, as well as the interpretation of the results. More in detail, the candidate commits himself to a careful data analysis whose outcome consists of the proposal of the methodologies presented throughout this text along with various applications in anomaly detection (that here are not discussed since results are still in an early stage). The work discussed here presents very few changes from the paper submitted in [282], also available as a preprint.

Chapter 6 – Explainable drug repurposing The study presented here was entirely executed by the candidate, who provided the mathematical background and developed the code to perform the analysis and validation of the methodology. The idea of the recommender system on gene similarity, originally provided by the candidate's supervisor F. Castiglione, was later explored and compared to other literature approaches by the candidate himself. The datasets (provided by coauthor Paolo Tieri) were organised, analysed, filtered, and integrated by the candidate. The analysis of the case study was not a prerogative of the candidate due to the very different background required. The sole novelty *w.r.t.* the published version in [270], is represented by a clarification on benchmarks results and the correction of a few discrepancies in notation.

List of contributions

In the following, we report the list of the candidate's contributions relevant to the topics covered in this work, organised by chapter. In the closing of the present thesis, in Author's contributions (p. C-1), an extended version of this list is reported, organised by publication kind rather than by topic. The extended list also includes other contributions the candidate worked on during his Ph.D. which are not relevant to the results presented in this thesis.

Chapter 3 – Graph contraction on attribute-based colouring

[287] E. Onofri. "On the theoretical aspects of colour contraction: γ-contraction". In preparation

[285] F. Lombardi and E. Onofri. "Parallel graph contraction on attribute-based colouring". In preparation

[283] M. Caprolu, F. Lombardi, and E. Onofri. "Speculative Polkadot? an overview". In preparation

[274] F. Lombardi and E. Onofri. "Some results on colored network contraction". In: *Journal of Ubiquitous Systems and Pervasive Networks* 17.2 (Dec. 2022), pp. 91–98. DOI: 10.5383/JUSPN.17.02.006

[278] F. Lombardi and E. Onofri. "Graph contraction on attribute-based coloring". In: *Procedia Computer Science* 201 (Apr. 2022). The 13th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 5th International Conference on Emerging Data and Industry 4.0 (EDI40), pp. 429–436. ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.03.056

Chapter 4 – Managing crowded museums

[284] M. Catrambone, P. Centorrino, E. Cristiani, E. Onofri, and C. Riminesi. "On the pollution impact of visitors inside crowded museums". In preparation

[289] P. Centorrino, E. Cristiani, P. Ferrara, D. Macchion, and E. Onofri. *Measurement and analysis of the visitors behavior in the Peggy Guggenheim Collection*. Technical report. IAC–CNR, Feb. 2023, pp. 1–12

[280] E. Onofri and A. Corbetta. "RSSi-based visitor tracking in museums via cascaded AI classifiers and coloured graph representations". In: *Collective Dynamics* 6 (Jan. 2022), pp. 1–17. DOI: 10.17815/CD.2021.131

[271] P. Centorrino, A. Corbetta, E. Cristiani, and E. Onofri. "Managing crowded museums: visitors flow measurement, analysis, modeling, and optimization". In: *Journal of Computational Science* 53 (Apr. 2021), pp. 1–17. ISSN: 1877-7503. DOI: 10.1016/j.jocs.2021.101357

[276] P. Centorrino, A. Corbetta, E. Cristiani, and E. Onofri. "Measurement and analysis of visitors' trajectories in crowded museums". In: *IMEKO TC-4*. IMEKO:2019-83. Florence, Italy: International Conference on Metrology for Archaeology and Cultural Heritage, Dec. 2019, pp. 423–428

Chapter 5 – Hybrid approach for traffic state estimation and forecast

[282] M. Briani, E. Cristiani, and E. Onofri. "Inverting the fundamental diagram and forecasting boundary conditions: how machine learning can improve macroscopic models for traffic flow". ARXIV:2303.12740. June 2023

Chapter 6 – Explainable drug repurposing

[270] F. Castiglione, C. Nardini, E. Onofri, M. Pedicini, and P. Tieri. "Explainable drug repurposing approach from biased random walks". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (July 2022). PMID:35839194, pp. 1009–1019. DOI: 10.1109/TCBB.2022.3191392

Rome, June 2023 *Elia Onofri*

Acknowledgements

Writing this acknowledgement section is an incredibly hard heartfelt endeavour, as I had the privilege of interacting with many extraordinary individuals throughout my academic journey. My hope is to express my gratitude to each of them without omission, though I acknowledge that this list can never be exhaustive. It is difficult even to decide who to thank first and who to acknowledge last; hence I think the most important figures for my academic formation and development will fulfil those positions.

First and foremost, my deep and sincere gratitude goes to Professor Marco Pedicini, who drove me during my whole career as a student at Roma Tre University; Marco was the first Professor who believed in me, following me since my bachelor studies, and I feel honoured having been able to create three thesis with him. My work with Marco is, in fact, not limited to what I present here: we achieved many results in the field of cryptography and I hope our collaboration will pursue and flourish in the future. His guidance throughout these years was cardinal to let me where I am now, and I can only acknowledge him as my academic father.

A special acknowledgement is to be reserved also for the Executive/Research Director Filippo Castiglione who introduced me to the world of machine learning first, and to computational biology then. Filippo deemed me suitable for a Ph.D. and retains my endless gratitude for having partially financed me during the last three years of my formation. Thanks to him I got in touch with international experts in the field of immunology, including physicists, biologists, medical doctors, and many others.

Apart from Marco Pedicini and Filippo Castiglione, many other professors and colleagues have contributed to my high formation both at Roma Tre University and at the institute “Mauro Picone” (IAC–CNR), and here I would like to thank a few: Marco Cianfriglia, one of the most gentle and capable computer scientist I know of who helped me out in countless situations; Alessandro Corbetta, the physicist who worked side-by-side with me, investing a huge amount of time in teaching me a countless number of skills in computer management, who showed me how to create beautiful (and meaningful) images, and who instructed me that images and

text should be mutually explanatory in most scientific papers; Roberto D'Autilia, the original stakeholder for the graph contraction algorithm I present in Chapter 3; Enrico Mastrostefano, the matchless expert in computer management who drove me in agent-based simulations and taught me that physicists argue on everything but with the best intents; Francesca Merola, the most enjoyable algebra professor and colleague of mine in teaching cryptography, without whom I could not have taught it; Marco Liverani, my first instructor in C language and, now, my colleague in teaching it; Flavio Lombardi, the most inspiring teacher I ever had, irreplaceable friend, and colleague of a kind; Alberto Paoluzzi, the mentor who crashed me into the world of real programmers and first made me work on a scientific paper;

Along with those high experts, a heartfelt thanks is devoted also to my fellow graduate students. In particular, I would like to thank Armando Capasso for his always-available support and for joining me in the hard work we embarked on when we decided to organise graduate student seminars in our math department and Bruno Renzi for keeping my smile always on. I think that Pietro Centorrino also falls in this section (at least *ad honorem*) since he decided not to embark himself on Ph.D. studies regardless of my suggestions, but rather to become an independent researcher and professor; his friendship and never-ending good mood supported me in many aspects of my research studies.

Acknowledgements are also required for those who, more or less directly, supported my research activities. They include, but are not limited to: Mauro Antonio Caprolu, who is collaborating with us on the analysis of Polkadot, and my student Francesca Bacco, who spotted some minor issues with the γ-contraction code, both discussed in Chapter 3; the staff of Galleria Borghese, Luciano Pensabene and the staff of Peggy Guggenheim Collection for their collaboration in our projects concerning museums, along with Alberto Contorno, Francesca D'Orazi, Francesca Sanzò, Luca Solinas, Sara Suriano, and all the friends who helped us with the ground measurements presented in Chapter 4; my coauthor Maya Briani, whose contribution to the numerical simulations was cardinal, Paolo Ranut, who referenced our work at Autostrade Alto Adriatico (ex Autovie Venete), along with Andrea Appella, Giulia Tatafiore, and my students Andrea Lisi and Davide Moretti who helped us in data analysis and performance evaluation for the results presented in Chapter 5; my coauthors Christine Nardini and Paolo Tieri, whose contribution as coauthors was essential for the development of the research presented in Chapter 6.

I left, however, the last spot of this long list of colleagues, mentors, and collaborators for (arguably) the most important person, without the support of which this work of thesis would not be a morsel of what it turned out to be. Last, but not least surely, I thence would like to thank my advisor, the Research Director Emiliano Cristiani, whose help, support, and guidance goes at the very least beyond my academic formation. I consider Emiliano a mentor of a kind and, most arguably, a friend. Always available to provide advice and explanations, Emiliano, above all, has allowed me to understand many of the hidden mechanisms behind the research field, and, therefore, has made me a more prepared and (hopefully) better mathematician. Without his contribution, I would not have ventured to tackle many of the topics covered in this work, as a significant portion of the contributions presented here are the result of

the collaborations we have had together over the last four and a half years. The trust he showed to me during this time has been a great source of motivation for me to improve over and over, and I hope I will be able to reciprocate all of this in the years to come.

On top of all of my colleagues, my deepest and strongest gratitude goes to my whole family who supported me throughout these long years. Without all of them, with my parents Daniela and Giovanni in the first place, I would not be where I stand today. Unfortunately, none of my grandparents will be there during my discussion, but I remember them with never-ending fondness. A special thanks also goes to Alberto, my longest-standing friend who I consider part of my family, and who keeps reminding me that, “despite everything, it is still me”.

My gratitude and eternal love, however, goes to my wife Francesca, who sustains and endures life with me every day, and to my daughter Iulia, whose journey I hope to support endlessly.

Rome, November 2023

Tak for alt,
Elia

Contents

Part III – Pedestrian & vehicular traffic

Part IV – Computational biology

List of Figures

List of Tables

List of Algorithms

List of acronyms and symbols

List of well known symbols

$\mathbb{N}, \mathbb{N}^+, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$	Non-negative and positive integers; integer, rational, real numbers
$\mathbb{F}_q, \mathbb{F}_2$	Finite field with q elements, binary field
$[a, b], (a, b)$	Closed and open intervals between a and b in $\mathbb{R}$
$\lfloor x \rfloor, \lceil x \rceil, \lfloor x \rceil$	Integer part of x, $x + 1$ and $x + 0.5$ for $x \in \mathbb{R} \backslash \mathbb{N}$
$\{\dots\}$	(unordered) set (without repetitions)
$\langle \dots \rangle$	(unordered) tuple (with repetitions)
$(\dots)$	(ordered) vector (with repetitions)
$S^n, S^{m \times n}$	n-length (column) vector and $m \times n$-sized matrix with values in S
$\cdot^{\mathbf{T}}, \cdot^{\tau}$	Matrix transpose *w.r.t.* main and off-diagonal
$a_i, B_{i,j}$	i-th element of vector $\mathbf{a}$, element in position i, j of matrix $\mathbf{B}$.
$\mathbf{0}^n, \mathbf{0}^{m \times n}$	zero n-length vector and zero $m \times n$-sized matrix
$\mathbf{1}^n, \mathbf{1}^{m \times n}$	unit n-length vector and identity $m \times n$-sized matrix
O, Ω, Θ	Asymptotical classes for upper- and lower-bounding
NP-hard, NP-c	Complexity classes of **NP** hard and complete problems
P, **NP**	Complexity classes of deterministic and non-deterministic polynomial problems

List of abbreviations from Chapter 1 – Graph theory

digraph	directed graph with self loops
graph	simple, undirected graph
colouring	non-proper colouring
DAG	Directed acyclic graph
ER	Erdös-Rényi model

List of symbols from Chapter 1 – Graph theory

$G = (V, E)$	A graph G with vertex set V and edge set E
$V^2, [V]^2, \langle V\rangle^2$	2-length vectors, sets, tuples with values in V
V_G, E_G	Vertex and edge set of the graph G
$n_G, \lvert G\rvert, \lvert V\rvert$	Order of the graph G
$m_G, \lVert G\rVert, \lvert E\rvert$	Size of the graph G
$u, v, w; e, f$	vertices and edges of a graph
$u \sim v, e \sim f$	adjacent nodes, adjacent vertices
$N(v), \partial(v)$	neighbourhood and degree of $v \in V$
$N(U), \partial(U)$	neighbourhood and degree of $U \subset V$
$\Delta(G), \delta(G)$	maximum- and minimum degree of G
$\partial(G)$	average degree of G
$G' \{\leq, <\} G$	Subgraph and proper subgraph
$\langle V'\rangle_G, \langle G'\rangle_G$	induced/spanned subgraph
$G \{\cup, \cap, *\} G'$	union, intersection, and join of graphs (right-join for digraphs)
$G \{+, -\} E'$	edges addition and removal from a graph
$G - V$	vertices removal from a graph
$X_{u,v}, \Gamma_{u,v}$	Walk (or trail) and path between u and v
$\lVert X_{u,v}\rVert$	Length of the walk
X_u, Γ_u	Tour and cycle from u to itself
$d(u, v)$	Distance between u and v ($d(u, v) = +\infty$ if disconnected)
r_T, T_r, ℓ	Root of a tree T, tree rooted in r, and height of the tree
$G \{=, \cong\} G'$	Graphs equality and graphs isomorphism
I^n, P^n, C^n, K^n	Independent-, path-, cycle-, and complete graph of order n
$K_{n_1,\dots,n_r}$	r-clique with partitions of size $n_1, \dots, n_r$
T_d^n	d-ary complete tree of order n.
ω_V, ω_E	Weight functions for vertices and for edges
γ, C, c	Colouring function, colour set and colour set size
$N^-(v), \partial^-(v)$	In-neighbourhood and in-degree (for digraphs)
$N^+(v), \partial^+(v)$	Out-neighbourhood and out-degree (for digraphs)
$G_1 \{\overset{\rightarrow}{*}, \underset{\leftarrow}{*}\} G_2$	right-, left-join of G_1 with G_2 (for digraphs)
$\mathbf{M}, \mathbf{A}$	Adjacency matrix and adjacency lists of a graph
$\mathbf{M}^{(k)}, \mathbf{W}^{(k)}$	k-step connectivity matrices (in $\mathbb{F}_2$ and $\mathbb{N}$)
G/U	Contraction of a graph G over the vertex subset U
$\mathfrak{m}_G(u_1, \dots)$	Contraction of a graph G seen as variadic function
$G' \{\preceq, \prec\} G$	Contraction graph and proper contraction graph
$\mathbf{ER}_{n,m}, \mathbf{ER}_{n,p}$	Uniform and binomial Erdös-Rényi model
$P, \mathbf{P}$	Markov chain transition probability and transition matrix
$\mathbf{P^t}$	Markov chain t-steps transition matrix
$\mathcal{T}(u)$	Random walk returning time set of the node u.
$\boldsymbol{\pi}, \boldsymbol{\Pi}$	Markov chain stationary distribution and stationary matrix

List of abbreviations from Chapter 2 – Machine learning[2]

AI	Artificial intelligence
ML	Machine learning
(R)MSE	(Root) mean squared error
ANN	Artificial neural network
FNN	Feed-forward neural network
MLP	Multi-layer perceptron
SLP	Single-layer perceptron
RNN	Recurrent neural network
LSTM	Long-short term memory
(A-)HCA	(Agglomerative) hierarchical clustering analysis
D-HCA	Divisive hierarchical clustering analysis
C-LINK	Complete Linkage
S-LINK	Single Linkage
UPGMC	Unweighted Pair Group Method with Mean Centroid

List of symbols from Chapter 2 – Machine learning

$\mathbf{x}, y$	Data sample, ground truth
$\mathbf{X}, \mathbf{y}$	Set of samples, set of ground truths
N, n	Number of samples in the dataset, number of features in a sample
D	Dataset (possibly labelled), see also $\mathbf{X}$
$\mathfrak{Q}, o$	Generic model, output of the model
$\mathcal{L}$	Generic (model) output space
k	Number of classes for classification/clustering tasks
$\hat{\mathbf{o}}, \hat{\mathbf{y}}$	Output/ground-truth (discrete) probability of being in a class
$\mathbf{p}, \mathbf{q}$	Probability distributions
$\mathbf{w}, b$	Weights, bias
f, a	Activation function, activation value
L_k, I	Linear activation with slope k, identity activation
φ_0	Heaviside activation
σ	Sigmoid activation
tanh	Hyperbolic tangent activation
(V, E, ω)	Edge-weighted digraph associated with an ANN
ℓ	Number of layers
a_v, v_a	activation value of neuron v and vice versa
$V_l, \mathbf{a}^{(l)}$	Neurones and activation value of layer l
$\mathbf{n}$	Number of neurones per layer in a MLP
$N_{\mathbf{IN}}, N_{\mathbf{OUT}}$	Number of input, output features

[2] Many more acronyms are introduced in this chapter while describing literature-related concepts, in particular *w.r.t.* methods names. However, we here solely report the acronyms that are useful for the comprehension of the concepts within this work. Furthermore, a few notation are also from Chapter 1.

T	Steps in a time series
$\mathbf{h}, N_{\mathbf{HID}}$	hidden state of RNN and corresponding size
$\mathbf{c}, \mathbf{g}$	cell state, gate of the LSTM
$\mathbf{W}, \mathbf{R}, \mathbf{b}$	Input weights, recurrent weights, biases
$\mathcal{C}, C$	Clustering, cluster
$\mathbf{M}, \mathbf{m}$	Set of centroids, centroid
$\boldsymbol{\mu}$	Mean centroid
$d(\cdot, \cdot)$	Distance amongst samples
$\mathcal{F}$	Family of clusterings
$d_{\mathbf{C}}(\cdot, \cdot)$	Distance amongst clusters
$\mathbf{d}, \mathbf{c}^{\xi}$	join-distance vector, size of ξ-significative clusters

List of symbols from Chapter 3 – Graph contraction on attribute-based colouring

$G = (V, E, \gamma)$	Input graph of the contraction procedure
G'	Output graph of the contraction procedure
n, n', m, m'	Order and size of G and G'.
γ, C, c	Vertex colour map, relative colour set, and number of colours
$G/\gamma\{u, v\}$	Colour-preserving contraction
$\mathfrak{m}_G^{\gamma}$	Colour-preserving contraction (procedure)
$G/_{\gamma}(U)$	Colour-preserving contraction (variadic form)
G/γ	Graph γ-contraction (colour contraction)
$N_{\gamma}(v), N_{\gamma}(U)$	Colour neighbourhood of v, of U
$\partial_{\gamma}(v), \partial_{\gamma}(U)$	Colour degree of v, of U
$\mathcal{S}, C_{\gamma}(G)$	Colour sub-partition and colour partition (also denoted $\mathcal{S}^{\star}$)
β	Map $V \to V'$ induced by contraction mapping evaluation algorithm
$\mathcal{S}_{\beta}$	Colour sub-partition induced by β
$D = (V, B)$	Rooted di-forest built according to contraction mapping algorithm
R, r_T	Root set of D, root of tree T
$\pi, \tilde{\pi}$	Map $V, \mathcal{S} \to R$ induced by D
$\hat{\alpha}, \tilde{\alpha}$	Map $R, \mathcal{S} \to \{0, \ldots, n'\}$ induced by r_T ordering
α	Map $R \to V'$ induced by $\hat{\alpha}$ ordering
G/β	β-contraction, iteration of the algorithm
`idx_t`, `col_t`	Data type for vertices and colours
p_1, p_2	Thresholds for regular ER model
p_-, p_+	Thresholds corresponding to p_1, p_2 for coloured ER model
$G_{\mathbf{FB}}$	Use case graph of Facebook

List of abbreviations from Chapter 4 – Managing crowded museums

BLE	Bluetooth low-energy
RSSI	Received signal strength indicator

Acc	Binary accuracy
PpR	People per room
Ret	Returning visitors
PDF	Probability density function
ToP	Time of permanence
ToT	Time over threshold
TVMM	Time-varying Markov model
CHF	Cumulative hazard function

List of symbols from Chapter 4 – Managing crowded museums

$\tilde{\cdot}$	Matrix convolution
$\bar{\cdot}$	Matrix (z-score, sum) normalisation
a_i, A	i-th antenna, number of antennas
$\mathcal{D}$	Room distance matrix
$\bar{\mathcal{D}}, \bar{\mathcal{D}}^*, \bar{\mathcal{W}}$	room- and trajectory-wise error metrics
δ	Time window
$\delta\mu, \delta\,\mathrm{VC}$	mean value relative error, variation coefficient relative error
$h_\star$	Hazard function for the whole museum
K_i	i-th colour cluster
$\mathcal{K}$	Room transition matrix
λ, k	Weibull distribution parameters
$\mathcal{M}(t)$	Time-dependent room transition matrix
$r_i, r_\star, R$	i-th room, entrance/exit room, number of rooms
$\mathbf{p}$	ToP vector
$\mathcal{R}$	RSSI matrix
$\mathfrak{R}, \mathfrak{F}, \mathfrak{C}, \mathfrak{Z}, \mathfrak{W}$	selectors
S_r	Survival function for room r
$\mathfrak{t}, \mathfrak{s}, N$	Two trajectories, number of trajectories
T	Length of a trajectory
$\mathcal{W}$	Trajectory distance matrix (Wasserstein-inspired)

List of abbreviations from Chapter 5 – Hybrid approach for traffic state estimation and forecast

TSE	Traffic state estimation
PDE	Partial differential equation
LWE	Lighthill–Whitham–Richards (traffic model)
PW	Payne–Whitham (traffic model)
ARZ	Aw–Rascle–Zhang (traffic model)
CTM	Cell transmission model (traffic model)
PINN	Physics-informed neural networks
PIDL	Physics-informed deep learning model

PUNN	Physics-uninformed neural networks
PRGP	Physics regularised Gaussian process
TLS	Technische Lieferbedingungen für Streckenstationen
$3T$	3-technology (sensors)
HF	High-flux (sensors)
LF	Low-flux (sensors)

List of symbols from Chapter 5 – Hybrid approach for traffic state estimation and forecast

f, ρ, v	Flux, density and velocity (variables)
$\mathbf{f}, \mathbf{v}$	Flux and velocity (vectors)
$\mathbf{f}^*, \mathbf{v}^*$	regularised flux and density (vectors)
$\rho_{\mathbf{MAX}}$	Maximum density allowed
$f_{\mathbf{MAX}}, \sigma$	Maximum reachable flux and corresponding optimal density σ
t_0	Present moment in time, time of nowcast prediction
$\Delta t_{\mathbf{PAST}}, \Delta t_{\mathbf{FUT}}$	Data are known for $[t_0 - \Delta t_{\text{PAST}}, t_0]$ and predicted for $(t_0, t_0 + \Delta t_{\text{FUT}})$
$\mathbf{X}$	Temporal sequence of feature vectors $\mathbf{x}$
$\mathbf{y}$	Ground-truth data
$N_{\mathbf{IN}}, N_{\mathbf{HID}}$	Number of input features of the ANN, hidden layers of LSTM
$\mathbf{o}, N_{\mathbf{PRED}}$	Prediction output and corresponding size
$\hat{\mathbf{o}}, N_{\mathbf{CLASS}}$	Classification normalised output, corresponding size, and best class
p_r	positive ratio of samples in the dataset
$\mathfrak{F}_c, \mathfrak{F}_p$	ANNs for detection and forecast of congestion events
$\mathbf{b}^{3T}$	Congestion events risen by $3T$ sensors
$\mathbf{b}^f, \mathbf{b}^v$	Flux and velocity heuristics for congestion events
$\mathfrak{P}^g$	ANN for forecast expected upcoming flux on group of sensors g
g_1, g_2, g_3, g_4	Boundary inflows of Venice, Trieste, Conegliano and Udine
S^k	k-th road segment between sensors s^k and s^{k+1} (out of N_S)
F	Numerical flux
$\Delta x, \Delta t$	Space and time discretisation step
N_t, N_x^k	Number of time intervals and space cells within the discretisation
$\rho_{\mathbf{L}}, f_{\mathbf{L}}, \rho_{\mathbf{H}}, f_{\mathbf{H}}$	Density and flux of light and heavy vehicles
ρ^N, ρ^F	Density obtained by the nowcast and by the forecast
$E^1(t)$	Relative L^1 error between ρ^N ρ^F

List of abbreviations from Chapter 6 – Explainable drug repurposing

BLAS	Basic linear algebra subprograms
DR	Drug repurposing
`cas-number`	Chemical abstracts service reference number
`UMLS`	Unified medical language system
OMIM	Online Mendelian inheritance in man

MeSH	Medical subject headings
ICD	International classification of diseases
NCD	Non-communicable disease
RA	Rheumatoid arthritis
GT	Ground truth
ROC	Receiver operating characteristic
AUC	Area under curve
DB	DrugBank (construction dataset)
DGN	DisGeNET (construction dataset)
ID	IDrug (validation dataset)
LL	Li and Lu article (validation dataset)
MC	MalaCards (construction dataset)
RDB	RepoDB (validation dataset)
SLAMS	Similarity-based LArge-margin learning of Multiple Sources (validation dataset)

List of symbols from Chapter 6 – Explainable drug repurposing

G_{DB}	Drug-gene graph based on *DB* dataset
G_{DGN}	Disease-gene graph based on *DGN* database
G_{MC}	Drug-disease graph based on *MC* relations
σ	Similarity score
$p(t)$	Presence ratio of gene t
$H(T)$	Shannon Entropy of the gene set T
$G_{\mathbf{drg}}, G_{\mathbf{dis}}$	Complete graph of drugs (disease)
ℓ	Length of a random walk, recommender iteration
$\mathfrak{p}$	Percentage of drugs to recommend
$(\mathcal{V}, \mathcal{E}, \omega_{\mathcal{E}})$	Final complete weighted digraph on $V_{\text{drg}} \cup V_{\text{dis}}$
$\mathcal{R}^{(\ell)}$	ℓ-th recommendation matrix
$\mathfrak{R}_{\ell}^{\mathfrak{p}}$	drug recommender system with parameters $\mathfrak{R}(\mathcal{R}, \ell, \mathfrak{p})$
X	biased random walk
$\mathcal{G}$	Markov chain
$\mathbf{\Pi}, \boldsymbol{\pi}$	Stationary distribution matrix (vector)

Mathematical notation

Throughout this work, a few well-known mathematical notations are assumed known. However, for the sake of completeness, we here report them divided into paragraphs.

Numeric set

$\mathbb{N}$ and $\mathbb{N}^+$ denote the set of non-negative and positive integers, $\mathbb{Z}$ identifies all the integers (relative numbers), $\mathbb{Q}$ stands for rational numbers and $\mathbb{R}$ for real numbers. We also employ $\mathbb{Z}_q$ as the integers modulo q, which actually is isomorphic to the quotient $\mathbb{Z}/q\mathbb{Z}$, and we use the $\equiv_q$ to denote an equivalence modulo q. We denote with $\mathbb{F}_q$ the finite field with q elements and, in particular, $\mathbb{F}_2$ represents the binary field – more formally, $(\mathbb{F}_2, +, \cdot)$, where $+$ is the *or* operator and $\cdot$ is the *and* operator. $[a, b]$ and (a, b) denotes the closed and the open interval between a and b in $\mathbb{R}$.

Truncating and rounding

For any $x \in \mathbb{R}$, floor operator $\lfloor x \rfloor$ specifies the closer integer less than x (truncated value of x), ceil operator $\lceil x \rceil$ states the closer integer bigger than x (truncated value of $x + 1$, unless $\in \mathbb{N}$) while rounding operator $\lfloor x \rceil$ rounds x to the closer integer (truncated value of $x + 0.5$).

Set, tuples, vectors, and matrices

We use curly brackets $\{\dots\}$ for a set of elements (unordered, without repetitions), angle brackets $\langle \dots \rangle$ for tuples (unordered, with repetitions), and round brackets $(\dots)$ for vectors (ordered with repetitions).

We usually denote vectors and matrices with small and capital bold Roman letters respectively. Given a set S and two dimensions $m, n \in \mathbb{N}^+$, we define the space of all n-length vectors with values in S as S^n and the space of all $m \times n$-sized matrices

with values in S as $S^{m \times n}$. Vectors are always interpreted as single-column matrices, *i.e.* $\mathbf{x} \in S^n$ corresponds to $\mathbf{x} \in S^{n \times 1}$. A superscript $\cdot^{\mathrm{T}}$ denotes the matrix/vector transposition, hence $\mathbf{x}^{\mathrm{T}} \in S^{1 \times n}$ while a superscript $\cdot^{\tau}$ denotes the matrix/vector transposition *w.r.t.* the off-diagonal, *i.e.* $\cdot^{\mathrm{T}}$ where columns and rows are flipped. The i-th element of a vector $\mathbf{x}$ is identified by x_i and, analogously, the element in position i, j of a matrix $\mathbf{M}$ is identified by $M_{i,j}$. The i-th row and the j-th column of a matrix $\mathbf{M} \in S^{m \times n}$ are denoted respectively with $M_{i,\cdot} \in S^n$ and $M_{\cdot,j} \in S^m$. The zero vector of length n is denoted by $\mathbf{0}^n$, while the zero matrix of size $m \times n$ is denoted by $\mathbf{0}^{m \times n}$. Analogously, the unit vector of length n (*i.e.* where each entry is one) is denoted by $\mathbf{1}^n$, while the identity matrix of size $m \times n$, *i.e.* $\mathbf{0}^{m \times n}$ where main diagonal entries are 1, is denoted by $\mathbf{1}^{m \times n}$.

Asymptotic classes

We adopt big-O and big-Ω notation for upper- and lower-bounding asymptotic classes respectively. We use the big-Θ notation for both upper- and lower-bounding together. Given two real-valued functions f, g, in formulas we have

$$
\begin{aligned}
f(x) \in O(g(x)) \quad &\iff \quad \exists c, x_0 \in \mathbb{R} \; s.t. \; f(x) < c \cdot g(x), \forall x > x_0 \\
f(x) \in \Omega(g(x)) \quad &\iff \quad \exists c, x_0 \in \mathbb{R} \; s.t. \; f(x) > c \cdot g(x), \forall x > x_0 \\
f(x) \in \Theta(g(x)) \quad &\iff \quad f(x) \in O(g(x)) \text{ and } f(x) \in \Omega(g(x))
\end{aligned}
$$

Complexity classes

We adopt canonical notation for complexity classes. In particular, given a decision problem $\mathcal{A}$, we say that $\mathcal{A} \in \mathbf{P}$, if it is solvable by a deterministic Turing machine in polynomial time and we say that $\mathcal{A} \in \mathbf{NP}$ if it is solvable by a non-deterministic polynomial-time Turing machine (or, analogously, if it is verifiable in polynomial time by a deterministic Turing machine). $\mathcal{A}$ is said to be **NP**-hard if there exists a polynomial reduction from each problem in **NP** to $\mathcal{A}$. A problem which is both in **NP** and **NP**-hard is said to be **NP-c** (or **NP**-complete), since proving its membership in **P** would solve the famous problem $\mathbf{P} =_? \mathbf{NP}$. Problems in **NP** (some of which are introduced in Section 1.5) are considered to be computationally intractable on the average case.

Part I
Underlying concepts and methods

Overview

Chapter 1
Graph theory

I wrote down this first chapter as an introduction to graph theory, mainly focusing on the topics that serve in the rest of the present work. The chapter is initially structured similarly to the lectures in informatics I hold at "Università degli Studi Roma Tre". *The content represents my personal approach to graph theory that takes inspiration from various books and research articles, including [6, 7, 12, 13].*

1.1 Introduction

Like many other areas in (applied) mathematics, it is not surprising that graph theory has been independently discovered many times throughout history. However, conversely *w.r.t.* many of the other branches of mathematics – which were typically motivated by fundamental problems of calculation, motion, and measurements – the problem which led to graph theory development were often little more than puzzles. The origins of the concept of *graph*s and the correspondent study can be traced back to the works of Euler, who mentioned the subject in his writings dealing with the well-known problem of Königsberg bridges [10].

Although Euler's initial problem may have seemed like a playful riddle, it was rooted in the physical world, and its solution provides a clear explanation for why graph theory captivates mathematicians. In fact, like many other applied mathematics fields, graph theory exhibits a remarkable capacity of abstraction from the real world. The surprising variety and depth of its theoretical results make, nowadays, graph theory an appealing and rich subject of study.

However, it took nearly two centuries for the first book on graph theory to be published in 1936 [15][1]. Kirchhoff, Cayley and Sylvester later rediscovered graph theory, with their investigations once again rooted in the realm of physics and chemistry. Kirchhoff's study of electric networks led to the development of fundamental concepts and theorems concerning trees in graphs, while Cayley and Sylvester explored graphs by building an isomorphism with the structure of single molecules, introducing the so-called *chemical graph theory*. Sylvester, in particular, was one of the first researchers who recognise the importance of graphs and their wide applicability, commenting on their combinatorial nature[2]. Another notable contributor

[1] The interested reader can refer to [3] for a book discussing the early stages of graph theory before 1936.

[2] "The theory of ramification is one of pure colligation, for it lakes no account of magnitude or position; geometrical lines are used, but have no more real bearing on the matter than those employed in genealogical tables have in explaining the laws of procreation" (quote is from [12]).

to graph theory was Hamilton, who approached the subject from a puzzle-solving perspective by creating a board game called "Around the World" where the objective was to determine a Hamiltonian (or traceable) path over a graph. Subsequently, the famous *four colour conjecture* gained prominence and has remained an enduring topic of interest ever since.

Within the present century, graph theory continues to experience numerous rediscoveries, particularly notable in its application to real-world problems. From computer science and network analysis to operations research and social network analysis, graphs offer powerful tools for modelling, analysing, and solving a wide range of practical problems, as we will briefly see also in the main contributions discussed in this work. The ability to represent relationships, connections, and dependencies between entities makes graph theory an indispensable tool in numerous fields. In addition, the abstract nature of graph theory enables the study of general principles and properties that are applicable across different domains. Concepts like connectivity, paths, cycles, and graph algorithms have far-reaching implications and find applications in diverse areas of science, engineering, and social sciences; although, a precise description of their applications is out of the scope of the present work.

As a natural consequence of the diverse fields where graph theory developed, a standard notation for graph theoretical objects is lacking. In fact, the choice of names for graph elements often reflects the specific context in which graphs are employed. As an example, when considering a communications network such as an email network, the entities involved are referred to as nodes, aligning with the terminology used in computer science. Conversely, different names may be used for objects in other domains, such as vertices or points in mathematical graph theory, molecular structures in chemistry, flow charts in programming, or human relations in social sciences. This further motivates the presence of Section 1.2 – and more in general the choice of having this as the first chapter of this work – which, despite being very introductory, serves the purpose of introducing a common line for the rest of this work.

1.1.1 Chapter organisation

The rest of this chapter is organised as follows. In Section 1.2 we introduce the basics of graph theory, covering the topics of adjacency (Section 1.2.1), sub-graphs and spanning (Section 1.2.2), and describing some useful operations within two graphs (Section 1.2.3); we continue by describing the different kinds of walks that can be performed along a graph (Section 1.2.4) and the consequent definition of connected graph, connected component, tree, and forest (Section 1.2.5). We close Section 1.2 discussing graph isomorphisms and graph classes (Section 1.2.6) and describing how to equip a graph with weights and, in particular, with colours(Section 1.2.7). Section 1.3 introduces some possible generalisation on graphs, including digraphs and multi-graphs. Section 1.4 delves into the problem of representing a graph, dis-

cussing both advantages and shortcomings of adjacency lists and adjacency matrices. Section 1.5 introduces the problem and the notation related to graph contraction, a topic that is cardinal later in Chapter 3. Section 1.6 describes how it is possible to generate random graphs, with particular attention to the well-known Erdös-Rényi models (Section 1.6.1) that are later used in Chapter 3 for benchmark purposes; a few other models are discussed as well for the sake of completion (Section 1.6.2). Closes the chapter Section 1.7 on random walks over graphs; after basic definitions are provided (Section 1.7.1), reachability in random walks is discussed (Section 1.7.2) and a few important properties for what follows are introduced (Section 1.7.3).

1.2 The basics

Let us consider a finite set V of size n (for this chapter we might assume $V \subsetneq \mathbb{N}$ and, usually, $V = \{0, \dots, n-1\}$). We recall V^2 as the *2-vectors* of V, or Cartesian product $V \times V$:

$$V^2 \cong V \times V = \{(u, v) \mid u, v \in V\} , \tag{1.1}$$

i.e. all the possible vectors of length 2 with elements from V. We denote with $[V]^2$ the *2-sets* of V:

$$[V]^2 = \{\{u, v\} \mid u, v \in V, u \neq v\} , \tag{1.2}$$

i.e. the set of all possible subsets of two distinct elements in V [3]. We denote with $\langle V \rangle^2$ the *2-uple* (or *duples*) of V:

$$\langle V \rangle^2 = \{\langle u, v \rangle \mid u, v \in V\} , \tag{1.3}$$

i.e. the set of all possible (unordered) couples of elements in V.

We define a (simple, undirected) *graph* (on V) as a pair of sets $G = (V, E)$ such that $E \subseteq [V]^2$. Elements $v \in V$ and $e \in E$ are called *vertices* and *edges*, respectively[4]. The usual way to depict a graph is by means of a planar drawing, where vertices are represented by points and edges by lines (see Figure 1.1).

The vertex set and the edge set *w.r.t.* a graph G are denoted as V_G and E_G respectively to remark it (useful when working with multiple graphs, like we do later in Chapter 6); hence we say that $G = (V_G, E_G)$.

We define the *order* of a graph G as the size of its vertex set and the *size* of G as the size of its edge set; we denote them as $n_G = |G| = |V_G|$ and $m_G = \|G\| = |E_G|$ respectively.

---

[3] $[\cdot]^2$ is also denoted by $\mathcal{P}_2(\cdot)$ since it corresponds to a subset of the power set $\mathcal{P}(\cdot)$ where all elements have dimension two: $[X]^2 = \mathcal{P}_2(X) = \{y \in \mathcal{P}(X) \mid |y| = 2\}$

[4] In literature, $v \in V$ and $e \in E$ are often called in various ways, including:

vertex	point	node	junction	0-simplex	entity
edge	line	arc	branch	1-simplex	relation

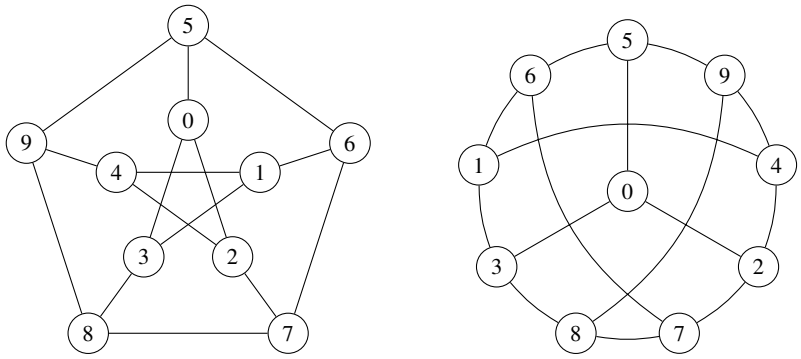


Fig. 1.1: Two representations of the Petersen graph, a well-known (simple, undirected) graph $G = (V, E)$, where $V = \{0, \ldots, 9\}$ and $E = \{\{0, 2\}, \{0, 3\}, \{0, 5\}\}, \{1, 3\}, \{1, 4\}, \{1, 6\}, \{2, 4\}, \{2, 7\}, \{3, 8\}, \{4, 9\}, \{5, 6\}, \{6, 7\}, \{7, 8\}, \{8, 9\}, \{9, 5\}\}$. Representations are equivalent. We see that G has order $|G| = 10$ and size $\|G\| = 15$. The graph is 3-regular, since $\partial(v) = 3, \forall v \in V$.

We (nearly) always adopt the same notation for vertices – using Latin letters u, v, w – and for edges – using Latin letters e, f.

1.2.1 Adjacency, neighbourhoods, and degrees

Given two vertices $u, v \in V_G$, they are said to be *adjacent* – denoted by $u \sim_G v$ – if there exists an edge $e \in E_G$ with $e = \{u, v\}$; in such a case we say that *e joins u* and *v* and that *u* and *v* are *incident* with *e*. In particular, *u* and *v* are called *endpoints* (or simply *ends* or *endvertices*) of *e*. Given two edges $e, f \in E_G$, they are said to be *adjacent* – denoted by $e \sim_G f$ – if they share one endpoint. If $u, v \in V$ (resp. $e, f \in E$) are not adjacent, then we say that they are *independent* and we write $u \nsim_G v$ ($e \nsim_G f$). We can extend the concept of vertex (resp. edge) independence to a set of vertices (resp. edge) if they are pairwise independent; in such case, we refer to such set as *stable*. Conversely, a set of vertices that are mutually adjacent is said to be a *clique*.

The set of vertices adjacent to a given vertex $v \in V$ is called *neighbourhood* of $v \in V$ and it is denoted by

$$N_G(v) = \{u \in V \mid u \sim_G v\} \,. \tag{1.4}$$

The size of $N(v)$ is referred to as *degree* of *v* and it is denoted by $\partial_G(v)$. Maximum, average and minimum degree of a graph are respectively denoted by $\Delta(G)$, $\partial(G)$, and $\delta(G)$, *i.e.*

$$\Delta(G) = \max_{v \in V_G} \partial(v) \,, \qquad \partial(G) = \frac{1}{|G|} \sum_{v \in V_G} \partial(v) \,, \qquad \delta(G) = \min_{v \in V_G} \partial(v) \,. \quad (1.5)$$

If, for some $k \in \mathbb{N}$, we have $\delta(G) = k = \Delta(G)$ $(= \partial(v), \forall v \in V_G)$ then the graph is said *k-regular*. Do note that the graph size and vertices degree are linked together since

$$\|G\| = \frac{1}{2} \sum_{v \in V_G} \partial(v) = \frac{1}{2} \partial(G) \cdot |G| \,. \quad (1.6)$$

Size and degree of vertices naturally extend to vertices sets, *i.e.* given $U \subset V$ we define

$$N_G(U) = \bigcup_{u \in U} N_G(u) \setminus U \quad (1.7)$$

and $\partial_G(U) = |N_G(U)|$.

1.2.2 Subgraphs and span

Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. If $V' \subseteq V$ and $E' \subset E$ we say that G' is a *subgraph* of G – and we write $G' \leq G$. In particular, if $G' \neq G$ and $G' \neq \emptyset$, we say that it is a *proper subgraph* – and we write $G' < G$.

If $G' \leq G$ and $x \sim_G y \Rightarrow x \sim_{G'} y, \forall x, y \in V'$, we say that G' is an *induced graph* of G by V'; in particular, we say that V' *spans* (or induces) G' over G and we write $G' = \langle V' \rangle_G$. By abuse of notation, given $G' \leq G$, we denote the subgraph spanned by G' – we write $\langle G' \rangle_G$ – as the subgraph spanned by V', *i.e.* $\langle G' \rangle_G = \langle V' \rangle_G$; do note that E' has no role in spanning a graph, except for the fact that $E' \subseteq E$ must hold, since $G' \leq G$.

Finally, we say that G' is a *spanning graph* if $G' \leq G$ and $\langle G' \rangle_G = G$, *i.e.* $V' = V$ (this notation is particularly useful when referring to some particular kind of subgraphs, like a tree, see later in Section 1.2.5).

1.2.3 Graphs operators

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs.We define the following operators between them

Graphs union We define the *graphs union* $G_1 \cup G_2$ as the graph

$$G = G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2) \,. \quad (1.8)$$

Graphs Intersection We define the *graphs intersection* $G_1 \cap G_2$ as the graph

$$G = G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2) \,. \quad (1.9)$$

Edges addition Given $F \subset [V_1]^2$, we define the *edges addition* $G_1 + F$ as the graph

$$G = G_1 + F = (V_1, E_1 \cup F) \ . \tag{1.10}$$

Edges removal Given a generic edge set F (typically $F \subset [V_1]^2$), we define the *edges removal* $G_1 - F$ as the graph

$$G = G - F = (V_1, E_1 \backslash F) \ . \tag{1.11}$$

Vertices removal Given a vertex set U (typically $U \subset V_1$), we define the *vertices removal* $G_1 - U$ as the graph

$$G = G_1 - U = \langle V_1 \backslash U \rangle_{G_1} \ . \tag{1.12}$$

We further define $G_1 - G_2$ as $G_1 - V_2$, a well-posed notation since removing vertices V_2 implicitly includes the removal of each edge insisting over V_2.

Graphs join If G_1 and G_2 are disjoint (*i.e.* $V_1 \cap V_2 = \emptyset$), we define the *graphs join* $G_1 * G_2$ as the graph

$$G = G_1 * G_2 = G_1 \cup G_2 + \{\{u, v\} \mid u \in G_1, v \in G_2\} \ , \tag{1.13}$$

i.e. the graph obtained by the union of G_1 and G_2 where all the vertices from G_1 are joined with all the vertices from G_2.

Figures 1.2, 1.3, and 1.4 provide some simple examples of the above-described operation for ease of readability.

In the following, we always drop the subscript $\cdot_G$ when it is clear from the context to which graph we are referring to, *e.g.* we simply write $u \sim v$ rather than $u \sim_G v$.

1.2.4 Traversability of a graph

Underlying the concept of connectivity of a graph $G = (V, E)$ – that we introduce in the upcoming Section 1.2.5 – there are the notions of *walk*, *trail*, and *path*, three progressive refinements of the same object. We define a *walk* $X_{u,v}$ between two nodes $u, v \in V$ as a non-empty succession of pair-wise adjacent nodes between u and v, *i.e.*

$$X_{u,v} = (x_0, x_1, \dots, x_{\ell-1}, x_\ell) \qquad s.t. \qquad \begin{cases} \ell \geq 0 & \\ x_0 = u & \\ x_\ell = v & \\ x_i \in V & \forall 0 \leq i < \ell \\ \{x_i, x_{i+1}\} \in E & \forall 0 \leq i < \ell \end{cases} \tag{1.14}$$

The vertices u and v are said to be the *source* and the *destination* of the walk. We say that the walk *traverse* the graph by *visiting* the edges x_i and *travelling* along the

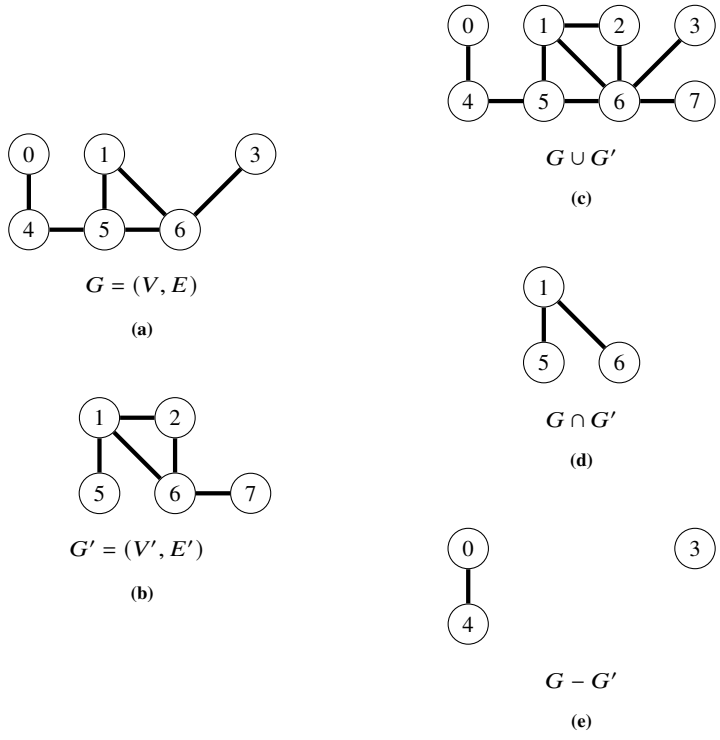


Fig. 1.2: An example of **(c)** union, **(d)** intersection, and **(e)** subtraction of two graphs **(a)** and **(b)**.

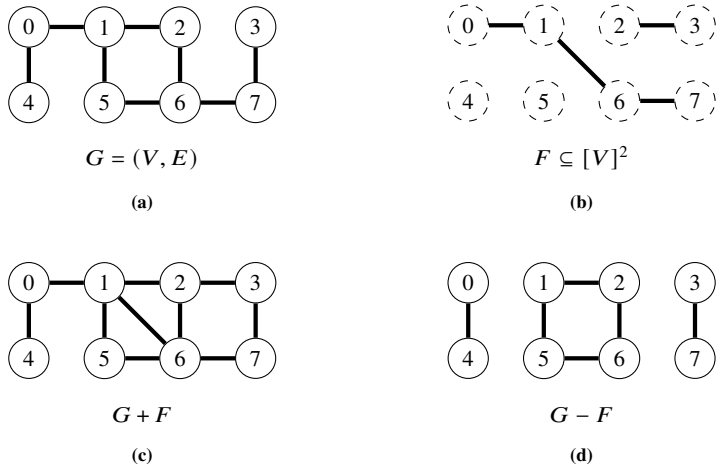


Fig. 1.3: An example of **(c)** addition and **(d)** removal of an edge set **(b)** from a graph **(a)**.

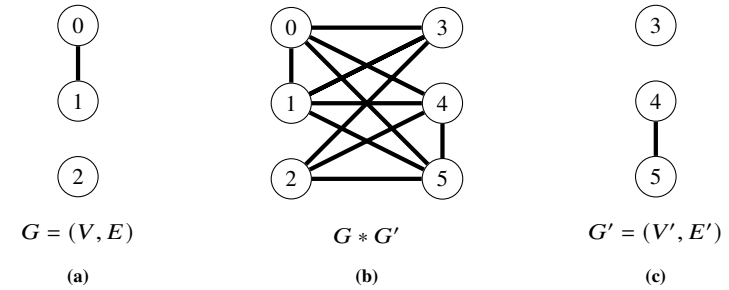


Fig. 1.4: An example of join **(b)** between two disjoint graphs **(a)** and **(c)**.

edges $\{x_i, x_{i+1}\}$ (or traversing them). The number of edges (with multiplicity) within the walk (*i.e.* ℓ) is referred to as *length of* X and it is denoted by $\|X\|$. Do note that $|X| = \|X\| + 1$ and if $\|X\| = 0$ the walk is said *trivial*.

A walk that travels along each edge at most once is said to be a *trail* while, if each vertex is visited at most once, we refer to the walk as a *path*. Paths are typically denoted by $\Gamma_{u,v}$ rather than $X_{u,v}$. In particular, a path is also a trail. Do note that a path also defines a subgraph G' of G where $|G'| = \|G'\| + 1$.

A non-trivial walk $X_{u,v}$ with matching source and target (*i.e.* $u = v$ and $\|X\| > 1$) is often called a *tour*. A tour $\Gamma_{u,u}$ (with abuse of notation) where each vertex and each edge is distinct (apart from $u = v$) is called a *cycle*[5]. Similarly to paths, a cycle can be seen as a subgraph G' of G where $|G'| = \|G'\|$. In particular, we can say that a path is a walk without inner cycles.

Do note that, given a cycle $\Gamma_{u,u}$, then for any vertex $v \in \Gamma_{u,u}$, with $u \neq v$, the cycle is divided into two paths $\Gamma_{u,v}$ and $\Gamma_{v,u}$ which are disjoint apart from the mutual $\{\text{source, destination}\}$ couples. Conversely, given two disjoint paths connecting $u, v \in V$, the concatenation of these two paths forms a cycle. It follows that all the vertices $v \in \Gamma_{u,u}$ admit a cycle $\Gamma_{v,v}$ themselves. A graph that does not admit any cycle is called *acyclic*. Do note that a graph is acyclic if and only if, for any couple of vertices $u, v \in V$, there exists at most a single distinct path $\Gamma_{u,v}$.

Before moving on to the next Section, it is worth (mainly for historical purposes) introducing two couples of famous problems related to graphs traversability. The first couple of problems were introduced back in the work of Euler (see [10]) we referred to in the introduction of this chapter (Section 1.1) and consists in finding a tour that traverses each edge exactly once, or *Eulerian tour*. A derivation of the same problem consists in finding a trail which traverses each edge exactly once, or *Eulerian trail*. A graph that admits an Eulerian (trail) tour is said a *(semi-) Eulerian graph*; many efforts were devoted in early graph theory to characterise Eulerian graphs. Particularly relevant results in this sense are the ones of Hierholzer (see [14]) which provided a linear time algorithm for evaluating Eulerian tours by proving that *a graph is Eulerian if and only if each of its vertices has even degree*.

[5] The requirement that each edge is distinct can be dropped in favour of $\|X\| > 2$.

The second couple of problems, that might seem very close to Eulerian trails and tours are determining whether there exists a path (resp. a cycle) that traverses all the vertices, or a *Hamiltonian path*[6] (resp. *Hamiltonian cycle*). A graph which contains a Hamiltonian path is said to be *traceable*, while a graph containing a Hamiltonian cycle is said to be *Hamiltonian*. Despite being very similar in definition to Eulerian graphs, also solely determining whether a graph is Hamiltonian/traceable is a well-known **NP-c** problem (see [6]).

1.2.5 Connectivity, trees and forests

Given a graph $G = (V, E)$, two vertices $u, v \in V$ are said to be *connected* if there exists at least a path (more in general a walk) $\Gamma_{u,v}$ (*disconnected*, otherwise). The minimum length of a path between u and v (if present) is denoted by $d(u, v)$ and it is said *distance between u and v*; such path is said to be the *shortest path between u and v*. If u and v are disconnected, we assume $d(u, v) = +\infty$, hence we have

$$d(u, v) = \min\left(\{\|\Gamma_{u,v}\|\} \cup \{+\infty\}\right). \tag{1.15}$$

The term distance is used since d defines a metric over the vertex set, holding (i) positivity, (ii) symmetry, and (iii) triangular inequality.

If all the vertices in V are pairwise connected, then the graph is said to be *connected*. A *component* of a graph is a maximal connected subgraph (maximal *w.r.t.* connectivity), *i.e.* a subgraph $G_1 \leq G$ such that for any $G_2 \leq G$ either $G_2 \leq G_1$ or G_2 is disconnected or, analogously, a connected subgraph $G' = \langle U \rangle_G$ spanned by $U \subseteq V$ such that $N(U) = \emptyset$. We denote with $C(G)$ the set of all the components of G. It follows that, if a graph admits a single component (*i.e.* $|C(G)| = 1$), then it is connected; in fact, given two components G' and G'' of G, either $G' = G''$ or $G' \cap G'' = (\emptyset, \emptyset)$.

If for any couple of vertices there exists at most a path connecting them, then the graph is said a *forest*; if there exists exactly one path connecting each couple of vertice, then the graph is said to be a *tree*[7]. It follows that a forest is a graph whose components are trees.

Forests and (in particular) trees have many applications in computer science. As an example, we will use trees in the context of graph spanning in Chapter 3. In fact, in the context of a connected graph (component), the minimal spanning graph whose also connected is called *spanning tree* (an ensemble of spanning trees is called *spanning forest*, see Figure 1.5).

In the following, we assume each graph to be connected if not differently specified. This restriction might seem very strict, however, when we analyse a graph we can typically analyse its component instead (being them independent one from the other).

---

[6] Hamiltonian paths are also said *Traceable paths*.

[7] Analogous definition to forest and tree are, respectively, *acyclic* graph and connected forest or connected acyclic graph.

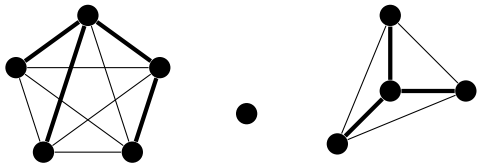


Fig. 1.5: A disconnected graph with its components. For each component, it is highlighted a spanning tree. All the spanning trees together form a spanning forest.

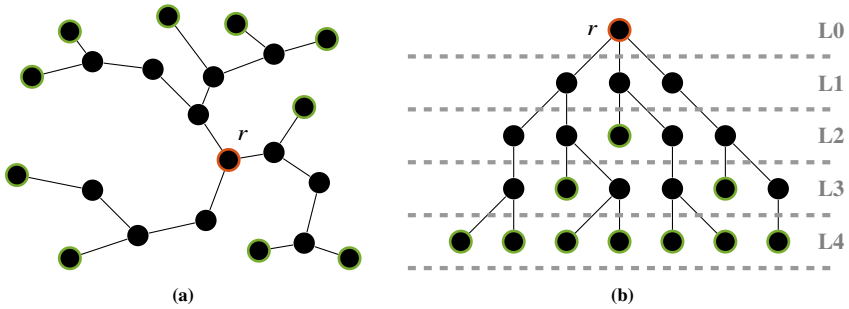


Fig. 1.6: (a) A generic tree T with $|T| = 22$ and **(b)** its corresponding rooted version in r. The root is highlighted in orange (and marked with r), leaves are highlighted in green.

Rooted trees

Given a tree $T = (V, E)$ it is always possible, and sometimes convenient, to select a given node $r \in V$ as a special node; such node is then called the *root* of the tree. A tree with a fixed root is often called *rooted tree* – denoted by T_r. In rooted trees, nodes are organised depending on their distance from r (which is called *height* of a node), hence creating *levels*. In particular, we define the k-th level of T as the set of all nodes with height k. The number of levels ℓ of a tree is called *height* of the tree, *i.e.* $\ell = \max\{d(u, r) \mid u \in V\}$. Nodes with degree 1 are called *leaves* of T and do represent the further points *w.r.t.* the root along a given path. Figure 1.6 depicts a tree along with the corresponding rooted version.

Given a vertex $v \in V, v \neq r$, $\exists!\Gamma_{v,r} = (v, x_1, \ldots, x_{k-1}, r)$. Nodes $x_1, \ldots, r$ are referred to as *ascendant* of the vertex v and, in particular, x_1 is said to be the *parent* (or *father*) of v. Conversely, v is referred to as *children* (or *child*) of x_1, hence the children of v are $N(v) - \{x_1\}$. Do note that a node is a leaf if and only if it has no child and r has no parent.

We define a *d-ary* tree (*binary*, if $d = 2$) as a rooted tree where all vertices have at most d children. Do note that ariety might depend on the choice of the root since the number of children of r is equal to $\partial(r)$, while the number of children of $v \neq r$ is equal to $\partial(v) - 1$. A d-ary tree where each node (up to level $\ell - 1$) has exactly d children is said *complete*. It is easy to prove the height of a generic d-ary graph T is bounded between $\lceil \log_d(|T|) \rceil \leq \ell < |T|$, where the equality holds if (but not only

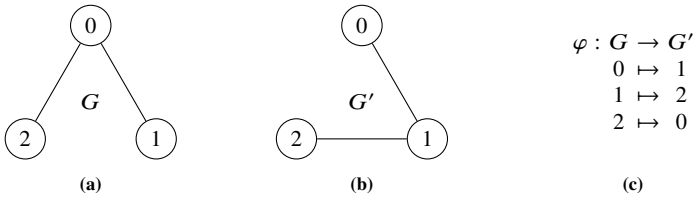


Fig. 1.7: Two sample graphs G and G' such that $G \neq G'$ but $G \cong G'$: **(a)** $G = (\{0,1,2\}, \{\{0,1\},\{0,2\}\})$. **(b)** $G' = (\{0,1,2\}, \{\{0,1\},\{1,2\}\})$. **(c)** A valid isomorphism between G and G'.

if) T is complete, hence making complete d-ary trees the "shortest" graphs (*i.e.* the graphs with smallest height) as possible.

1.2.6 Graphs classification

Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. Graph equality (=), *i.e.*

$$G = G' \qquad \Longleftrightarrow \qquad V = V' \quad \text{and} \quad E = E' \,, \tag{1.16}$$

is an equivalence relation which is typically too strict. In order to relax the equality, we can think of an equivalence relation that only considers the structure of the edges, rather than the "identity" of the vertices. This approach makes sense in many settings since graphs are typically used to study *relations* rather than *entities*. Hence, we can define an equivalence relation that is "equality up to *isomorphism*". More formally, we say that G and G' are *isomorphic* – and we write $G \cong G'$ – if there exists an isomorphism φ between them, *i.e.* a vertex bijection:

$$\begin{aligned} \varphi : G &\to G' \\ v &\mapsto v' \end{aligned} \qquad s.t. \qquad \{u,v\} \in E \iff \{\varphi(u), \varphi(v)\} \in E' \,. \tag{1.17}$$

See Figure 1.7 for an example of isomorphism. In particular, if $G = G'$ then an isomorphism φ is called *automorphism* and the two graphs are said to be *automorphic*.

We define a *graph invariant* as a graph map that assigns equal values to isomorphic graphs, *e.g.* order and size of a graph are invariants. Analogously, we define a set of graphs that is closed under isomorphism as a *graph property* , *e.g.* having a 3-clique as a sub-graph (*i.e.* three vertices mutually connected, see below).

Being $\cong$ an equivalence relation, it makes sense to talk about *classes of graphs* and to study them depending on their similarities, invariants and properties. In particular, we define a few classes of graphs that will be useful later:

Empty graph A graph is said to be *empty* if $G = (\emptyset, \emptyset)$. Do note that the empty graph forms an equivalence class with a single graph.

Trivial graph A graph is said to be *trivial* if it is isomorphic to $G = (\{1\}, \emptyset)$, *i.e.* if its order is $|G| = 1$.

Independent graph A graph $G = (V, E)$ is said to be *n-independent* – and we write I^n – if $|G| = n$ and $E = \emptyset$.

Path graph A graph $G = (V, E)$ is said to be a *n-path* – and we write P^n – if $|G| = n$ and there exists a path that traverses all the nodes and all the edges[8].

Cycle graph A graph $G = (V, E)$ is said to be a *n-cycle*, with $n > 2$ – and we write C^n – if $|G| = n$ and there exists a cycle that traverses all the nodes and all the edges[9].

Complete graph A graph $G = (V, E)$ is said to be *n-complete* – and we write K^n – if $|G| = n$ and $\{u, v\} \in E, \forall u, v \in V$, *i.e.* $\|G\| = \binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n \cdot (n-1)}{2}$.

***r*-partite graph** A graph $G = (V, E)$ is said to be *r-partite* if there exists a r-partition $V_1, \ldots, V_r$ of V[10] such that $\langle V_i \rangle_G \cong I^{|V_i|}$ for each $0 < i \leq r$ (*i.e.* $\{u, v\} \in E \Rightarrow u \in V_i, v \in V_j$ with $i \neq j$). If $r = 2$ we call the graph *bipartite* rather than 2-partite.

***r*-clique** A graph $G = (V, E)$ is said to be an r-clique (biclique if $r = 2$) if there exists an r-partition $V_1, \ldots, V_r$, with $n_i = |V_i|$, such that $G \cong I^{n_1} * \cdots * I^{n_r}$, that is $\forall u, v \in V, \{u, v\} \in E \iff u \in V_i, v \in V_j$ with $i \neq j$. In particular, we denote it as $K_{n_1,\ldots,n_r}$. A biclique $K_{1,n}$ is called n-star. Do note that an n-clique of size n is isomorphic to K^n.

***d*-ary complete tree** We denote with T_d^n the class of d-ary complete graphs of order n (see Section 1.2.5). Do note that $T_1^n \cong P^n$.

Do note that K^1, I^1, P^1, and T_d^1 are all examples of trivial graphs. Analogously, $P^2 \cong K^2 \cong K_{1,1}$ and $K^3 \cong C^3 \cong K_{1,1,1}$ (which is typically called *triangle*).

An exhaustive list of all the isomorphism classes of graphs G of order $1 \leq |G| \leq 6$ (organised by graph size) can be found in [12, Appendix 1]. The analogous list for digraphs G (see later Section 1.3) of order $1 \leq |G| \leq 4$ can be found in [12, Appendix 2]. Finally, a list of all the possible trees T of order $1 \leq |T| \leq 10$ can be found in [12, Appendix 3].

1.2.7 Weighted and coloured graphs

The notion of distance along the graph introduced in Section 1.2.5 and, in particular, in (1.15) is based on the search for a *shortest path* connecting two nodes. This is a classical problem in combinatorial optimisation literature, which can be solved by means of many different algorithms like, *e.g.*, the well-known *Dijkstra algorithm*. Other important problems in graph theory combinatorial optimisation include, *e.g.*,

[8] Analogously, we can say G is connected and the degree of each node is 2 but two nodes which have degree 1.

[9] Analogously, we can say G is connected and the degree of each node is 2.

[10] We recall a r-partition of V being a set of r (non-empty) proper subsets $V_1, \ldots, V_r \subsetneq V$ such that $\bigcup_{i=1}^{r} V_i = V$ and $V_i \cap V_j = \emptyset, \forall 1 \leq i < j \leq r$.

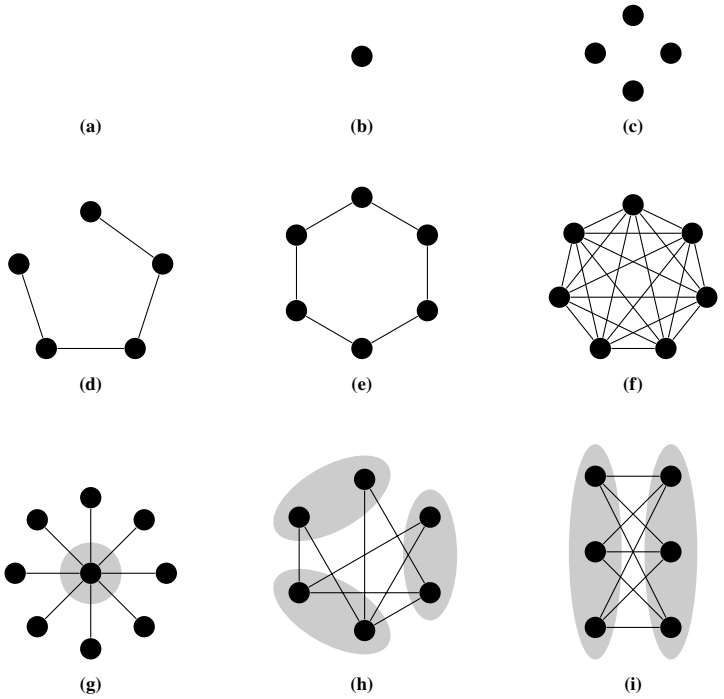


Fig. 1.8: Nine different graph classes. **(a)** The empty graph. **(b)** The trivial graph I^1. **(c)** The 4-independent graph I^4. **(d)** The 5-path graph P^5. **(e)** The 6-cycle graph C^6. **(f)** The 7-complete graph K^7. **(g)** The 8-star graph $K_{1,8}$. **(h)** A 3-partite graph. **(i)** The biclique $K_{3,3}$.

the search for the maximum number of independent paths connecting two nodes, *i.e.* the *maximum flux* problem.

Both shortest-path and maximum-flux problems are typically grounded in the physical world, as they were originally motivated by the representation of some real-world use case. It is the case *e.g.* of a sat-nav searching for a (sub-)optimal path in a road network or the optimisation of the flux along a railway network (see [17, 18] for two historical reviews on those problems). It is natural with these problems (and many others) to extend the binary notion of edges (1 if the edge is present, 0 if the edge is absent) to incorporate numerical values, known as *weights*. Weights represent additional information about the relationships between the graph vertices depending on the underlying problem; as an example, they might represent vertices mutual *distances* (or edges traversal *cost*), when referring to the shortest path problem (often denoted as *minimum cost* problem in this context), or to edges *capacity* or *resistance*, when referring to the maximum flux problem.

More formally, a *edge-weighted graph* is a graph $G = (V, E)$ equipped with an edge-weight function ω_E – and we write $G = (V, E, \omega_E)$ – *i.e.* a mapping $\omega_E : E \to \mathbb{R}$ (or $\mathbb{R}^k$) that maps each edge to one (or multiple) weights in $\mathbb{R}$. Typically, ω_E is chosen as a non-negative function; despite not being mandatory,

it usually makes sense in many contexts (what does a negative capacity represent?) and prevents having negative-valued cycles that might make the process of finding minimal-cost paths get stuck.

The concept of edge-weighting naturally extends to vertices by considering a vertex-weight function, *i.e.* a mapping $\omega_V : V \to \mathbb{R}$ (or $\mathbb{R}^k$). A graph $G = (V, E, \omega_V, \omega_E)$ which is equipped with both an edge weighting ω_E and a vertex weighting ω_V function is said to be a *total weighted graph*. We refer the reader interested in combinatorial problems of minimum cost and maximum flux to [6].

We already said that weights typically restrict codomain to non-negative values. However, in many applications, weights take values in a (countable) finite set C. We define such a weight mapping as a *typological* function as it associates an entity with a *type*[11] from the finite set of types C. Without any loss in generality, we might assume that $C \subseteq \mathbb{N}$, $|C| = c < +\infty$.

In the context of graph theory, elements of C are often referred to as *colours* and typological functions are often called *colourings* and denoted with γ_V and γ_E (for vertex- and edge-colouring, with colours C_V and C_E reps.). A graph where vertices (resp. edges) are equipped with a colouring is said a *vertex-coloured graph* (*edge-coloured graph*). In particular, if both vertices and edges are equipped with a colouring, the graph is said to be *total-coloured*. The term *colour* derives from the fact that such typological information can be represented by depicting nodes and edges in various colours; many examples are present throughout this thesis, including, *e.g.*, the graph representation of museum-like environments in Chapter 4 where colours represent expert knowledge injection.

It is important to notice that, in the classical literature of graph theory, many efforts were devoted in studying *proper graph colourings* – often abbreviated simply as *colourings*. A vertex colouring γ_V is said to be *proper* for the graph $G = (V, E)$ if $\forall u, v \in V$ such that $u \sim v$ we have $\gamma_V(u) \neq \gamma_V(v)$; in other words, γ_V is proper if G is c-partite and it defines a valid c-partition of the vertices. A similar concept holds for *proper edge colourings* γ_E, where $\forall e, f \in E, e \sim f \Rightarrow \gamma_E(e) \neq \gamma_E(f)$. **In the present work, we do not face the proper colouring problems** (for further details, we refer the interested reader to [7]) but rather colourings in the sense of clusterings (see also Chapter 2), *i.e.* we will consider sets of adjacent nodes of the same colour as a single entity within the graphs (cf. Section 1.5).

1.3 Digraph and other generalisations

So far, we defined graphs where the relation $\sim$ established by E is commutative, *i.e.* $u \sim v \iff v \sim u$, anti-reflexive, *i.e.* $v \nsim v, \forall v \in V$, and it is not transitive, *i.e.* $u \sim v$

[11] The concept of *typological function* derives in this context from computer science jargon *categorical function*; however, in order to disambiguate from mathematical *category theory*, we prefer here to adopt the term *type* rather than *category*.

and $v \sim w$ does not necessarily imply that $u \sim w$. The commutative property is the reason why we refer to the graph as *undirected*, since there is no distinction between connecting u with v and v with u. The anti-reflexive property motivates the name *simple*, since no self-connection is allowed.

It is possible to generalise a graph so that these two properties do not hold.

In particular, in order to drop commutativity in a graph $G = (V, E)$, we require $E \subseteq \{\{u, v\} \in V^2 \mid u \neq v\}$ rather than $[V]^2$. In such case, we refer to G as a simple directed graph, or *simple digraph* . In digraphs, an edge $e = (u, v)$ is also denoted $u \to_G v$ (or simply $u \to v$) and the endpoints of e are referred to as *source* and *destination* of the edge.

Along with the definition of neighbourhood, in digraphs, we can also define the concept of in-neighbourhood $N^-(v)$

$$N^-(v) = \{u \in V \mid (u, v) \in E\} \tag{1.18}$$

and out-neighbourhood $N^+(v)$

$$N^+(v) = \{w \in V \mid (v, w) \in E\} \,. \tag{1.19}$$

The corresponding sizes are called in-degree ∂^- and out-degree ∂^+. It is rather clear that $\partial(v) = \partial^-(v) + \partial^+(v)$ and, analogously to (1.6), we have

$$\|G\| = \sum_{v \in V} \partial^-(v) = \sum_{v \in V} \partial^+(v) \,. \tag{1.20}$$

Digraph spanning and digraph operators behave like for regular graphs (cf. Sections 1.2.3 and 1.2.2), with the exception of digraph join; in fact, being edges endpoints no longer commutative, given two digraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we define the digraph (directed) join $G_1 \overrightarrow{*} G_2$ as

$$G = G_1 \overrightarrow{*} G_2 = G_1 \cup G_2 + \{(u, v) \mid u \in G_1, v \in G_2\} \,, \tag{1.21}$$

In particular, we denote $G_2 \overrightarrow{*} G_1$ also as $G_1 \underset{\leftarrow}{*} G_2$ and we denote $G_1 \underset{\leftarrow}{*} G_2 + G_1 \overrightarrow{*} G_2$ as $G_1 \overset{\rightarrow}{*} G_2$. If it is clear from the context that G_1 and G_2 are digraphs, then we refer to $G_1 \overset{\rightarrow}{*} G_2$ adopting the same notation from undirected graphs, *i.e.* $G_1 * G_2$.

Walks, tours, trails, paths, and cycles in a digraph $G = (V, E)$ are naturally bounded to follow the direction of the edges (hence being called directed walks, directed tours, ...). Therefore, it can happen that given two nodes $u, v \in V$, there exists $\Gamma_{u,v}$ while $\Gamma_{v,u}$ does not. However, in order to keep graph connectivity symmetric, we say that two nodes $u, v \in V$ are connected if there exists a path $\Gamma_{u,v}$ in the underlying undirected graph. Conversely, if there exists a directed path from u to v, we say that u is *strongly connected* to v. If u and v are mutually strongly connected we simply say that they are *strongly connected*. It follows that a digraph is said to be *connected* (resp. *strongly connected*) if all of its vertices are *connected* (resp. *strongly connected*).

Given a digraph $G = (V, E)$, if there exists a vertex v such that there exists a directed path from v to $w, \forall w \in V$ (or, analogously from w to $v, \forall w \in V$), then the digraph is said to be *rooted*. In fact, it is easy to see that there exists a spanning tree which can be rooted in v with all the edges directed from parents to children.

More in general, in the context of digraphs, the concept of tree naturally extends also to *direct acyclic graphs* (or DAGs). In DAGs, no directed cycles are allowed; however, given $u, v \in V$, more than a single path $\Gamma_{u,v}$ can coexist (as far as no path $G_{v,u}$ are present). It follows that the underlying undirected graph of a DAG is not necessarily a tree. Consequently, the concept of rooted tree extends naturally to rooted DAGs: here the single root is replaced by a set of roots such that $\forall w \in V$ exists a root v within the set such that v is strongly connected to w.

In order for a graph $G = (V, E)$ to drop anti-reflexivity, we require the ability to have self loops, *i.e.* edges from a vertex v to itself. This is pretty straightforward in digraphs since we can assume $E \subset V^2$; in such case, we refer to it as a *digraph* with self loops (or simply *digraph*). Conversely, since the edges of (undirected) graphs are made of sets rather than vectors, we define a *graph with self loops* by requiring $E \subseteq [V]^1 \bigcup [V]^2$, where $[V]^1$ represents the 1-sets of V, *i.e.* $\{\{v\}, v \in V\}$, where $\{v\}$ denotes a self loop of v.

The last notation (undirected graphs with self loops) is pretty unusual in graph theory and it is canonically preferred to oppose (simple, undirected) graphs with the concept of *multi-graphs*. Intuitively, a multi-graph is a graph where multiple edges are allowed between the same couple of vertices and self loops are permitted. We define a multi-graph as a triple $G = (V, E, \phi)$, where V is the set of vertices, as usual, E is a (finite) set of symbols, and $\phi : E \to \langle V \rangle^2$ is a map that assigns to each edge $e \in E$, its endpoints in $\langle V \rangle^2$.

It is pretty clear how to combine the above definitions to generate a *multi-graph without self loops* (or simple multi-graph), a *multi-digraph*, and all of the other possible combinations[12].

Many other generalisations are possible for a graph; for completeness, let us cite, *e.g.*, *hypergraphs*, where an edge can join more than a couple of vertices, *i.e.* $E \subseteq \mathcal{P}(V)$, or *infinite graphs* (see [7, Chapter 8]), where V and E are not limited to be finite.

1.4 Representing a graph

We already saw in Section 1.2 that graphs are mathematical objects that can be either represented as couples of sets (eventually equipped with a number of functions) or pictorially as points in the space and lines joining them. However, being powerful data structures in computer science it is worth to discuss also how they can be efficiently represented in a computer.

[12] However it is out of the scope of this chapter, since in the following we will only work with graphs, digraphs and multi-graphs.

Let $G = (V, E)$ be a graph with order n and size m. We assume that the vertex set is given by $V = \{1, \dots, n\}$ and hence there is no need to store it. More in general, V is numerable, and hence there exists a bijection between V and $\{1, \dots, n\}$; it is then possible to store V appropriately as a vector and still refer to vertices according to their indices $v \in \{1, \dots, n\}$. In particular, two main approaches exist to store the edge set E:

Adjacency matrix We define the adjacency matrix of G as $\mathbf{M} \in \mathbb{F}_2^{n \times n}$ where $M_{u,v} = 1 \iff u \sim v$. Do note that elements on the main diagonal of $\mathbf{M}$ are zero if the graph is simple and the matrix is symmetric unless G is a digraph. Depending on the average degree $\partial(G)$ (actually on m), it might be beneficial to store the matrix as sparse, since the sparsity[13] of the matrix is given by $1 - {}^{m}/_{n^2}$.

Adjacency lists We define the adjacency lists of G as a n-length vector $\mathbf{A}$ of vertices sets $(a_1, \dots, a_n)$, where $a_u = \{v \in V \mid u \sim v\}$, *i.e.* $a_u = N(u)$. In particular $|a_u| = \partial(u)$ (or $\partial^+(v)$ if G is a digraph), hence making the total size of $\mathbf{A}$ optimal, being equal to $\sum_{u=1}^{n} |a_u| = \sum_{u=1}^{n} \partial(u) = m$.

Each of the two storage approaches presents both vantages and shortcomings. For example, in the common case where $m \ll n^2$, adjacency lists are much more space efficient. Space efficiency comes, however, with a bigger computational time for checking whether two specific nodes $u, v \in V$ are adjacent (the entire a_u should be checked, which costs at least $O(\log n)$ in the case of an ordered representation, and $O(n)$ otherwise) comparing to a nearly-linear cost for checking $M_{u,v}$. Conversely, if the whole neighbourhood of a vertex $v \in V$ is to be explored (like it happens in visits, see [6]), $\mathbf{A}$ is optimal in time, being already $\partial(v)$ in size, while $\mathbf{M}$ (more specifically $M_{v,\cdot}$) always require $O(n)$ steps, regardless of the vertex degree.

Furthermore, in digraphs, while $a_u = N^+(v), \forall v \in V$, extracting the in-neighbourhood is a computationally oversized task in $\mathbf{A}$; it requires, in fact, traversing all the adjacency lists $a_u, u \neq v$. Conversely, in $\mathbf{M}$, both extracting $N^+(v)$ and $N^-(v)$ are pretty straightforward tasks; in fact, they require determining the (truth values of the) u-th row/column respectively of $\mathbf{M}$.

Another important advantage of the matrix representation is the capability of modifying the graph (*i.e.* adding and removing edges) in linear time since it is only needed to modify the actual matrices values. Conversely, requiring to add/remove a value from a set is potentially much more complex in adjacency lists since, depending on the way they are implemented, it might cause the rewriting of the entire set.

However, the best advantage of $\mathbf{M}$ over $\mathbf{A}$, is given by the possibility of evaluating the connectivity of the graph by means of matrix exponentiation. In fact, performing the multiplication $\mathbf{M} \times \mathbf{M}$ in $\mathbb{F}_2$ yields a new matrix $\mathbf{M}^{(2)} \in \mathbb{F}_2^{n \times n}$ where $M^{(2)}_{u,v} = 1 \iff \exists \Gamma_{u,v}$ with $|\Gamma_{u,v}| = 2$. More in general we have:

$$\mathbf{M}^{(k)} = \mathbf{M}^k = \overbrace{\mathbf{M} \times \cdots \times \mathbf{M}}^{k \text{ times}}, \qquad \mathbf{M}^{(k)} \in \mathbb{F}_2^{n \times n} \tag{1.22}$$

where it holds

[13] The sparsity of a matrix is defined as the ratio of the non-zero elements.

$$M^{(k)}_{u,v} = 1 \qquad \Longleftrightarrow \qquad \exists \Gamma_{u,v} \quad \text{with} \quad |\Gamma_{u,v}| = k \ . \tag{1.23}$$

In particular, (1.23) is easy to prove by induction over k, being $\mathbf{M}^{(1)} = \mathbf{M}$ which represents the paths of length 1.

Analogously, performing the exponentiation in $\mathbb{N}$ rather than in $\mathbb{F}_2$ also provides interesting data. It is easy to see that if we build $\mathbf{W}^{(k)}$ as:

$$\mathbf{W}^{(k)} = \mathbf{M}^k = \overbrace{\mathbf{M} \times \cdots \times \mathbf{M}}^{k \text{ times}} , \qquad \mathbf{W}^{(k)} \in \mathbb{N}^{n \times n} \tag{1.24}$$

then $W^{(k)}_{u,v}$ represents the number of distinct paths joining u and v of length k.

Finally, worth of mention is also the representative capability of adjacency matrices in the context of edge-weighted graphs. In fact, in particular, if edge weights are single values – say the weight function being $\omega_E : E \to \mathbb{R}$– we can directly replace the boolean value $M_{u,v}$ with weight value $\omega_E(\{u, v\})$ hence obtaining $\mathbf{M} \in \mathbb{R}^{n \times n}$ (with a proper representation of edges that are not present).

We will use adjacency matrix exponentiation later in Section 1.7.2 and Chapter 6 in the context of random walks. Conversely, we will rely on adjacency lists in Chapter 3 due to its contained space cost (which is particularly important when handling graphs of order $\sim 10^7$ and beyond). It is in fact mandatory to take into account that dense matrix representations are not admissible when considering graphs of order $\sim 10^6$ and beyond, since they correspond to (at least) $\sim 10^{12}$ bit, *i.e.* > 100 gigabytes of memory; conversely, considering a sparse matrix representation lessens the computational advantages due to the constraints added by the implementation which, despite being carefully optimised (see, *e.g.*, SparseBLAS [4]), typically relies on a three-array structure.

1.5 Hard problems and graph contraction

Graph theory has given rise to numerous famous problems, many of which have significant implications in computer science and mathematics. Amongst these problems, a non-negligible class is formed by **NP-c** problems. We have introduced a few so far (more or less explicitly), including:

Hamiltonian paths and cycles Determine whether a graph is Hamiltonian or traceable is a well-known **NP-c** problem with relevant applications in network routing, DNA sequencing, and scheduling algorithms.

Travelling salesman problem (TSP) It is a classic optimisation problem requiring finding the shortest Hamiltonian cycle in a complete weighted graph. The TSP is an **NP-c** problem that has extensive applications in logistics, transportation planning, and resource allocation.

Graph colouring problem Finding a proper minimal colouring (where the size of the colour set c is minimal) is an **NP-c** problem with applications in scheduling,

register allocation in compilers, and frequency assignment in wireless communication.

Independent set problem Detecting the largest set of independent vertices (or analogously the bipartition in V_1, V_2 with that maximises $|V_1|$) is an **NP-c** problem with relevance in areas such as social network analysis, task scheduling, and resource allocation in computer networks.

Clique problem Retrieving the largest clique of a given graph is another **NP-c** problem with applications in social network analysis, data mining, and computational biology.

Many other problems, such as the Steiner tree problem, the knapsack problem, and the vertex cover problem, fall into the **NP** complexity class (see [6] for a detailed description). Solving them by brute force is an unfeasible task even when the graph order is low and many other "simpler" problems are still tough when the order is contained (say $|G| \sim 10^5$). Most of the graph problems present, in fact, challenging computational hurdles, and although efficient algorithms for solving them exactly are not known, various approximation algorithms and heuristics have been developed to find near-optimal solutions in practice.

Until very recently, techniques for solving problems on large graphs were typically developed in the context of sequential algorithms (as also discussed in [5]). Most of these techniques, like depth-first search and Dikjsra algorithm (for which we refer the reader to [6]) are, however, very complex to parallelise (see [19]). Hence, we now present a suitable approach for parallel applications, an implementation of which we will discuss later in Chapter 3.

A common practice when facing a problem on a large graph $G = (V, E)$ is to consider a simplified version of it capable of retaining important connectivity information and preserving the relationships between vertices. Hence, the idea is to consider a "condensed" version of G that captures the essential characteristics of the original graph, allowing for more efficient analysis and computation.

A very common way to compact a graph is via *graph contraction*. Given two adjacent vertices $u, v \in V$, we define the *contraction* of u and v – and we denote it as $G/\{u, v\}$ – as a novel graph $G' = (V', E')$ where u and v are dropped and a new vertex w is added in place of them, such that it retains all the connections that u and v had (without multiplicity). More formally, we have that

$$V' = (V \backslash \{u, v\}) \cup \{w\} \tag{1.25}$$

and

$$E' = \{\{x, y\} \in E \mid x, y \in V \backslash \{u, v\}\} \cup \{\{x, w\}, \forall x \in N_G(\{u, v\})\} \,. \tag{1.26}$$

or, analogously,

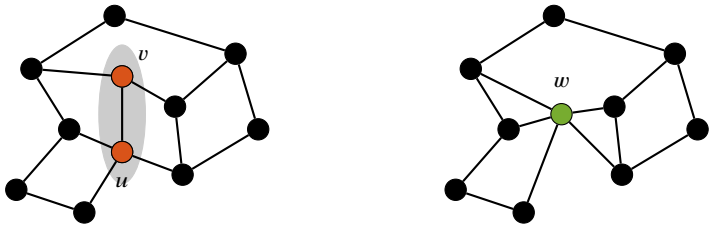


Fig. 1.9: (a) a graph G and **(b)** the corresponding contraction $G' = G/\{u, v\}$. Contracted vertices $u, v \in V_G$ are highlighted in orange and the resulting blended vertex $w \in V_{G'}$ is highlighted in green.

$$G/\{u,v\} = \big(G - \{u,v\}\big) \cup \big(\langle N_G(\{u,v\})\rangle_G * I^1\big)\ .^{14} \tag{1.27}$$

Figure 1.9 shows pictorially a sample of graph contraction. It is worth noticing that in (1.27) we lose reference of the novel vertex w, hence it could be preferable to write $(\{w\}, \emptyset)$ instead of I^1. It is in fact important to highlight that w is somehow derived by u and v as well as edges $\{x, w\}$ are derived by (the combination of) $\{x, u\}$ or (and) $\{x, v\}$. As an example, in the context of weighted graphs, vertices and edges might inherit a combination of the original weights.

By abuse of notation, we can say that the contraction is a procedure $\mathfrak{m}_G(u, v)$ that returns a blended vertex $w \notin V$ from u and v and a set E_w of blended edges joining w to $G - \{u, v\}$ obtained from the incident edges on u and v such that $N_{G'}(w) = N_G(\{u, v\})$. Hence, we can re-write (1.27) as

$$G/\{u,v\} = \Big(\big(G - \{u,v\}\big) \cup \big(\{w\}, \emptyset\big)\Big) + E_w\ , \tag{1.28}$$

with $(w, E_w) \leftarrow \mathfrak{m}_G(u, v)$.

In particular, $\mathfrak{m}_G$ satisfies commutative ($\mathfrak{m}_G(u, v) = \mathfrak{m}_G(v, u)$) and associative ($\mathfrak{m}_G(u, \mathfrak{m}_G(v, w)) = \mathfrak{m}_G(\mathfrak{m}_G(u, v), w)$) properties if the same properties hold for the above-mentioned blending procedures. In what follows, we assume this holds true[15], hence $\mathfrak{m}_G$ can be seen as a *variadic* function, *i.e.* it can be applied to any set of arguments $U = \{u_0, \ldots, u_k\}$, that is $\mathfrak{m}_G(U) = \mathfrak{m}_G(u_0, \ldots, u_k) = \mathfrak{m}_G(\ldots(\mathfrak{m}_G(u_0, u_1)\ldots), u_k)$.

Given a vertex subset $U \subset V$, it is important to note that the variadic nature of $\mathfrak{m}_G$ ensures that the natural generalisation of $G/\{u, v\}$ to G/U is well defined, *i.e.*

---

[14] Do note that $\langle N_G(\{u,v\})\rangle_G$ is somehow a more readable notation to specify a subgraph of G with vertex set $N_G(\{u,v\})$; however, we only require $(N_G(\{u,v\}), \emptyset)$ since we are not interested in mutual connections between them.

[15] This is trivially true for a simple unweighted graph but, more in general, we can distinguish the contraction of u in v from the contraction of v in u, depending on how additional data are generated for w and its corresponding edges However, such an outcome is out of the scope of this work, since we will only consider additive or invariant blendings over weighted graphs, *i.e.* where $\omega_V(w) = \omega_V(u) + \omega_V(v)$ or where $\gamma_V(w) = \gamma_V(u) = \gamma_V(v)$ with similar behaviour on edges.

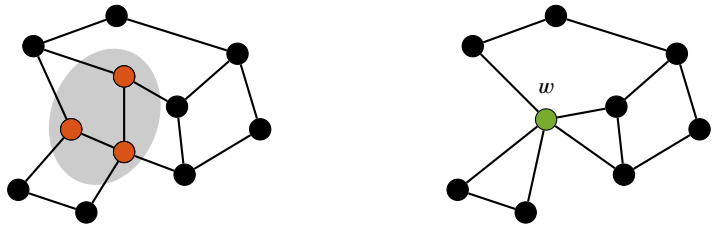


Fig. 1.10: **(a)** a graph G and **(b)** the corresponding contraction $G' = G/U$. The set of contracted vertices $U \subseteq V_G$ are highlighted in orange and the resulting blended vertex $w \in V_{G'}$ is highlighted in green.

$$G/U = \big(G - U\big) \cup \big(\langle N_G(U)\rangle_G * I^1\big)\,, \tag{1.29}$$

or, analogously

$$G/U = \Big(\big(G - U\big) \cup \big(\{w\}, \emptyset\big)\Big) + E_w\,, \quad \text{with} \quad (w, E_w) \leftarrow \mathfrak{m}_G(U)\,. \tag{1.30}$$

It is clear that contraction $G/\{u, v\}$ is a particular case of G/U, where $|U| = 2$.

In particular, we refer to G/U as a *contraction graph* of G and we denote it as $G/U \preceq G$. Equality holds only if $|U| < 2$ while, otherwise, we write $G/U \prec G$ and we define G/U it a *proper* contraction graph of G. Figure 1.10 reports an example of contraction of a set of vertices.

A further problem arises when talking about graph contraction: how to determine which vertex needs to be contracted (and at which time)? The answer mainly depends on what the expected outcome is. We here report the two most common use cases and the corresponding strategy:

induction strategy In order to solve a problem over a large graph G, contraction is used to produce a sequence of condensed graphs $G_1 \succ G_2 \succ \cdots \succ G_k$ (potentially up to $k = n - 1$); the problem is then solved on G_k and the solution is, iteratively, transported up to G, *i.e.* given a solution for G_i a solution for G_{i-1} is evaluated. In this context, vertices to contract are typically determined by picking the two endpoints of a random edge of the graph (edge contraction), picking a vertex and contracting its neighbourhood (start contraction), or picking independent binary-tree subgraphs and contracting progressively their nodes into the root (tree contraction).

group analysis strategy In order to study connectivity-related properties and interactions between (and within) "groups" of vertices in large graphs, contraction is used to produce a condensed graph where each group is represented by a single node. Such groups are typically natural with the problem definition or might arise from *e.g.* some clustering technique. As we discuss in Section 1.2.7, such pieces of information can be sketched as a colouring function γ_V. Hence, it makes sense to consider the contraction induced by γ_V – or *colour contrac-*

tion – where two nodes are contracted whether they are adjacent and share the same colour. Analogously, different notions of colour contraction can be given if colouring is provided to edges, *e.g.*, two vertices are contracted whether the edge joining them is from some specific colour subset $C' \subset C$.

A simple example of the first approach is given by the problem of determining the components of a large graph, where one can contract the graph up to an independent graph and then reconstruct the components by tracing the contraction backwards. A more complex example is discussed *e.g.* in [11], where authors employ graph contraction in the development of a fast parallel algorithm to determine reachability in DAGs.

A number of examples for the second approach are scattered throughout this work and can be found later in Chapter 3 and 4. Simpler examples include, *e.g.*, studying behaviour and interaction between members of a social network sharing some characteristics.

1.6 Generating random graphs

An essential tool when it comes to analysing algorithms and procedures designed for solving problems over graphs is the capability of generating (large) random graphs with particular properties (depending on the underlying problem). Apart from the mere testing task (which is the way we use it in this work and our reason to introduce this topic), it is worth mentioning that the theory of *random graphs* is a subject of its own. In fact, with his pioneering work [8], Erdös introduced the *probabilistic method* as a sophisticated and versatile proof technique: roughly speaking, in order to prove that a given property holds, it is possible to prove that, for any $n \in \mathbb{N}^+$, the probability that a random graph of order n has such probability is strictly positive.

The probabilistic method highlights, amongst the others, the importance of being able to determine how to sample graphs from the set $\mathcal{G}^n$ ($\mathcal{G}^{n,m}$) of all the possible graphs of order $n \in \mathbb{N}$ (and size $m \leq n^2$). In the rest of this section, we will primarily introduce the early definition of Erdös–Rényi graph and we will later describe a few other models of random graphs.

1.6.1 Erdös-Rényi graphs

One of the most well-known approaches for generating random graphs is the Erdös-Rényi (ER) model, extensively analysed by Paul Erdös and Alfréd Rényi in the [9]. ER graphs are characterised by their simplicity and stochastic nature.

In particular, two similar variants of the model were introduced in [9], one simpler to generate and the other more suitable for testing applications due to the higher level of randomness:

Uniform model Given $n, m \in \mathbb{N}$ with $0 \leq m \leq n^2$, we define the *uniform random model* – and we denote it as $\mathrm{ER}_{n,m}$ – as the operation of sampling a graph from $\mathcal{G}^{n,m}$ uniformly at random, *i.e.* all of the $\binom{M}{m}$ possible graphs with order n and size m are equally probable within the model, with M being the number of possible edges $M = \binom{n}{2}$. In other words, in a graph $G \leftarrow_{\$} \mathrm{ER}_{n,m}$ we have that a given edge has probability m/M of being present and probability $M-m/M$ of being absent (see [1] for the analogous model in multi-graphs, a simpler context since possible graphs are M^m rather than $\binom{M}{m}$).

Binomial model Given $n \in \mathbb{N}$ and $p \in [0, 1]$, we define the *binomial random model* – and we denote it as $\mathrm{ER}_{n,p}$ – as the operation of sampling a n-order graph where each possible edge is present with probability p, being the decision concerning the different edges independent. In other words, $\mathrm{ER}_{n,p}$ model samples a specif graph $G \in \mathcal{G}^n$ with probability $p^m(1-p)^{\binom{n}{2}-m}$, where $m = \|G\|$ (and all the graphs with equal size are equally probable).

In particular $\mathrm{ER}_{n,p}$, can be seen as a generalisation of $\mathrm{ER}_{n,m}$ where m is not fixed, but rather a random variable sampled from the binomial distribution $\textsc{Bin}(\binom{n}{2}, p)$. Hence, we have

$$\mathbb{P}\left\{\|G\| = m \mid G \leftarrow_{\$} \mathrm{ER}_{n,p}\right\} = \binom{\binom{n}{2}}{m} p^m (1-p)^m \,, \tag{1.31}$$

since it corresponds to achieving m successes within a sample from $\textsc{Bin}(\binom{n}{2}, p)$. We recall the expectation of $m \leftarrow_{\$} \textsc{Bin}(\binom{n}{2}, p)$ being $\mathbb{E}[m] = \binom{n}{2} p$, meaning that $\mathrm{ER}_{n,m}$ and $\mathrm{ER}_{n,p}$ are equivalent *on average* when

$$p = \frac{m}{\binom{n}{2}} \sim \frac{2m}{n^2} \qquad \text{or} \qquad m = \binom{n}{2} p \sim \frac{n^2 \cdot p}{2} \,. \tag{1.32}$$

It is not surprising that in [9] results are formulated in terms of $\mathrm{ER}_{n,m}$ rather than $\mathrm{ER}_{n,p}$ since they are simpler to prove but still holds for the binomial model (on average). In particular, components size and structure of random graphs are studied *w.r.t.* the relation between n and m, providing threshold functions for many interesting properties, like *e.g.*, the fact that $m_0 = n^{k-2/k-1}$ is a threshold between the probability of finding a tree subgraph of order k in $G \leftarrow_{\$} \mathrm{ER}_{n,m}$ is very high ($m > m_0$) or very low ($m < m_0$) [9, Corollary 1 of Theorem 1].

The two most fascinating and important results concern the number of components and their size (and will be later used in Section 3.4 for benchmark creation). In particular, all of the following properties hold *with overwhelming probability* with $n \to \infty$:

Components size The ratio $m/n \sim 1/2$ (that is $p \sim 1/n$) is a threshold on the largest component size $\rho_{n,m}$, that divides between having $\rho_{n,m} \in O(\log n)$ and $\rho_{n,m} \in \Omega(n - \log n)$[16]. In particular, there is a "double-jump" behaviour:[17]

- $p < 1/n$ The graph is made of many distinct small components of size bounded by $O(\log n)$;
- $p = 1/n$ The largest component of the graph has size $\Theta(n^{2/3})$;
- $p > 1/n$ The graph presents a single "giant" component that connects all but at most $O(\log n)$ vertices.

Global connectivity The ratio $mn/\sim \log n/2$ (that is $p \sim \log n/n$) is a threshold for the graph connectivity. In particular, the threshold is sharp (meaning that behaviour changes for ϵ small at will) on both having isolated vertices and components of size greater than one, which combined implies:

- $p < (1-\epsilon) \cdot \log n/n$ The graph contains isolated vertices;
- $p > (1+\epsilon) \cdot \log n/n$ The graph is connected.

Finally, it is important to note that generating ER graphs is relatively straightforward and can be done efficiently. This is particularly true for the uniform model, where it is simply needed to extract m (or $1 - m$ when $m > 1/2\binom{n}{2}$) different couples of vertices. Conversely, for what concerns the binomial model it is always possible to first extract a value for m which is compatible with the distribution built over n and p and then to run the uniform model with parameters n and m.

1.6.2 Other random graph models

In addition to ER graphs, several other random graph models have been developed to capture different characteristics and phenomena. We report a few notable examples for completeness.

Watts-Strogatz model Proposed by the homonym researchers in [20], this model generates graphs with small-world properties (*i.e.* most of the nodes have limited mutual distance). It starts with a regular lattice and then rewires a fraction of the edges to create shortcuts, resulting in high local clustering and short average path lengths.

Barabási-Albert model Introduced by homonym authors in [2], this model generates graphs that exhibit scale-free properties (*i.e.* where node degree distribution follows a power-low distribution, with a little number of central "hubs" and many low degree "satellites"). Graphs are built according to a preferential attachment

[16] Do note that these two results are complementary since they can be obtained one from the other considering $p = n/2$ and $p = 1 - n/2$.

[17] The proven result in [9, Theorem 9b] is much more complex, since it precisely characterises the size of the giant component for all $c = p/2 \sim m/n$ in terms of the solution of the equation $x(c)e^{-x(c)} = 2ce^{-2c}$, however this level of detail is out of the scope of this work.

mechanism, where new vertices are added, and each new vertex connects to existing vertices with a probability proportional to their degree.

Stochastic block model (SBM) The SBM is a generative model that captures the community structure observed in many real-world networks. It assigns vertices to different *blocks* or communities and determines the probability of edge existence based on the blocks to which the vertices belong.

Configuration model It generates random graphs with a specified degree sequence. It starts with a set of free endpoints (or *stubs* in the jargon), where each stub represents a vertex and its degree. The stubs are then randomly paired to form edges, resulting in a random graph that matches the given degree sequence.

These models, among others, provide versatile tools for generating random graphs that exhibit specific properties or mimic real-world phenomena. Their broad application ranges from complex systems to study network dynamics and, as it is done later in Chapter 3, to evaluate the performance of graph algorithms.

1.7 Random walks and Markov chains

We close this chapter by introducing random walks on graphs, a fundamental concept in graph theory which have applications in various fields, including computer science, physics, and social network analysis. Briefly, a random walk can be seen as a memory-less process that outputs a walk on a graph. In particular, random walks can be analysed using the underlying probabilistic concept of Markov chain, a powerful tool which is very closely entangled with random walks. We here only scratch the surface on the topic of Markov chains, for which we refer the reader to [16].

In particular, we will use random walks in Chapter 6 to build a recommender system that works on the combination of two complete weighted graphs joined by a bipartite graph between the two. A Markov model *with memory* is also used in Chapter 4 in order to design a digital twin simulating visitor trajectories in museum-like environments.

1.7.1 Basic definitions

A *random walk* on a n-order graph $G = (V, E)$ is a stochastic memory-less process that starts at an initial vertex $v_0 \in V$ and moves along a walk $X = (x_0, x_1, \dots)$ where at each step t, the element x_t is determined probabilistically according to some distribution P over $N(x_{t-1})$, that is

$$\mathbb{P}\{x_t = v \mid x_{t-1} = u, \{x_s = u_s\}_{s=0}^{t-2}\} = \mathbb{P}\{x_t = v \mid x_{t-1} = u\} = P(u, v) \ . \tag{1.33}$$

Hence, the choice of the next vertex is typically determined by a probability distribution that assigns probabilities to the (outgoing) edges of x_{t-1} that does depend

neither on t nor on $x_0, \ldots, x_{t-2}$. Such distribution probabilities are typically built over some property of the graph vertices or edges. In particular, we refer to a

Simple random walk In *simple random walks*, the probability of moving from u to v (assuming the walk is in u) is given by

$$P(u,v) = \begin{cases} 1/\partial(u) & \text{if } u \sim v \\ 0 & \text{otherwise} \end{cases} \tag{1.34}$$

($1/\partial^+(u)$ in the case of a digraph).

Weighted random walk In the context of a positively weighted graph (say edge-weighted, but it follows analogously in vertex-weighted), a *weighted random walk* assigns the probability of moving from u to v as proportional to the weight ω_E of the edge joining them, *i.e.*

$$P(u,v) = \begin{cases} \frac{\omega_E(\{u,v\})}{\sum_{w \in N(u)} \omega_E(\{u,w\})} & \text{if } u \sim v \\ 0 & \text{otherwise} \end{cases}, \tag{1.35}$$

where the otherwise clause can be dropped if we consider the complete graph $G' = K^n$ with weight function $\omega_{E'}(\{u,v\}) = \omega_E(\{u,v\})$ if $u \sim_G v$ and 0 otherwise. It follows that simple random walks are a particular case of weighted random walks where $\omega_E(e) = 1, \forall e \in E$.

Biased random walk *Biased random walks* are a further generalisation of weighted random walks that builds P depending on a generic *bias* (or preference) towards certain vertices or edges. Such a bias can be based on various factors, such as vertex degrees, edge weights, or specific attributes of the vertices.

In particular, the probability distribution P represents *per se* a weight function for G if interpreted as a digraph (since typically $P(u,v) \neq P(v,u)$) and it is typically represented by a weighted adjacency matrix $\mathbf{P} \in \mathbb{R}^{n,n}$ (cf. Section 1.4) which is *right-stochastic*, *i.e.* $\sum_{v \in V} P_{u,v} = 1, \forall u \in V$. In probability field, such a matrix is typically referred as a *transition matrix* and can be used as the basis of a *Markov chain* (Markov process discrete both in time and space), *i.e.* a process defined on a set of elements $\mathcal{X}$ (= V) and a transition matrix $\mathbf{P}$ that samples a sequence of random variables X_i from $\mathcal{X}$ (starting with $X_0 = x_0$) such that, $\forall x, y \in \mathcal{X}$, we have:

$$\mathbb{P}\{X_{t+1} = y \mid X_t = x, \{X_s = x_s\}_{s=0}^{t-1}\} = \mathbb{P}\{X_{t+1} = y \mid X_t = x\} = P_{u,v}. \tag{1.36}$$

The relationship within (1.33) and (1.36) (and in particular between random walks and Markov chains) is evident, henceforth, in the following, we will refer to them interchangeably[18].

[18] This assumption is merely for sake of simplicity since in the present work we will not deepen in the probability theory behind random walks. There is indeed a deep difference between the two concepts, as a random walk is more likely a sample from a Markov chain rather than *a* Markov chain itself. From the probabilistic point of view, we can say that Markov chains model a type of random walks. However, we refer to [16] for a deepen discussion on this topic.

1.7.2 Evolution of a random walk

It is clearly a matter of interest to study the evolution of a random walk and, in particular, to be able to describe precisely the *reachability* (or reach probability) within a Markov chain, *i.e.* the probability that a random walk starting on u (i) is on vertex v at time t or (ii) has reached vertex v within time t.

We can state the first task as determining $\mathbb{P}\{x_t = v \mid x_0 = u\}$. In particular, it is worthwhile to notice that if $t = 1$ trivially we have

$$\mathbb{P}\{x_1 = v \mid x_0 = u\} = P(u, v) = P_{u,v} \tag{1.37}$$

and if $t = 2$ we have

$$\begin{aligned}\mathbb{P}\{x_2 = v \mid x_0 = u\} &= \sum_{w \in V} \mathbb{P}\{x_2 = v \mid x_1 = w\}\mathbb{P}\{x_1 = w \mid x_0 = u\} \\ &= \sum_{w \in V} P(w, v)P(v, u) \\ &= \sum_{w \in V} P_{u,w}P_{w,v} \\ &= (\mathbf{P} \times \mathbf{P})_{u,v} = (\mathbf{P}^2)_{u,v}\end{aligned} \quad , \tag{1.38}$$

where the first equality holds due to the memory-less property (1.36). Hence, it is easy to see by induction that

$$\mathbb{P}\{x_t = v \mid x_0 = u\} = (\mathbf{P}^t)_{u,v} = {P^t}_{u,v} \, , \tag{1.39}$$

where we are denoting $\mathbf{P^t} = \mathbf{P}^t$. The similarity between (1.39) and (1.22) is pretty straight, in particular in the case of weighted graphs.

1.7.3 Basic properties

Markov chains and random walks can have several important properties. we here report a few we will need in what follows:

Irreducible A Markov chain is said to be *irreducible* if, $\forall u, v \in V \exists t > 0$ such that ${P^t}_{u,v} > 0$, hence requiring that each node is reachable in finite time from each other vertex. It is straightforward to observe that a Markov chain is irreducible if and only if the underlying graph is (strongly) connected.

Aperiodic Let $\mathcal{T}(v) = \{t \geq 1 \mid {P^t}_{v,v} > 0\}$ be the set of times when a random walk starting on $v \in V$ is allowed to return to it. We define the *period* of a vertex $v \in V$ as the greatest common divisor of $\mathcal{T}(v)$ and we say that a Markov chain is said to be *aperiodic* if all the states have period 1 (*periodic*, otherwise). In particular, it is important noticing that, if the chain is irreducible, then the period

is equal for each node[19]. If the chain is irreducible, it is then sufficient to show the existence of two tours of coprime length to prove that the chain is aperiodic. More in general, a Markov chain is aperiodic if the underlying graph is not bipartite. Finally, it is trivial to notice that a chain is aperiodic if the underlying graph admits self loops (and at least one is present).

Positive recurrent Given a random walk X, we define the *hitting time* of a vertex $v \in V$ as the first time t for which $x_t = v$. We define a Markov chain as being *positive recurrent* if the expectation of such a t is finite for every node. In particular, if the chain is irreducible, then it is positive recurrent (see [16, Lemma 1.13]).

Stationary distribution A probability distribution $\boldsymbol{\pi}$ on V is said *stationary* with regards to $\mathbf{P}$ if it is a left invariant, *i.e.* $\boldsymbol{\pi}^\mathrm{T} \times \mathbf{P} = \boldsymbol{\pi}^\mathrm{T}$.

Ergodic A Markov chain is said to be *Ergodic* if it is irreducible and aperiodic. In particular, an Ergodic Markov chain admits a unique stationary distribution $\boldsymbol{\pi} \in \mathbb{R}^n$ to which any random walk (regardless of the starting point) approaches for sufficiently large times[20]. In other words, we have that

$$\mathbf{P}^t \xrightarrow{t \to \infty} \boldsymbol{\Pi}\,, \tag{1.40}$$

where $\boldsymbol{\Pi} = \mathbf{1}^n \times \boldsymbol{\pi}^\mathrm{T}$.

References

[1] T. Austin, R. Fagen, W. Penney, and J. Riordan. "The number of components in random linear graphs". In: *The Annals of Mathematical Statistics* 30.3 (1959), pp. 747–754. DOI: 10.1214/aoms/1177706204 (cit. on p. 25).

[2] A.-L. Barabási and R. Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (Oct. 1999), pp. 509–512. DOI: 10.1126/science.286.5439.509 (cit. on p. 26).

[3] N. Biggs, E. K. Lloyd, and R. J. Wilson. *Graph Theory, 1736–1936*. Great Clarendon Street, Oxford: Oxford University Press, 1986. ISBN: 0 19 853916 9 (cit. on p. 3).

[19] Since the chain is irreducible, then $\forall u, v \in V, \exists a, b \in \mathbb{N}^+$ such that $P^a{}_{u,v} > 0$ and $P^b{}_{v,u} > 0$. Then $\ell = a + b \in \mathcal{T}(u) \cap \mathcal{T}(v)$. Furthermore, $\forall t \in \mathcal{T}(u)$, we have that $k \cdot t + \ell \in \mathcal{T}(v), \forall k \in \mathbb{N}$. Hence $\gcd(\mathcal{T}(v))$ divides $t, \forall t \in \mathcal{T}(u)$, and therefore $\gcd(\mathcal{T}(v))$ divides $\gcd(\mathcal{T}(u))$, *i.e.* $\gcd(\mathcal{T}(v)) \leq \gcd(\mathcal{T}(u))$. Analogously, we can show that $\gcd(\mathcal{T}(u)) \leq \gcd(\mathcal{T}(v))$, hence proving that $\gcd(\mathcal{T}(v)) = \gcd(\mathcal{T}(u))$.

[20] A proof of the existence and the uniqueness of the stationary distribution can be found in [16, Section 1.5]. A bound on the convergence of (1.40), along with a proof of the convergence can be found in [16, Theorem 4.9]. Some estimations on the time needed for approaching $\boldsymbol{\pi}$, or *mixing time*, are discussed in [16, Section 4.5]. Recalling these concepts here would require also introducing many other concepts, including, *e.g.* the *total variation* of two distribution probabilities, which is out of the scope of the present work.

[4] L. S. Blackford, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, et al. "An updated set of basic linear algebra subprograms (BLAS)". In: *ACM Transactions on Mathematical Software* 28.2 (2002). NIST:50982, pp. 135–151 (cit. on pp. 20, 215).

[5] G. E. Blelloch, M. Reid-Miller, and K. Tangwongsan. *Graph Contraction and Connectivity*. Lecture notes in "Parallel and Sequential Data Structures and Algorithms". LECTURES : 16, 17. 2012 (cit. on pp. 21, 68).

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. 2nd. ISBN: 978-0-262-03293-3. The MIT Press, 2001. ISBN: 0262032937 (cit. on pp. 3, 11, 16, 19, 21, 117).

[7] R. Diestel. *Graph theory 3rd ed*. Vol. 173. 33. Springer-Verlag, 2005, p. 12 (cit. on pp. 3, 16, 18).

[8] P. Erdös. "Graph Theory and Probability". In: *Canadian Journal of Mathematics* 11 (1959), pp. 34–38. DOI: 10.4153/CJM-1959-003-9 (cit. on p. 24).

[9] P. Erdös, A. Rényi, et al. "On the evolution of random graphs". In: *Publ. Math. Inst. Hung. Acad. Sci* 5.1 (1960), pp. 17–60. DOI: 10.1515/9781400841356.38 (cit. on pp. 24–26).

[10] L. Euler. "Solutio problematis ad geometriam situs pertinentis". In: *Commentarii academiae scientiarum Petropolitanae* 8 (1741), pp. 128–140 (cit. on pp. 3, 10).

[11] S. Guattery and G. L. Miller. "A Contraction Procedure for Planar Directed Graphs". In: *Proc. 4th ACM Symp. on Parallel Algorithms and Architectures*. SPAA '92. San Diego, California, USA: ACM, 1992, pp. 431–441. ISBN: 089791483X. DOI: 10.1145/140901.141935 (cit. on pp. 24, 67).

[12] F. Harary. *Graph Theory*. Reading, Massachussetts: Addison-Wesley publishing company, 1969 (cit. on pp. 3, 14).

[13] T. Harju. *Lecture notes on graph theory*. University of Turku, Finland, 2011 (cit. on p. 3).

[14] C. Hierholzer and C. Wiener. "Ueber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren". In: *Mathematische Annalen* 6.1 (Mar. 1873), pp. 30–32. DOI: 10.1007/bf01442866 (cit. on p. 10).

[15] D. Konig. *Theorie der endlichen und unendlichen Graphen*. Leipzig, Germany: BG Teubner-Verlag, 1936 (cit. on p. 3).

[16] D. A. Levin and Y. Peres. *Markov chains and mixing times*. Vol. 107. American Mathematical Soc., 2017. ISBN: 1470429624 (cit. on pp. 27, 28, 30).

[17] A. Schrijver. "On the history of the shortest path problem". In: *Optimization Stories*. EMS Press, Jan. 2012, pp. 155–167. DOI: 10.4171/dms/6/19 (cit. on p. 15).

[18] A. Schrijver. "On the history of the transportation and maximum flow problems". In: *Mathematical Programming* 91.3 (Apr. 2014), pp. 437–445. DOI: 10.1007/s101070100259 (cit. on p. 15).

[19] J. Shun, Y. Gu, G. E. Blelloch, J. T. Fineman, and P. B. Gibbons. “Sequential random permutation, list contraction and tree contraction are highly parallel”. In: *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2014, pp. 431–448. DOI: 10.1137/1.9781611973730.30 (cit. on p. 21).

[20] D. J. Watts and S. H. Strogatz. “Collective dynamics of ‘small-world’ networks”. In: *Nature* 393.6684 (June 1998), pp. 440–442. DOI: 10.1038/30918 (cit. on p. 26).

Chapter 2
Machine learning

This second chapter introduces, with no claim of completeness, some basic concepts on machine learning. The field I am trying to cover is so extended that more than a book would be required to tackle it suitably (many of which, e.g. [28, 33, 52] can be found in the reference section). However, I organised it in order to highlight the minimum necessary contents to navigate through the reading of this work unscathed.

2.1 Introduction

Within the last few decades, we have constantly tried to capture the essence of human intelligence in order to imbue it in some mechanical process. In fact, the field of artificial intelligence (AI) traces its roots back to the mid-20th century, with one of the earliest influential works in the field being the seminal paper "Computing Machinery and Intelligence" by British mathematician and logician Alan Turing [59]. The paper presented the concept of the *Turing Test*, a test to determine whether a machine can exhibit intelligent behaviour indistinguishable from that of a human. It is worth noting that, while Alan Turing is often considered the founder of AI (being an early exponent of the theory that the human brain is in effect a digital computer, see [30]), there were earlier contributions and works that paved the way for the field. For instance, the development of formal logic, symbolic reasoning systems, and early computational devices in the early 20th century all played a role in shaping the concepts and ideas that led to the emergence of AI as a field of study.

Hence, we can see AI as the multidisciplinary field that focuses on the development of intelligent systems capable of performing tasks that typically require human intelligence. These tasks include problem-solving, pattern recognition, decision-making, and natural language processing. The term multidisciplinary is here more actual than ever since AI encompasses a wide variety of branches, including machine learning, natural language processing, computer vision, robotics, biology, and expert systems and its applications permeate most of the fields of science.

In this setting, Machine Learning (ML) has emerged as a powerful field that focuses on the development of algorithms and models enabling computers to learn from data and improve their performances without being explicitly programmed. Instead of relying on handcrafted rules and instructions, ML algorithms learn patterns and relationships directly from the data by making use of statistical techniques, optimisation methods, and mathematical models. ML is then capable of making predictions, classifying data, discovering patterns, and performing decisions according to the model they are learned on.

ML has gained significant attention in recent years due to several factors. First, the exponential growth of data generated in various domains, such as social media, healthcare, finance, and e-commerce, has created a need for automated methods to extract meaningful insights and knowledge from this vast amount of information. Second, advances in computational power and storage capabilities have made it feasible to process and analyse large datasets. Finally, machine learning techniques have demonstrated remarkable success in solving complex problems, leading to significant advancements in areas such as image and speech recognition, natural language processing, and autonomous vehicles.

2.1.1 Mathematics and machine learning

The scope of this chapter is to provide an introduction to the fundamental concepts and techniques in machine learning, later used in this work. In fact, if graph theory is nowadays considered a proper field of mathematics (and we required Chapter 1 due to the huge discrepancy in notation), conversely, some mathematicians still look at machine learning suspiciously, as a subject proper to computer scientists only. The present work, however, humbly provides a few examples explaining why mathematicians too should be well aware of the machine learning world: Chapter 5, in particular, shows how machine learning can serve mathematical modelling and Chapter 6 provides an example of how machine learning techniques can naturally arise from strong mathematical concepts like information entropy and biased random walks on graphs.

Revising the worlds of Deisenroth, Faisal, and Ong (see [33]), too often ML is taught and used as a set of black-box methodologies, often neglecting the strong underlying mathematical backbone in favour of a "simpler" ready-to-use toolbox interpretation[1]. In fact, numerous mathematical fields play a crucial role in ML hence further motivating the approach of mathematicians to it:

Linear algebra Fundamental in ML, the field of linear algebra deals with vectors, matrices, and their operations. Concepts such as vector spaces, linear transformations, eigenvectors, and eigenvalues are essential for understanding and implementing algorithms like principal component analysis (PCA), linear regression, and matrix factorisation methods [21].

Calculus Calculus is essential for optimisation algorithms used in machine learning, such as gradient descent. Concepts like derivatives and gradients are used

[1] "As machine learning becomes more ubiquitous and its software packages become easier to use, it is natural and desirable that the low-level technical details are abstracted away and hidden from the practitioner. However, this brings with it the danger that a practitioner becomes unaware of the design decisions and, hence, the limits of machine learning algorithms. [. . .] For historical reasons, courses in machine learning tend to be taught in the computer science department, where students are often trained in Programming languages, data analysis tools, large-scale computation and the associated frameworks, but not so much in mathematics and statistics and how machine learning builds on it."

for finding optimal points or directions in a function's landscape. Differential calculus enables the analysis of functions and their rates of change, while integral calculus is used for measuring accumulated quantities [27].

Probability theory It provides the tools for modelling uncertainty and making predictions in ML. In particular, the Bayesian approach to probability allows the interpretation of ML methods under the concept of statistical inference [52].

Statistics Tools and techniques from statistics are crucial for making inferences and drawing conclusions from data. It encompasses concepts such as hypothesis testing, confidence intervals, and regression analysis. Statistical techniques are employed for model selection, model evaluation, and analysing the significance of results in machine learning [42].

Optimisation theory One of the main aspects of ML is the ability to find optimal solutions to problems. In ML, optimisation algorithms are used in model parameter selection, to minimise loss functions, and to maximise objective functions. Techniques like gradient descent, convex optimisation, and stochastic optimisation are employed to find optimal solutions efficiently [26].

Information theory It quantifies the amount of information contained in data and provides measures of information content, such as entropy and mutual information. It plays a crucial role in tasks such as data compression, feature selection, and measuring the effectiveness of learning algorithms [52].

Differential equations They are used to model dynamic systems and phenomena. In machine learning, differential equation models, such as ordinary differential equations and partial differential equations, are employed in areas like time series analysis, dynamical systems modelling, and deep learning. In Chapter 5, *e.g.*, we revise the literature on how differential equations can be used to build the so-called physics-informed neural networks (PINNs).

Discrete mathematics The vast field of discrete mathematics encompasses combinatorics, graph theory, and logic, among other areas. Combinatorics is relevant to analysing the combinatorial properties of data, while logic forms the basis for reasoning and rule-based systems. Discrete mathematics is essential for understanding algorithms and data structures used in machine learning [28, 33].

Cryptography [2] Cryptography and data security play a key role in allowing the usage of data for model training, while retaining a certain level of confidentiality, integrity, and authenticity of data and models (see *e.g.* [56]). It is, for example, the case of federated learning, that employs cryptographic protocols and techniques (mainly related to multi-party computation) to ensure that sensitive data remains encrypted and secure during the training process [50].

Graph theory Last but not least, graph theory provides a framework for representing and analysing relationships and dependencies in data. Graph-based algorithms, such as graph neural networks and *PageRank*, are used for tasks like recommender systems, social network analysis, and knowledge graph representation. In the present work, we analyse and introduce different approaches in machine learning related to graph theory, like cascaded localisers based on colour cluster

[2] Concurrently, it is worthy of mentioning also the contribution of machine learning to cryptography (see *e.g.* [54]), a different field to which the candidate is particularly attached.

representations of museum-like environments in Chapter 4 and a drug recommender system based on biased random walks over a complete knowledge graph in Chapter 6.

Hence, this chapter is motivated by the natural need for a bridge between the mathematical world and the field of computer science, a task that we clearly cannot hope to achieve in these few pages. For this reason, we here provide the formalisation of the concepts that are strictly necessary for the reader, and we further refer to the (much more) suitable books on this topic (*e.g.* [33, 41, 42, 52]).

2.1.2 A gentle overview of machine learning

Machine learning can be broadly categorised into three main families of approaches depending on how data are structured and used in the models:

Supervised learning In supervised learning, the algorithms learn from labelled examples to make predictions or classify novel (unseen and un-labelled) data. Examples of supervised learning include the well-known problem of handwritten digit recognition [34] and email spam classification.

Unsupervised learning Unsupervised learning deals with finding patterns or structures in un-labeled data. It is the case, *e.g.* of density estimations and community detection.

Reinforcement learning Conversely to the previous two approaches, reinforcement learning involves the concept of agent that typically learns to interact with a given environment. The typical scope is the maximisation of a reward signal. Famous examples include game playing (like the ability to play chess, Go, and video games) and autonomous driving.

Many complex approaches, however, do not properly fit into this three-fold categorisation. It is the example, *e.g.*, of anomaly detection tasks, that can be tackled in many different ways, including semi-supervised learning approaches where supervised and unsupervised approaches are mixed.

Concurrently, machine learning approaches are also classified according to the underlying task they have been developed for. Some example include

Regression It consists in predicting a continuous value or quantity based on input features. Albeit there exist examples of unsupervised regression, the typical regression model is supervised. Linear regression, polynomial regression, and support vector regression are examples of regression techniques.

Classification The objective of classification is to label input instances within predefined classes or categories. Binary classification involves, *e.g.*, distinguishing between two classes, while multi-class classification involves assigning instances to multiple classes. Classification tasks are usually supervised techniques since the corresponding unsupervised approach is often referred to as clustering (see below). Examples include logistic regression, decision trees, and support vector machines.

Density estimation The goal of density estimation is to provide a model for the distribution of the dataset. Density estimation models are unsupervised in nature and, sometimes, they are referred also as unsupervised regression. Kernel density estimation, Gaussian mixture models, and self-organising maps are commonly used approaches.

Clustering Clustering tackles the problem of grouping similar instances together based on their intrinsic characteristics or similarity measures. They are the unsupervised counterpart of classification tasks since aim to discover patterns and structures in data without *a priori* labelling. k-means, hierarchical clustering, and DBSCAN are some common clustering techniques.

Dimensionality reduction Models in dimensionality reduction try to reduce the number of input features or variables while preserving essential information. Dimensionality reduction techniques, such as Principal Component Analysis (PCA) and t-distributed Stochastic Neighbour Embedding (t-SNE), help in visualising and compressing data while retaining key patterns.

Anomaly detection Anomaly detection concerns the identification of rare or unusual instances in a dataset, *i.e.* data points that differ significantly from the norm. Despite the existence of supervised approaches, anomaly detection is mainly focused on unsupervised solutions. One-class SVM, isolation forest, and auto-encoders are typically used for anomaly detection.

Recommender systems Recommender systems (or recommendation systems, RSs) provide personalised recommendations or suggestions to users based on their preferences and historical data. RSs are typically tackled as semi-supervised approaches; however, both supervised and unsupervised techniques exist. Collaborative filtering, content-based filtering, and hybrid methods are commonly used in recommender systems.

Natural Language Processing (NLP) Finally, NLP deals with text and language data, including tasks like sentiment analysis, text classification, named entity recognition, machine translation, and question-answering systems. Just like RS, NLP can be tackled both as a supervised or unsupervised task.

2.1.3 Chapter organisation

Throughout the rest of this chapter, we primarily focus on supervised learning techniques for regression and classification and unsupervised learning techniques for clustering. In fact, these are the main ML-related topic that we cover respectively in Part III (*i.e.* Chapters 4 and 5) and in Chapter 4. In particular, in Section 2.2 we discuss more in detail the setting of supervised learning, introducing the basic notation for datasets and models that will be used throughout this work; we also briefly introduce the tasks of assessing models performances (Section 2.2.1) and how training in ML is different from regular optimisation techniques (Section 2.2.2). Then, in Section 2.3 we discuss artificial neural networks and their applications in classification and prediction tasks, introducing the early perceptron model (Section 2.3.1),

the classical feed-forward neural networks (Section 2.3.2), and the recurrent neural networks (Section 2.3.3); multi-layer perceptron and long-short term memory models are discussed in greater details. Finally, in section 2.4 we introduce the problem of clustering data and we focus our attention on two different approaches: the centroid-based clustering (Section 2.4.1) and the hierarchical clustering analysis (Section 2.4.2); k-means and agglomerative hierarchical clustering techniques are discussed in more in depth.

2.2 Supervised learning: regression and classification

As we introduced in Section 2.1, ML is about constructing models capable of *learning patterns* from data. In this sense, the term *learning* often implies some form of optimisation of an objective function on such data.

As we described in Section 2.1.2, when it comes to supervised learning, data are equipped with some form of *labelling*, *i.e.* the expected behaviour that the model should reproduce on that specific datum. In the context of ML, data are often called *samples* and labels are also referred to as *ground truth*; a set of samples is called *dataset* and a set of samples with corresponding ground truth is called a *labelled dataset*. Hence, *learning* from a labelled dataset means tuning the model to try reproducing as accurately as possible such samples and, possibly, extend the same behaviour on previously unseen samples. In particular, depending on whether the labels are taken from a finite or infinite set, the corresponding task is referred to as *classification* or *regression* respectively. Do note that, in the following, we usually refer to single-labelled samples, *i.e.* each sample is labelled with a single value; however, the discussion is analogous when multiple labels are considered per sample, hence yielding problems of multi-regression and multi-classification.

More formally, a sample $\mathbf{x}$ represents a list of *features*, *i.e.*, individual measurable properties of the sample. In the following we will usually assume features being values in $\mathbb{R}$, hence a sample is a feature vector of some length n, *i.e.* $\mathbf{x} \in \mathbb{R}^n$. Then, we can define a *dataset* as

$$\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=0}^{N-1}, \qquad \mathbf{x}^{(i)} \in \mathbb{R}^n \ . \tag{2.1}$$

If samples are labelled, we define the corresponding ground truth as

$$\mathbf{y} = \{y^{(d)}\}_{i=0}^{N-1} \ , \qquad y^{(d)} \in \mathcal{L} \tag{2.2}$$

where $\mathcal{L}$ represent some kind of ground truth set (in other words we are assuming that the feature vector $\mathbf{x}^{(i)}$ is *labelled* with value $y^{(d)}$). We further define a *labelled dataset* as the set

$$D = \{\mathbf{X}; \mathbf{y}\} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=0}^{N-1} \ . \tag{2.3}$$

A model is then some sort of procedure $\mathfrak{L}$ that maps a sample to a corresponding value, *i.e.*

$$\begin{aligned} \mathfrak{Q} : \mathbb{R}^n &\to \mathcal{L} \\ \mathbf{x} &\mapsto o \end{aligned} \tag{2.4}$$

In particular, we say that $\mathfrak{Q}$ is a *classifier model* if $\mathcal{L}$ is finite, *i.e.* $|\mathcal{L}| = k < +\infty$, and we say that it is a *regressive model* otherwise, *i.e.* they solve respectively a classification task or a regression task. In what follows, we refer to $\mathcal{L} = \{0, \dots, k-1\}$ for classification tasks and we refer to $\mathcal{L} = \mathbb{R}$ for regression tasks.

It is important to note that, usually, classification tasks are modelled as multi-regression tasks where each class is concurrently assigned to a regression value; the regression output $\mathbf{z}$ can be then normalised to resemble a probability vector $\hat{\mathbf{o}}$ where i-th entry specifies the probability of belonging to i-th class. Consequently, a class o can be determined as the regression output yielding the maximum value, *i.e.*

$$o = \operatorname{argmax}(\hat{\mathbf{o}}) = \operatorname{argmax}(\mathbf{z}) \ . \tag{2.5}$$

A common way to apply normalisation over a regression output $\mathbf{z} = (z_0, \dots, z_{k-1})$ is the so-called *SoftMax* function, where the probability of the i-th class is given by:

$$\hat{o}_i = \operatorname{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=0}^{k-1} e^{z_j}}, \qquad i = 0, \dots, k-1. \tag{2.6}$$

In general, given a dataset $\mathbf{X}$, we denote the evaluation of $\mathfrak{Q}$ over $\mathbf{X}$ as $\mathfrak{Q}(\mathbf{X})$ and we denote the corresponding output with $\mathbf{o}_{\mathcal{L}}$, or simply $\mathbf{o}$ when $\mathcal{L}$ can be derived by the context; in mathematical terms we have

$$\mathbf{o} = \mathfrak{Q}(\mathbf{X}) = \{\mathfrak{Q}(\mathbf{x}^{(i)})\}_{i=0}^{N-1}, \qquad o \in \mathcal{L} \ . \tag{2.7}$$

It is clear from the above description that three questions arise:

1. How can the model $\mathfrak{Q}$ be built?
2. How can we assess the performance of a model *w.r.t.* a labelled dataset?
3. How can we improve on such performance, hence actually *training* the supervised model?

For what concerns the first question, we present in the upcoming Section 2.3 a few models that we later use in the course of this work. In fact, providing a complete overview of all the models in the literature is out of the purpose of this introductory chapter and it is, in general, fairly impossible. We refer the reader to the books [25, 40] for a wider, yet incomplete, list of models.

The second and the third answers are, indeed, closely related and we discuss them in the following two Sections 2.2.1 and 2.2.2.

2.2.1 Assessing models performances

Being able to evaluate the performances of a model requires some form of objective function, or *loss function* in ML jargon. A loss function, in the context of supervised learning, expresses some form of *cost w.r.t.* to how $\mathbf{o}$ is dissimilar from $\mathbf{y}$.

Two examples of loss functions later used in Chapter 5 (one for classification problems and one for regression problems) are the following:

Mean squared error (regression) A classical way to evaluate loss when $\mathcal{L} = \mathbb{R}$ is to employ the (root) mean squared error, *i.e.*

$$\text{MSE} = \frac{1}{N}\sum_{i=0}^{N-1}\left(o^{(i)} - y^{(i)}\right)^2, \qquad \text{RMSE} = \sqrt{\text{MSE}}\,. \tag{2.8}$$

k-cross entropy (classification) A simple way to evaluate loss when $\mathcal{L} = \{0, \dots, k-1\}$ is to use k-cross entropy[3]. Given two discrete distribution probabilities $\mathbf{p}$ and $\mathbf{q}$ over $\mathcal{L}$, we define their k-cross entropy as follows

$$k\text{-cross-entropy}(\mathbf{p}, \mathbf{q}) = \sum_{j=0}^{k-1} p_j \cdot \ln(q_j)\,. \tag{2.9}$$

Here $\mathbf{p}$ plays the role of the ground truth of sample $\mathbf{x}$ and hence corresponds to a vector $\hat{\mathbf{y}}$ where all entries are set to zero but the entry y (*i.e.* the correct class) which is set to 1 and $\mathbf{q}$ is defined as the output $\hat{\mathbf{o}}$ of a normalisation function (see above, cf. SoftMax (2.6)) over the same sample $\mathbf{x}$. Total entropy is further defined as

$$k\text{-cross-entropy}(\mathbf{X}, \mathbf{y}) = \frac{1}{N}\sum_{i=0}^{N-1} k\text{-cross-entropy}(\hat{\mathbf{y}}^{(i)}, \hat{\mathbf{o}}^{(i)})\,. \tag{2.10}$$

A wider list of loss functions can be found in [28, Chapter 3].

2.2.2 Training of a model

Once a loss function is provided, many approaches exist to minimise it by tuning the model parameters. The solution to the minimisation problem is, however, usually intractable analytically[4], hence approximate (iterative) methods are used instead; it

[3] The general definition of *cross entropy* over $\mathbb{R}$ rather than $\{0, \dots, k-1\}$ naturally follows from (2.10) as $\int_{\mathbb{R}} p(x)\ln(q(x))\,dx$, with proper definitions for $p(x)$ and $q(x)$.

[4] There exists a few examples for which the optimal model is known, like the linear regression problem. However, in most of the cases, the number of model parameters is so high that the problem is unfeasible.

is the case of, *e.g.*, *gradient descent*, *momentum method*, and *simulated annealing* (see [28, Chapter 4] for a description of these and other methods).

Adopting an iterative approach allows, however, a key feature in ML that makes it different from regular optimisation processes. In some sense, we can say that optimisation in ML tackles concurrently the minimisation of two types of errors along the iterations (*epochs* in ML jargon): the *training error* and the *generalisation error*.

More formally, when the optimisation is performed, the dataset D is split into three parts being *training set*, *test set* and *validation set*. We assume that all these sets are identically distributed *w.r.t.* D, being the training the largest and the validation the smallest set. Size-wise, they approximately are in correspondence to (70%, 20%, 10%) of the entire dataset and serve different purposes. We denote with *training error*, *generalisation error*, and *validation error* the correspondent evaluation of the loss function.

The optimisation is then executed on the training set only, hence the training error lays the base for the minimisation procedure. However, unlike regular optimisation where the procedure stops when the training error gets stuck in some (local) minima, a further stopping condition is assumed on the monotony of the generalisation error; in other words, if at some point of the procedure, the generalisation error starts growing instead of decreasing (natural fluctuation should be taken into account), then we stop the optimisation. In fact, in such a situation, we say that the procedure is *over-training* and, consequently, the model is *overfitting* the training set, hence losing the capability of generalising results. Analogously, if training stops before reaching the minimum for the generalisation error, then the procedure is *under-training* and, consequently, the model is *underfitting* the training set. Figure 2.1 shows a training example. Conversely, validation set and validation error are used to tune the so-called *hyper-parameters* (learning parameters, model structural parameters, . . .).

The above-described approach to the minimisation problem is what we call a *learning* or *training* procedure. In what follows we do not cover the wide variety of methods and problems related to model training, and we refer the reader to [25] for any further detail.

2.3 Artificial neural networks

Artificial neural networks (ANNs) are computational models inspired by the structure and functioning of biological neural networks, particularly the human brain. ANNs have gained significant attention in the field of ML and have proven to be powerful tools for solving complex tasks across various domains. In particular, they are well-suited for tackling supervised tasks like the ones introduced in Section 2.2. In this section, we explore the foundation of ANNs, starting with the concept of Rosenblatt's perceptron and then delving into feedforward and recurrent neural networks.

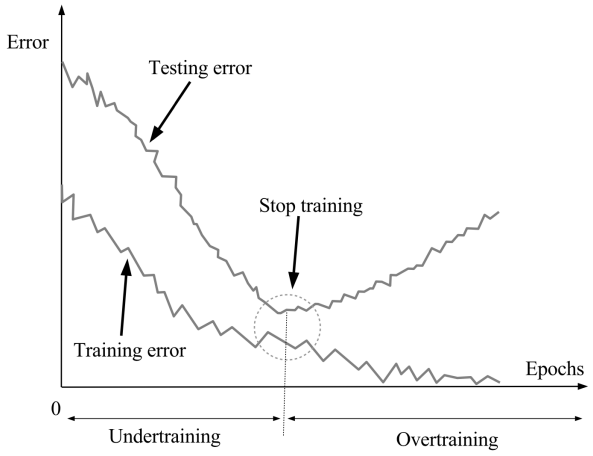


Fig. 2.1: Example of a typical training procedure. Both the training and generalisation error initially lowers. At some point, the generalisation error reaches a minimum point and then starts growing back again. When the minimum is reached, the training procedure should stop, otherwise overfitting occurs.

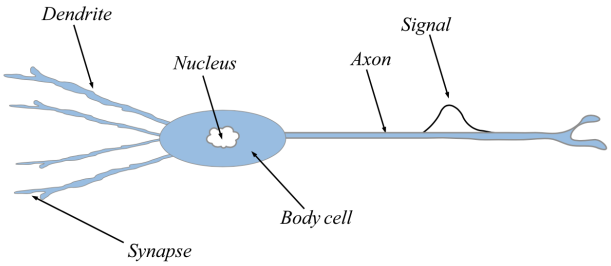


Fig. 2.2: A biological neuron cell. The axon outputs the (electrical) signal that was processed on the input signals conducted by dendrites. Each dendrite is connected to an axon via a synapse.

2.3.1 The perceptron

The perceptron is one of the earliest and fundamental building blocks of artificial neural networks. Introduced by Rosenblatt in [55] as a simplified model of a biological *neuron*, the perceptron consists of a single computing cell with one or more inputs (say n) and a single output. The original idea of Rosenblatt was to mimic the behaviour of the neuron cells, where n electrical signals are transmitted to the *body-cell* through *dendrites* and, if a threshold is reached, then a signal is outputted via the *axon* (see Figure 2.2).

In the mathematical representation, the n signals represent the input features of a sample. Features are processed through a linear combination resembling the dendrites

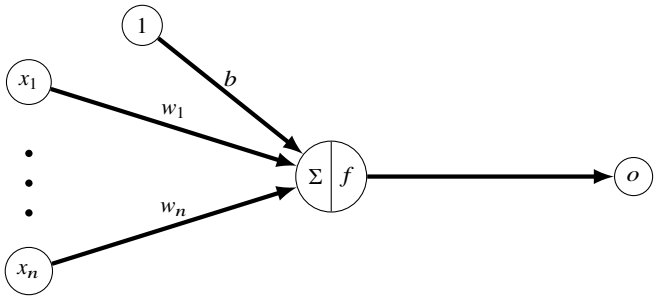


Fig. 2.3: A perceptron, *i.e.* an artificial neuron cell. Input signals are processed by a weighted convolution and the output is the result of the activation function f.

connection, hence resulting in the internal state of the perceptron (*activation value*, in the jargon). Then, the output corresponds to the application of a non-linear function (or *activation function*) to this activation value (see Figure 2.3). To stick to the biological model, historically, the activation function behaved as threshold function on some threshold $\bar{b}$; however, this constraint was quickly dropped in favour of different kinds of (possibly derivable) functions.

More formally, let $\mathbf{x} \in \mathbb{R}^n$ be the *feature vector*, let $\mathbf{w} \in \mathbb{R}^n$ be a vector of *weights* and let $b \in \mathbb{R}$ be a constant (or *bias*). Then the activation value $a \in \mathbb{R}$ is given by

$$a = b + \sum_{i=1}^{n} w_i \cdot x_i \,, \tag{2.11}$$

where the summation in (2.11) is often denoted by $\langle \mathbf{w}, \mathbf{a} \rangle$ or $\mathbf{w}^{\mathrm{T}} \times \mathbf{a}$. Then, given an activation function $f : \mathbb{R} \to \mathbb{R}$, the output $o \in \mathbb{R}$ is derived as $o = f(a)$.

Classical choices for the activation functions are the following (see also Figure 2.4)

Linear function Given a slope coefficient $k \in \mathbb{R}^+$, we canonically define the linear function as the straight line given by

$$L_k(a) = k \cdot a \,, \tag{2.12}$$

with L_1 being denoted by I, or *identity function*.

Heaviside function Also denoted as *step function* centred in 0, the Heaviside function is given by

$$\varphi_0(a) = \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{if } a \leq 0 \,. \end{cases} \tag{2.13}$$

Do note that the Heaviside function resembles as the biological definition of the perceptron with $\bar{b} = -b$.

Sigmoid function Typically, the following simplified version of the sigmoid is used

$$\sigma(a) = \frac{1}{1 + e^{-a}} \in (0, 1) \,. \tag{2.14}$$

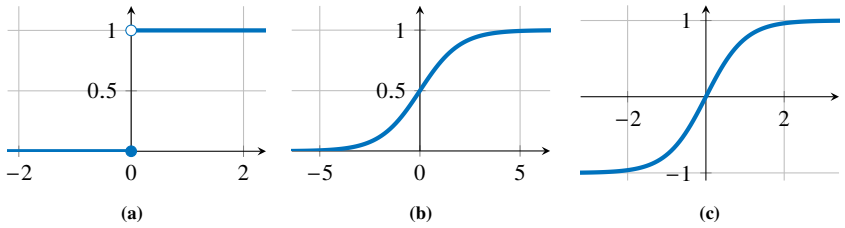


Fig. 2.4: Three widely employed activation functions. **(a)** Sign function. **(b)** Sigmoid function. **(c)** Hyperbolic tangent.

Do note that $\sigma(a) = 1 - \sigma(-a)$ holds.

Hyperbolic tangent We recall the well known hyperbolic tangent being

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \in (-1, 1) \ . \tag{2.15}$$

The perceptron as a classifier

When the Heaviside function is chosen, it is rather clear that, upon the choice of $\mathbf{w}$, the perceptron actually divides the feature space into two subspaces: one where the perceptron evaluates to 1 and the other where the perceptron evaluates to 0. The *division boundary* between these two subspaces is identified by $a = 0$ and it is, in particular, a hyperplane (*i.e.* a space of dimension $n - 1$).

The behaviour we have described is a simple example of binary classifier not requiring a normalisation function (cf. SoftMax 2.6). In fact, the input sample $\mathbf{x}$ is either classified as class zero or as class one directly applying $\varphi_0(b + \mathbf{w}^{\mathrm{T}} \times \mathbf{x})$.

It is easy to see that, despite being a quite limited model, perceptrons can still model interesting datasets, like, *e.g.*, the outcome of the AND and OR logic operators. Problems arise (trivially) when non-linearity in data occurs: as an example, it is easy to prove that a single perceptron is not able to model the XOR logic operator.

From the perceptron to the artificial neural network

Resembling the biological structure of neural networks it is then natural to consider a model where multiple perceptrons are connected together, forming a so-called *artificial neural network* (ANN). In this contest, we usually drop the term perceptron in favour of the broader name *artificial neuron*[5]. We can then represent an ANN as an edge-weighted connected digraph (V, E, ω), where vertices V represent neurones

[5] The terms *perceptron* and *artificial neuron* are often used interchangeably in the literature, as they refer to similar concepts. However, Rosenblatt's perceptron is more likely a specific kind of artificial neuron where the bias is embedded in the Heaviside activation function (step function)

and directed edges E shows how signals travel between them thanks to the weight function ω. In this scheme, input features are directly injected into (some of the) neurones referred to as *input neurones*, that later transmit them to the other neurones through weighted connections. Analogously, the output of the network is taken from (some of the) neurones, referred to as *output neurones*, after the network is evaluated. In particular, input and output neurones are not necessarily disjoint.

Here, the structure of the network, *i.e.* V, E are hyper-parameter of the model. Conversely, given a model V, E, the parameter to be tuned is the weight function ω which, we recall, corresponds to a weight per edge, *i.e.* tunable parameters are in number $|E| \leq |V| \cdot (|V| - 1)$.

ANNs are then classified in terms of their digraph structure and, in particular, can be divided into two main classes depending on the presence or the absence of loops:

Feed-forward neural networks When the underlying graph is a DAG, the resulting network is such that the update of each artificial neuron does not depend on the state of the neuron itself (*i.e.* the network has no feedback). Examples of feed-forward neural networks are *multi-layer perceptrons* (MLP, see below) and *convolutional neural networks* (CNN, see [48]).

Recurrent neural networks When the underlying graph presents some cycles, then there is a feedback effect amongst artificial neurones. Examples of recurrent neural networks include *long-short term memory* (LSTM, see below) and *Hopfield network* (when the graph is homomorphic to K^n, see [44]).

2.3.2 Feed-forward neural networks

Feed-forward neural networks (FNNs) are the most commonly used kind of artificial neural network. Their structure does not present any feedback, hence the corresponding digraph is a DAG (rooted in the input neurones). The input neurones are organised in a set V_0 called *input layer* and they are such that $\partial^-(v) = 0, \forall v \in V_0$. The corresponding values are organised in a vector $\mathbf{a}^{(0)}$ (where, with abuse of notation, we are denoting a relationship between the set of vertices and the vector of values denoting with a_v the value of v and with v_a the vertex whose value is a). It follows that input neurones do not process data in the FNN model and hence serve as a mere input for the rest of the network. Conversely all (and typically sole) the neurones $v \in V$ such that $\partial^+(v) = 0$ are output neurones (otherwise they serve no purpose in the network).

In general, no constraints are given on activation function and each neuron can be equipped with a different one. However, a single activation function $f : \mathbb{R} \to \mathbb{R}$ is usually the case for all non-output neurones (typically sigmoid activation is used). Then the evaluation of a neuron $v \in V \backslash V_0$ is given by

---

and whose task is binary classification. Artificial neurons, on the other hand, are more general and encompass a wider range of computational models used in ANN, including the perceptron.

$$a_v = \sum_{\substack{u \in V \\ (u,v) \in E}} \omega(u,v) \cdot f(a_u) \,, \tag{2.16}$$

and neurones can be evaluated from the input layer onwards. The output provided by an output neuron $u \in V$ is then given by $f'(a_u)$, where $f' : \mathbb{R} \to \mathcal{L}$ is the activation function of the output neuron (typically equal for all of them).

Despite not being a strict requirement of FNNs, usually, other neurones can be organised in *layers* V_l too, depending on their distance l from the input neurones (with correspondent values $\mathbf{a}^{(l)}$). Clearly, this assumption requires that each neuron $u \in V \backslash V_0$ is equally distant from all the input neurones (at least *w.r.t.* the ones it is connected to).

Then let $\ell + 1$ the total number of layers. In particular, layers can be seen as a partition $\{V_0, \dots, V_\ell\}$ of the nodes such $N^-(V_l) \subseteq V_{l-1}$ and $N^+(V_l) \subseteq V_{l+1}$, hence forming a $(\ell + 1)$-partite graph. Do note that, in general, having more than a neuron $v \in V \backslash V_0$ with $\partial^-(v) = 0$ per layer makes little or no sense since such a neuron behaves as a bias for the upcoming layer (*i.e.* it has a fixed value, say 1). However, a single bias neuron per layer is typically included, it is denoted with b_l, and its index in $\mathbf{a}^l$ is 0, *i.e.* $a_0^{(l)} = 1$.

A second common constraint satisfied by FNNs is that V_ℓ (the last layer) is made of all and sole the output neurones. If this is the case, then V_l is referred to as *output layer*, while all the other layers $V_l, 0 < l < \ell$ are referred to as *hidden layers*.

Finally, when the number of hidden layers is greater than one, FNNs are called *deep neural networks* (DNNs). FNNs, and in particular DNNs, have shown remarkable success in various applications, including image and speech recognition, natural language processing, and pattern recognition. Their ability to learn hierarchical representations of the input data through the hidden layers enables them to capture intricate patterns and make accurate predictions.

Single/Multi-layer perceptron

A very common case for a layer V_l in a FNN is when all the neurones (but bias neurones) receive as input all the outputs of the previous layer, *i.e.* $N^-(v) = V_{l-1}, \forall v \in V_l \backslash b_l$. Then V_l is referred to as a *dense layer*. In other words, we have that V_l is dense if $V_{l-1} * (V_l - b_l)$ is the spanned subgraph $\langle V_{l-1} \cup V_l \rangle_G$ of the DAG G.

A FNN where all the layers are dense (but the input) is called a *multi-layer perceptron* (MLP) model and, in particular, its DAG model is defined as

$$\bigcup_{l=1}^{\ell} V_{l-1} * (V_l - b_l) \,. \tag{2.17}$$

The scheme for a generic MLP is provided in Figure 2.5.

Then, if we define $n_i = |V_l| - 1, 0 \leq i < \ell$ and $n_\ell = |V_l|$ (where the term '-1' is due to the bias neuron), an MLP can be safely defined by the $(\ell + 1)$-length vector

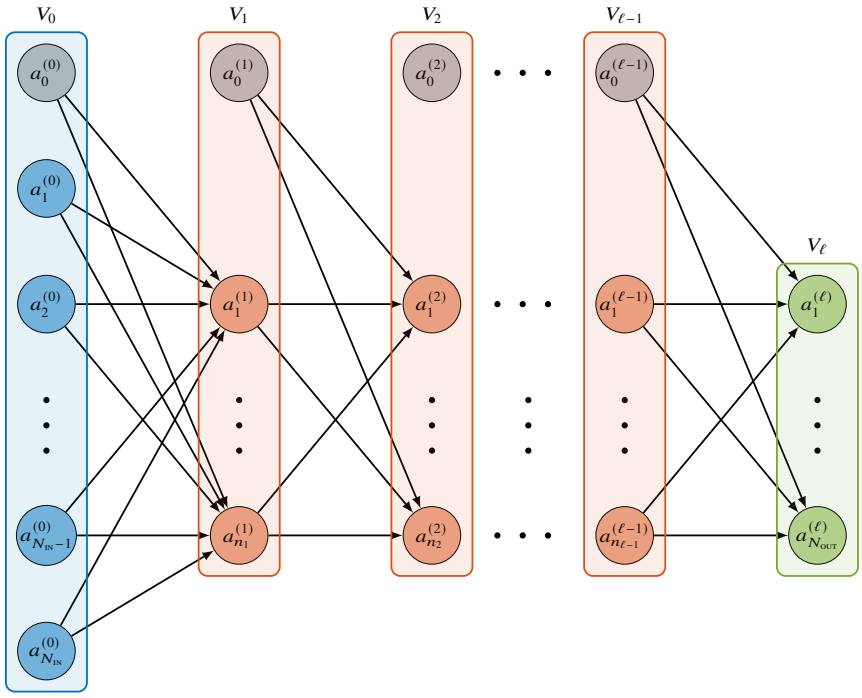


Fig. 2.5: Scheme for a generic MLP. Data are processed from the input layer V_0 (in blue) through hidden layers $V_l, 0 < l < \ell$ (in orange) up to the output layer V_ℓ (in green). Bias neurones are highlighted in grey.

$\mathbf{n} = (n_0, n_1, \dots, n_\ell)$, where we recall that $n_0 = n = |V_0| = |\mathbf{x}| = N_{\text{IN}}$ represents the number of input features and $n_\ell = |V_\ell| = N_{\text{OUT}}$ represent the number of outputs of the network.

MLPs of the form $\mathbf{n} = (N_{\text{IN}}, N_{\text{OUT}})$, *i.e.* where $G = I^{N_{\text{IN}}} * I^{N_{\text{OUT}}}$, are often referred to as *single-layer perceptrons* (SLPs). In particular, it follows that the SLP with the identity activation function is merely a N_{OUT}-convolution of the features. Analogously, an SLP with Heaviside activation function and $N_{\text{OUT}} = 1$ is equivalent to Rosenblatt's perceptron.

It is easy to see that, if $f : \mathbb{R} \to \mathbb{R}$ is chosen as activation function for all the neurones, then (2.16) assumes the following form

$$a_j^{(l)} = \sum_{i=0}^{n_{l-1}} w_{i,j}^{(l)} \cdot f(a_i^{(l-1)}) \,, \tag{2.18}$$

where $\mathbf{w}^{(l)} \in \mathbb{R}^{n_{l-1} \times n_l}$ is the weight matrix of layer l, *i.e.* defined as $w_{i,j}^{(l)} = \omega(u_i, v_j), \forall u_i \in V_{l-1}$ and $\forall v_j \in V_l$. In particular, a powerful way to state (2.18) is the vectorial form, *i.e.*

$$\mathbf{a}^{(l)} = \mathbf{w}^{(l)^{\mathrm{T}}} \times f(\mathbf{a}^{(l)}) \,. \tag{2.19}$$

A remarkable fact in MLPs is given by the universal approximation theorem for neural networks (see [31]) which states that every continuous function mapping $\mathbb{R}$ intervals to $\mathbb{R}$ intervals can be approximated arbitrarily closely by an MLP of the form $\mathbf{n} = (N_{\text{IN}}, N_{\text{H}}, N_{\text{OUT}})$. This result holds for a wide range of activation functions, *e.g.* for the sigmoidal functions.

2.3.3 Recurrent neural networks

While FNNs excel at processing static and independent data, they are limited in their ability to handle sequential or temporal data. Recurrent neural networks (RNNs) address this limitation by introducing recurrent connections, *i.e.* feedback loops. A loop within the graph enforces, in fact, a concept of time within the network since, conversely to FNNs, a neuron can influence itself during the evaluation of the model. Intuitively, this allows information to be stored and propagated through time along the network.

Along with the introduction of a temporal scale on the network, is reasonable to assume a temporal scale also on samples[6]. More in detail, a *time series* is a particular kind of sample whose features present some form of temporal correlation and, hence, are better represented as a matrix $\mathbf{X} \in \mathbb{R}^{N_{\text{IN}} \times T}$ rather than a vector. In particular, we have $\mathbf{X} = (\mathbf{x}_1, \dots \mathbf{x}_T)$, with $|\mathbf{x}_t| = N_{\text{IN}}, \forall t \in 1, \dots, T$, where $\mathbf{x}_t$ represents the *features of time* t. T is called *length of the series* and N_{IN} represents the number of (temporal independent) features: it is the case, *e.g.*, of hourly-measured temperature and humidity within a day ($N_{\text{IN}} = 2, T = 24$, see future work of Chapter 4) or flux and velocity minute-wise reading from a sensor deployed on a highway (see Chapter 5).

Typically, RNNs are then described in terms of a basic building block, or *cell*, that progressively builds an output $\mathbf{h}_t, \forall t \in \{1, \dots, T\}$ taking as input the output $\mathbf{h}_{t-1}$ of the cell at time $t-1$ along with the features $\mathbf{x}_t$ at time t. Here $\mathbf{h}$ is referred to as *hidden cell* and it is not bounded to share its size N_{HID} with the size N_{IN} of $\mathbf{x}$. In fact, even if $N_{\text{HID}} = N_{\text{IN}}$ is widely adopted, N_{HID} should be properly hyper-tuned during the selection of the model.

Figure 2.6 depicts the general form of an RNN in its unfolded representation. The unfolded representation highlights how an RNN is, in reality, a FNN where (i) contiguous layers[7] are organised in cells, (ii) input features are fed little by little after each cell, and (iii) the structure of the cell and its weights are constrained to be equal cell-by-cell.

Despite their effectiveness, RNNs suffer from the vanishing gradient problem, where the gradients used to update the weights can diminish or explode over time, leading to difficulties in learning long-range dependencies. To address this, variants such as *independent RNN* (IndRNN, [49]) and Gated Recurrent Unit (GRU, [29]) were developed, which employ specialised units and gating mechanisms to better

[6] This is not the case with all RNNs, an example being Hopfield networks that are more focused on the evolution from an initial situation provided by the input.

[7] Here layers are enumerated from the first set $\mathbf{x}_1$ of features.

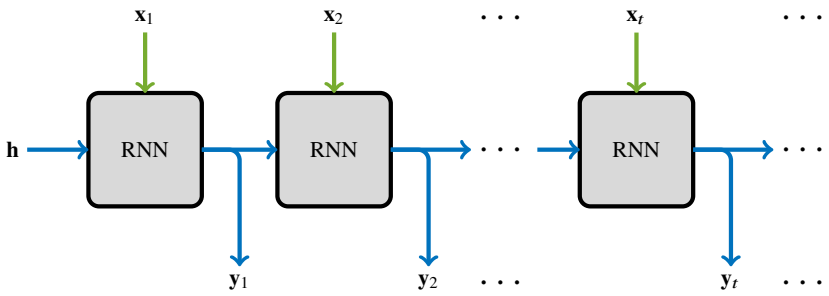


Fig. 2.6: Unfolded representation of an RNN. The cell is depicted multiple times despite being always the same (same structure and same weights). Size $N_{\text{HID}} = |\mathbf{h}_t|$ of the hidden state and size $N_{\text{IN}} = |\mathbf{x}_t|$ of the input features are independent. The hidden state is also the output of the network.

preserve and control the flow of information. In the following, we describe one of the most famous variants of this kind.

Long-Short term memory

Long-short term memory (LSTM) are one of the most famous and widest-used classes of RNN. The LSTM model was introduced by Hochreiter and Schmidhuber in [43] to overcome the traditional problem of vanishing gradient that affected regular RNN, making them capable of capturing long-term dependencies in sequential data.

The idea underlying the LSTM is to divide the output $\mathbf{h}_t$ of each step from the processed data that generates it, hence keeping a sort of *internal memory* $\mathbf{c}_t$ (or *cell state*) of the LSTM itself. Both $\mathbf{h}_t$ and $\mathbf{c}_t$ shares the same dimension N_{HID} (namely $|\mathbf{h}_t| = |\mathbf{c}_t| = N_{\text{HID}}$) which is related to the "length of the memory" (time window) the LSTM is capable to capture. The cell state can be then updated depending on the novel input features $\mathbf{x}_t$ conditioned over the previous output $\mathbf{h}_{t-1}$. A pictorial representation of the unfolded process is given in Figure 2.7 to be compared with the one from RNN (cf. Figure 2.6).

More in detail, the LSTM update is based on four different SLP $\mathbf{g}^{\text{F}}$, $\mathbf{g}^{\text{I}}$, $\mathbf{g}^{\text{C}}$, and $\mathbf{g}^{\text{O}}$, or *gates* (with $|\mathbf{g}^{\cdot}| = N_{\text{HID}}$). Each gate evaluates over both the current-step input features $\mathbf{x}_t$ and the previous-step output $\mathbf{h}_{t-1}$. In particular, the activation function is set to regular sigmoid for all the neurones but the ones from $\mathbf{g}^{\text{C}}$ which are equipped with hyperbolic tangent. More explicitly, we have:

$$\begin{aligned}
\mathbf{g}^{\text{F}}{}_t &= \text{sigmoid}\left(\mathbf{b}^{\text{F}} + \mathbf{W}^{\text{F}} \times \mathbf{x}_t + \mathbf{R}^{\text{F}} \times \mathbf{h}_{t-1}\right), \\
\mathbf{g}^{\text{I}}{}_t &= \text{sigmoid}\left(\mathbf{b}^{\text{I}} + \mathbf{W}^{\text{I}} \times \mathbf{x}_t + \mathbf{R}^{\text{I}} \times \mathbf{h}_{t-1}\right), \\
\mathbf{g}^{\text{C}}{}_t &= \tanh\left(\mathbf{b}^{\text{C}} + \mathbf{W}^{\text{C}} \times \mathbf{x}_t + \mathbf{R}^{\text{C}} \times \mathbf{h}_{t-1}\right), \\
\mathbf{g}^{\text{O}}{}_t &= \text{sigmoid}\left(\mathbf{b}^{\text{O}} + \mathbf{W}^{\text{O}} \times \mathbf{x}_t + \mathbf{R}^{\text{O}} \times \mathbf{h}_{t-1}\right),
\end{aligned} \tag{2.20}$$

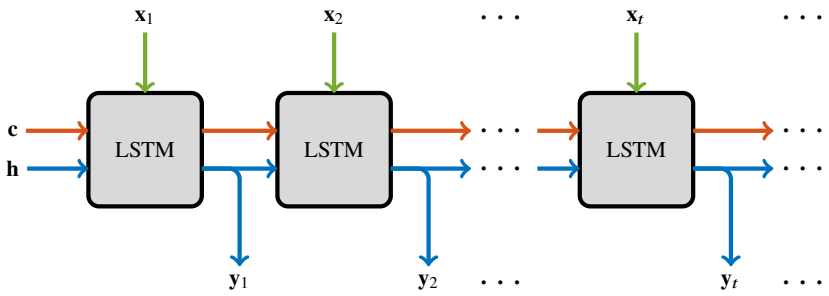


Fig. 2.7: Unfolded representation of an LSTM (cf. Figure 2.7). The cell is depicted multiple times despite being always the same (same structure and same weights). The sizes $|\mathbf{h}_t|$ and $|\mathbf{c}_t|$ of hidden and cell state (respectively) are equal to the output size N_{OUT} (being $\mathbf{h}_t$ the actual output). However, N_{OUT} is independent from the input size $N_{\text{IN}} = |\mathbf{x}_t|$.

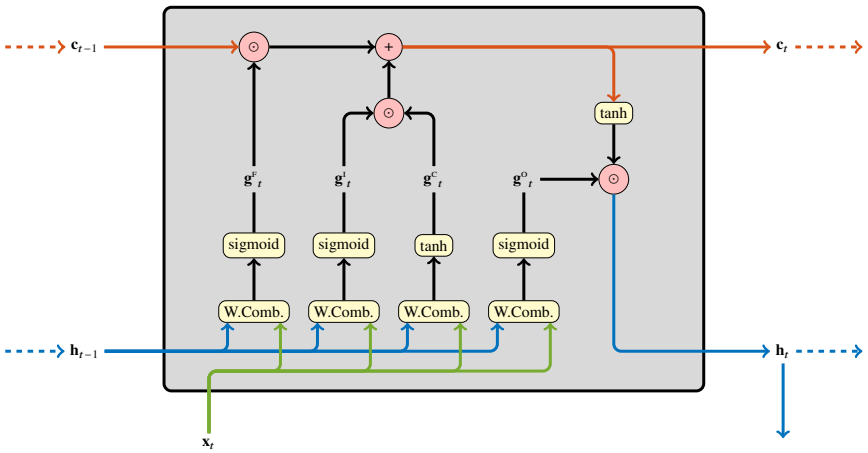


Fig. 2.8: Schematic structure of the LSTM computing unit evaluating input at time t. W.Comb. represents the weighted combination with bias given as $\mathbf{b}^{\cdot} + \mathbf{W}^{\cdot} \times \mathbf{x}_t + \mathbf{R}^{\cdot} \times \mathbf{h}_{t-1}$. All the components are vectors of length N_{HID} but $\mathbf{x}$ which is of length N_{IN}. We recall that $\mathbf{h}_t$ is also the output of the network.

where $\mathbf{W}^{\cdot}$, $\mathbf{R}^{\cdot}$, and $\mathbf{b}^{\cdot}$ are called respectively *input weights*, *recurrent weights*, and *biases* and represents the parameters of the model.

Each gate serves a different purpose, where the first three rule the update of the cell state $\mathbf{c}$ and the fourth determine the next hidden state $\mathbf{h}$ (regulating the cell state contribution). More formally:

$\mathbf{g}^{\mathbf{F}}$ The *forget gate* weakens $\mathbf{c}_{t-1}$ by applying a transformation within the range $(0, 1)$.

$\mathbf{g}^{\mathbf{I}}$ The *input gate* decides how the candidate influences $\mathbf{c}_t$, being a transformation of range $(0, 1)$.

$\mathbf{g}^{\mathbf{C}}$ The *candidate gate* represents the cell input activation that regulates c, being of range $(-1, 1)$.

$\mathbf{g^O}$ The *output gate* decides how $\mathbf{c}_t$ will compose the output, applying a transformation of range $(0, 1)$.

Hence, (see also Figure 2.8 for a pictorial representation) the update rules are given by

$$\mathbf{c}_t = \mathbf{g^F}_t \odot \mathbf{c}_{t-1} + \mathbf{g^I}_t \odot \mathbf{g^C}_t, \qquad \mathbf{h}_t = \mathbf{g^O}_t \odot \tanh(\mathbf{c}_t), \tag{2.21}$$

where $\odot$ denotes the Hadamard (element-wise) product.

2.4 Cluster analysis

As we have seen in the previous sections, perceptron first and ANNs then are capable of performing different kinds of tasks, including learning how to classify samples from a labelled dataset.

However, the process of labelling each sample in the dataset, including the training, test, and validation sets, is typically labour-intensive. As an example in visitor tracking scenarios (see later Section 4.5), accurately labelling the trajectory of a guest requires either following them during their visit or manually analysing data afterwards, both of which are time-consuming solutions that do not always produce the expected accuracy.

More in general, it is not always possible to correctly formalise *a priori* which class set $\mathcal{L}$ the classification problem should rely on: getting back to the museum example, one could be interested in determining typical paths without knowing them *a priori* (see later Section 4.6.3). Additionally, even when a concept of class representative is provided, it can be beneficial to use methods that automatically label samples and update the representative definitions to improve accuracy.

All of the above observations highlight the importance of addressing classification problems using unsupervised methods, leading to the concept of *clustering*. In other terms, we can say that clustering is a fundamental task in unsupervised learning, where similar data samples are grouped together based on their intrinsic patterns or properties.

Naturally, clustering plays a crucial role in exploratory data analysis, pattern recognition, and data compression. Beyond the requirement of grouping similar data into *clusters*, clustering also provides – as principal output or as a byproduct – a natural way to represent each cluster by means of a *representative element*. This representative element can either be a sample from the dataset or an artificially generated one that combines elements belonging to the cluster. In the first case, we refer to this representative as a *medoid* of the cluster, while in the latter case (which is typically more frequent), we refer to it as *centroid*.

The notion of *similarity w.r.t.* data samples is broad in the context of clustering and can involve various characteristics of the features, including geometric position, connectivity, or statistical distribution. Consequently, there is no single, universally applicable definition of *cluster* as it rather depends on how the underlying problem

is defined. Instead, there exists a wide range of clustering models, each with its own peculiarities, characteristics, and assumptions.

In the upcoming sections, we will explore two main clustering approaches: centroid-based clustering in Section 2.4.1 and hierarchical clustering in Section 2.4.2. These approaches provide insights into different ways of addressing the clustering problem and represent important methodologies in unsupervised learning. However, for the sake of completeness, in what follows we first discuss how different clustering models can be classified.

Clustering models

We have already highlighted that no precise definition cluster can be univocally provided. However, we can mainly classify clustering techniques depending on the relation that clusters induce over data.

More formally, given an (un-labelled) dataset D, we can classify a clustering technique $C(D)$ depending on how the clustering $C = (C_1, \dots, C_k)$ divides samples $\mathbf{x} \in D$, where C_i represents the clusters. In particular, we say the clustering being *strict* if $\forall \mathbf{x} \in D$ there exists at most one cluster $C \in C$ such that $\mathbf{x} \in C$. If the existence is guaranteed, *i.e.* $\exists! C \in C$, then the clustering is referred to as *hard*. It is rather clear that a clustering is hard if and only if it defines a proper partition of the data samples. If conversely a sample is allowed in being in more than a single cluster, then the clustering is said *fuzzy* (or *soft*).

Apart from this broad classification, clustering techniques are classified after the definition of cluster they work over. In the following, we provide an extended list (with no claim of completeness) of nine different kinds of clustering techniques along with their corresponding assumptions:

Centroid models Centroid models tackle the problem of clustering as an optimisation problem *w.r.t.* the *centroids* of the clusters. In other words, centroids are determined first and samples are assigned to the cluster whose centroid they are closest to. A popular example of centroid model is the *k-means* clustering [51].

Connectivity models Connectivity models emphasise the connectivity or proximity between data points. They consider data points that are closely connected to be part of the same cluster. Examples include *hierarchical clustering* (HCA) algorithms that create clusters based on some definition of distances between data points.

Distribution models Distribution models assume that the data points in each cluster follow a specific statistical distribution. *Gaussian mixture models* (GMMs) are widely used in this category, where, *e.g.*, each cluster is represented by a Gaussian distribution. The clustering algorithm estimates the parameters of the distributions to assign data points to the appropriate clusters.

Density models Density-based models focus on identifying regions of high density in the data space, which are then considered clusters. Data samples in low-density regions are considered noise or outliers. *Density-based spatial clustering*

of applications with noise (DBSCAN, [38]) is a popular density-based clustering algorithm that groups data points based on their density and connectivity.

Subspace models Subspace models aim to discover clusters that exist in specific subspaces or subspaces with certain characteristics. This approach is useful when different subsets of features contribute to different cluster structures. Subspace clustering algorithms, such as *COBWEB* [39] and *CLIQUE* [22], find clusters in different subspaces of the data.

Group models Group models, also known as partitioning models, partition the data space into a predetermined number of clusters. Each data point is assigned to a single cluster. The simplest examples include *partitioning around medoids* (PAM, [46]) and *fuzzy c-means* (FCM, [37]), where the objective is to minimise the dissimilarity between the data points and the medoids or cluster centroids.

Graph-based models Graph-based models represent the data as a graph, where data samples are nodes and edges represent relationships or similarities. Clusters are identified as connected components or communities in the graph. *Spectral clustering* (see, *e.g.* [53]) and *Markov cluster algorithm* (MCL, [35]) are examples of graph-based clustering approaches. In Chapter 3, we assume having graphs equipped with a clustering (there represented as a vertex-colour map γ) and we discuss the problem of extracting the *contracted* graph where nodes are the representative of such clustering.

Signed graph models Signed graph models take into account positive and negative relationships between data samples, such as friendship or rivalry. These models aim to identify cohesive clusters with positive relationships within the clusters and negative relationships between clusters.

Neural models Neural models use ANNs to perform clustering. A famous example is given by *self-organising maps* (SOM, [47]).

2.4.1 Centroid-based clustering

The first class of models we introduce are the so-called *centroid-based* methods. In centroid-based clustering, samples $\mathbf{x}$ are assumed as points in $\mathbb{R}^n$ and each cluster is represented by a *centroid* $\mathbf{m} \in \mathbb{R}^n$. Given a set of k centroids $\mathbf{M} = \{\mathbf{m}_1, \dots, \mathbf{m}_k\}$, these define a clustering $C = (C_1, \dots, C_k)$ where the points are grouped according to the closer centroid. In mathematical terms, we have that

$$\mathbf{x} \in C_i \quad \iff \quad d(\mathbf{x}, \mathbf{m}_i) < d(\mathbf{x}, \mathbf{m}_j),\ \forall j \neq i\,, \tag{2.22}$$

where the definition is well-posed as far as points (or centroids) are considered in *general position*[8] and where $d : \mathbb{R}^n \to \mathbb{R}$ is some sort of metric (typically Euclidean norm is used). It is easy to see that the centroids define a *Voronoi tessellation* [60]

[8] Being in *general position* is a well-known concept in computational geometry, where points are not provided with integer-valued position but rather with real-valued position. In this setting, it is assumed that the probability of having two couple of equidistant points is negligible.

of the space, and that the points are organised based on their membership in these cells.

If the number k of centroids is fixed, a typical problem related to Voronoi tessellation is the well-known *facility location problem*, which (in this context) requires founding the set of k centroids such that the distance between samples and centroids is minimised, *i.e.* the solution of the minimisation problem

$$\operatorname*{argmin}_{\mathbf{M}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mathbf{m}_i) \ . \tag{2.23}$$

The problem is hard also in many sub-variants (it is *e.g.* **NP**-hard also on graphs and while the supreme metric is adopted), hence many algorithms were developed to find sub-optimal solutions.

If Euclidean distance is considered (*i.e.* $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$), a widely studied variation of (2.23) is the case of the minimisation of *within-cluster sum of squares* (WCSS), *i.e.* the minimisation of the measure

$$\sum_{i=1}^{k} \sum_{\mathbf{x}, \mathbf{y} \in C_i} \|\mathbf{x}, \mathbf{y}\|^2 \ . \tag{2.24}$$

In (2.24), centroids does not figure explicitly and the minimisation problem can be stated in terms of clusters as follows

$$\operatorname*{argmin}_{C} \sum_{i=1}^{k} \sum_{\mathbf{x}, \mathbf{y} \in C_i} \|\mathbf{x}, \mathbf{y}\|^2 \ . \tag{2.25}$$

However, it is easy to see that then (2.25) corresponds to the formulation (2.23) where centroids are bounded to be the geometrical average of the points within their cluster, *i.e.*

$$\mathbf{m}_i = \boldsymbol{\mu}_i = \mu(C_i) = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x_i} \ , \qquad \forall i \in \{1, \dots, k\} \ . \tag{2.26}$$

In particular, we can state (2.25) in terms of (2.26) as follows

$$\operatorname*{argmin}_{C} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x}, \boldsymbol{\mu}_i\|^2 \ . \tag{2.27}$$

k-means model

The optimisation problem described in (2.27) is still **NP**-hard, however efficient sub-optimal algorithms are known. It is the case, *e.g.* of *k-means* model, one of the most famous models in centroid-based clustering. In particular, k-means resemble an

Algorithm 1: $\texttt{kMeans}(D, k) \mapsto C$

Input: A dataset D of size N and a number of clusters k
Output: A clusterisation C of D suboptimal for (2.27)

foreach $(i \in \{1, \dots, k\})$ **do**
  $C_i \leftarrow \emptyset$;
  $C'_i \leftarrow \emptyset$;

// Randomly assign each sample to a cluster
foreach $(i \in \{1, \dots, N\})$ **do**
  $j \leftarrow_{\$} \{1, \dots, k\}$;
  $C'_j \leftarrow C'_j \cup \{\mathbf{x}_i\}$;

// Repeat until convergence
while $(C_i \neq C'_i, \exists i \in \{1, \dots, k\})$ **do**
  // Update new clusters
  foreach $(i \in \{1, \dots, k\})$ **do**
    $C_i \leftarrow C'_i$;
    $C'_i \leftarrow \emptyset$;

  // Evaluate new centroids
  foreach $(i \in \{1, \dots, k\})$ **do**
    $\mathbf{m_i} \leftarrow {}^{1}/_{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x_i}$;

  // Evaluate new clusters
  foreach $(i \in \{1, \dots, N\})$ **do**
    $d^* \leftarrow \|\mathbf{x}_i - \mathbf{m_1}\|$;
    $j^* \leftarrow 1$;
    foreach $(j \in \{2, \dots, k\})$ **do**
      if $(d^* > \|\mathbf{x}_i - \mathbf{m_j}\|)$
        $d^* \leftarrow \|\mathbf{x}_i - \mathbf{m_j}\|$;
        $j^* \leftarrow j$;
    $C'_{j^*} \leftarrow C'_{j^*} \cup \{\mathbf{x}_i\}$;

return $(C'_1, \dots, C'_k)$;

iterative procedure that progressively evaluates a (sub-)optimal Voronoi tessellation by making use of the centroid constraint provided in (2.26).

Given a set of centroids $\mathbf{M}$, the main idea of k-means is, in fact, to iteratively evaluate a clusterisation C via (2.22) and then use it to evaluate a novel set of centroids $\mathbf{M}$ via (2.26). The procedure is then iterated until convergence is reached, *i.e.* no sample change in cluster assignment. A pseudo-code of the model is provided in Algorithm 1

It is easy to see that, under the general position assumption (see Footnote 8), the algorithm converges; in fact, each step of Algorithm 1 lowers the total cost of the configuration, cf. (2.27). However, it is proved that, in the worst case, the algorithm takes $2^{\Omega(\sqrt{N})}$ iterations to reach convergency, see [24]. Worst-case scenarios are nevertheless unfair and resemble critical sample distributions. It is in fact also proven

that, if samples are randomly perturbed, then the algorithm converges in linear time both in N and k, see [23].

An important topic in k-means is also related to the choice of initial clusters; in fact, despite the model is deterministic, it can provide different outcomes depending on different initial conditions. For this reason, it is useful to execute the model multiple times while varying initial conditions and to compare the results in terms of the cost of the configuration provided as output. The proposed solution in Algorithm 2.27 is to initialise a random clustering and we evaluated centroids on it. However, randomly generated centroids serve the purpose as well.

A second important thing to notice is that k must be set *a priori*. In the next section, we will discuss a clustering method that overcomes this problem by building a family of clusters at the varying of $k \in \{1, \dots N\}$.

2.4.2 Hierarchical clustering analysis

We now proceed in the description of *hierarchical clustering analysis* (HCA), a powerful clustering technique that, unlike k-means, does not assume the knowledge of k *a priori*, but rather builds progressively a family $\mathcal{F}$ of hard clusterings. In particular, the family $\mathcal{F}$ is formed by a succession of hard clusterings of the dataset that are one the refinement of the other.

Depending on the construction strategy of $\mathcal{F}$, HCA is divided into two variants, namely

Agglomerative HCA When $\mathcal{F}$ is built with a *bottom-up* strategy, by progressively merging clusters up to a unique one, the HCA is referred to as *agglomerative* (A-HCA).

Divisive HCA When $\mathcal{F}$ is built with a *top-down* strategy, hence starting from a single omni-comprehensive clustering which gets progressively divided, the HCA is referred to as *divisive* (D-HCA).

In general, merges (or splits) are determined in a greedy manner with the help of some form of cluster metric (see below).

In what follows we analyse A-HCA (and we drop the 'A-' for readability) since it is the one that we later use in Section 4.6.3 and, in general, the discussion is similar for the two approaches. However, we refer the reader to [45, Chapter 6] for the detailed description of D-HCA approach referred, there, as *divisive analysis clustering* (DIANA).

More in details, let $D = \{\mathbf{x}_i\}_{i=1}^N$ be an (un-)labeled dataset. (A-)HCA produces the family $\mathcal{F} = (C_1, \dots, C_N)$ such that $C_i, 1 < i \leq N$ represents a one-step refinement of C_{i-1}, meaning that each C_i is a sub-partition of $C_j, \forall i < j \leq N$ and, in particular, C_i differs from C_{i-1} by a sole couple of clusters that get merged in i-th step. Formally, we have that $C_1 = \{\{\mathbf{x}\}, \forall \mathbf{x} \in D\}$ is the clustering where all the samples belong to a different cluster. Then, if we let $A, B \in C_{i-1}$ be the closest clusters *w.r.t.* some metric in C_{i-1}, then C_i is defined as

$$C_i = C_{i-1} \backslash \{A, B\} \cup (A \cup B) \qquad \forall i \in \{2, \dots, N\}. \tag{2.28}$$

Do note, in fact, that the described procedure lasts exactly $N - 1$ steps and, unlike k-means which depends on centroid initialisation, it is deterministic (unless two couples of clusters share the same distance).

However, a piece of information is still missing: a proper definition of a *metric* within the cluster space, or *linkage* in the jargon, *i.e.* a function d_{C}

$$\begin{aligned} d_{\mathrm{C}} : [D]^* \times [D]^* &\to \quad \mathbb{R} \\ (A, B) \quad &\mapsto d_{\mathrm{C}}(A, B) \end{aligned}, \tag{2.29}$$

where $[D]^*$ denotes a set of arbitrary size over D. Various methods can be found in the literature to define d_{C}, most of which resemble some metric $d : D \times D \to \mathbb{R}$ over the space of samples. In the following, we assume such metric as given (simple metrics include *e.g.* Euclidean, Manhattan, and, in general, L^p norm), and we define a few useful derived clustering metrics for clusters $A, B \in [D]^*$ (see also [36]).

SLINK The *single linkage* is defined as the shortest distance dividing two clusters

$$\textsc{slink}(A, B) = \min\{d(\mathbf{a}, \mathbf{b}) \mid \mathbf{a} \in A, \mathbf{b} \in B\} \,. \tag{2.30}$$

CLINK The *complete linkage* is defined as the largest distance dividing two clusters

$$\textsc{clink}(A, B) = \max\{d(\mathbf{a}, \mathbf{b}) \mid \mathbf{a} \in A, \mathbf{b} \in B\} \,. \tag{2.31}$$

UPGMC The *unweighted pair group with mean centroids* is defined as the distance dividing the (mean) centroids $\boldsymbol{\mu}_A$, $\boldsymbol{\mu}_B$ of the clusters, *i.e.*

$$\textsc{upgmc}(A, B) = d(\boldsymbol{\mu}_A, \boldsymbol{\mu}_B) \,. \tag{2.32}$$

Here centroids are evaluated just like in classical k-means (see (2.22)), however, different methodologies for centroids/medoids evaluation can be used leading to different *unweighted pair group centroids* (UPGC) definitions.

In particular, it is interesting that UPGMC naturally provides also an explicit representative of the clusters (being the mean of the internal point). This behaviour, however, comes at a high cost that might penalise the method. In fact, if we consider the series $\mathbf{d}$ defined by the progressive distances of the joint centroids (*i.e.* where d_i represent the distance of the joined clusters at time i), then $\mathbf{d}$ is positive monotone by definition in SLINK and CLINK, while it can potentially decrease in UPGMC.

HCA dendrogram and other representations

The last remark on $\mathbf{d}$, is strictly related to a different issue in HCA: which clustering should be considered amongst $\mathcal{F}$? In fact, by looking at the sole family, no direct hints are provided on which clustering better suits the data *w.r.t.* the others. Like

many issues in ML, there is no definitive answer to this question; however, a few representations of the family $\mathcal{F}$ can be beneficial. We here report three of them:

Distance plot First and foremost, we already introduced the concept of the join-distance vector $\mathbf{d}$. A simple yet useful method to determine which clustering is more suitable is to choose a given threshold $\hat{d}$ and to consider the last HCA iteration joining two clusters with corresponding distance bounded by $\hat{d}$. In other words, a clustering C_i can be chosen as $i = \min\{i \in \{1, \dots, N \mid d_i > \hat{d}\} - 1$. A suitable threshold for $\hat{d}$ might naturally emerge by the underlying problem or it can be determined via the *elbow method* applied to the plot of $\mathbf{d}$ (*i.e.* a sudden change in the slope in $\mathbf{d}$, see also [58]).

Size plot A different approach can be to evaluate the number $\mathbf{c}^\xi$ of *ξ-significant clusters*, *i.e.* clusters of size greater than $\xi \in \mathbb{N}$. In formulas, we have $\mathbf{c}^\xi \in \mathbb{N}^N$, where ${c_i}^\xi = |\{C \mid |C| > \xi, C \in C_i\}|$. Usually, a *plateau* can be found in the plot of $\mathbf{c}^\xi$, at the varying of ξ (see later Section 4.6.3). The *ratio* beneath this approach consists in the fact that it should be possible to locate a number of significant clusters which are *stable* during the process, *i.e.* that absorb other smaller clusters but are not joined with other stable clusters. If this is the case, then a suitable stopping point for the HCA procedure corresponds to when these stable clusters join together.

Dendrogram Finally, it is useful to note that, since clusterings are progressive refinement of the same partition, then it is possible to represent the procedure as a rooted graph $G = (V, E)$ often referred to as *dendrogram*. In particular, let $V = \{C \in C, \forall C \in \mathcal{F}\}$ be the set of distinct clusters amongst the different clusterings. Then, an edge exists between two nodes (*i.e.* clusters) if one is the result of the join operation *w.r.t.* the other. It is rather clear that G is then a binary tree rooted in the single cluster contained in C_N and where the leaves are the clusters formed by the single samples $\{\mathbf{x}\}$. Pictorial representation of dendrograms can help assess both $\mathbf{d}$ and $\mathbf{c}$ as the vertices can be organised accordingly in height.

An example of the above-described representations is shown in Figure 2.9.

On the complexity of HCA

A pseudo-code representation of the HCA procedure is provided in Algorithm 2. It is rather clear that the computational time of HCA is, in the general (unoptimised) case, unsatisfactory bounded by $O(N^3 \cdot \bar{d})$, where $\bar{d}$ represents the cost of the evaluation of the metric function d_c. Optimised versions of the algorithm provide a tradeoff by storing clustering distances in a matrix and selectively updating (and evaluating) only distances that changed due to cluster joins. However, in the case of SLINK and CLINK, optimal algorithms of complexity $O(N^2)$ are known (see [32, 57]).

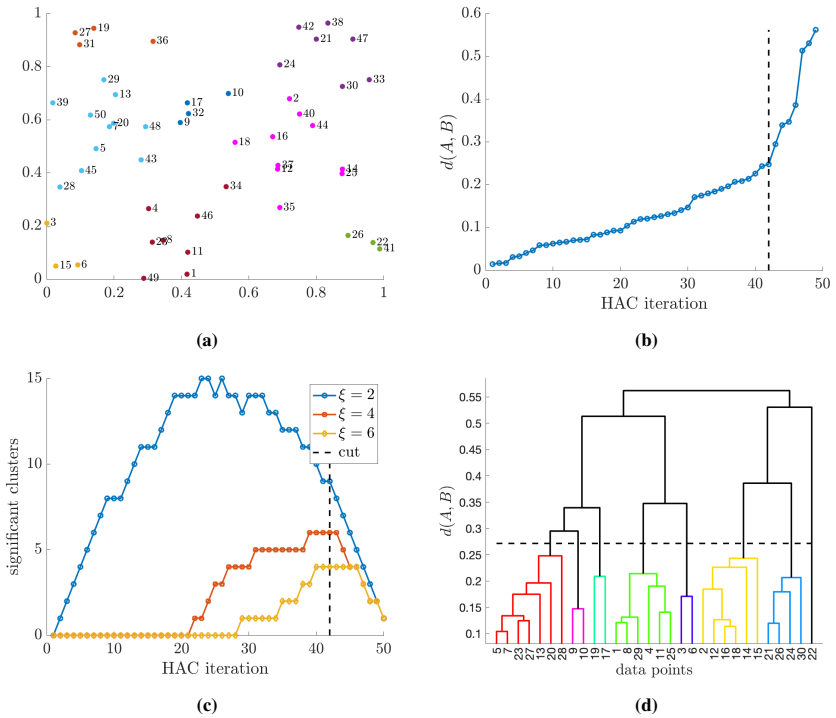


Fig. 2.9: An example of hierarchical clustering analysis of $N = 50$ randomly generated points in $\mathbb{R}^2$ with UPGMC. **(a)** $N = 50$ points along with their clusterisation when HCA is stopped at iteration 42 (*i.e.* 8 clusters). **(b)** corresponding cluster-join distance **d**; as it can be seen, an elbow can be located at iteration 42. **(c)** the number of significant clusters for different values of ξ; again, iteration 42 identifies the end of a plateau. **(d)** the dendrogram associated with the procedure, where y-coordinate reports the distance from **d** at which the join was made; the cut at iteration 42 is highlighted with a dashed line.

References

[21] C. C. Aggarwal. *Linear algebra and optimization for machine learning*. 1st ed. Vol. 156. Springer Cham, 2020. ISBN: 978-3-030-40344-7. DOI: 10.1007/978-3-030-40344-7 (cit. on p. 34).

[22] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. "Automatic subspace clustering of high dimensional data for data mining applications". In: *ACM SIGMOD Record* 27.2 (June 1998), pp. 94–105. DOI: 10.1145/276305.276314 (cit. on p. 53).

[23] D. Arthur, B. Manthey, and H. Röglin. "k-Means Has Polynomial Smoothed Complexity". In: *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, Oct. 2009. DOI: 10.1109/focs.2009.14 (cit. on p. 56).

Algorithm 2: `hierarchicalClustering`$(D, d_{\mathrm{C}}) \to \mathcal{F}$

Input: A un-labelled dataset D and a cluster metric definition d_{C}
Output: A clusterings family $\mathcal{F}$ given as a map $\mathcal{F} : i \to C_i$

```
// Build initial clustering made of single samples only
C ← ∅;
foreach (i ∈ {1, ..., N}) do
    C ← C ∪ {x_i};
F[1] ← C;
foreach (i ∈ {2, ..., N}) do
    // Locate the closes clusters according d_C
    d* ← +∞;
    foreach (U ∈ C) do
        foreach (V ∈ C | U ≠ V) do
            if (d_C(U, V) < d_BEST)
                d* ← d_C(U, V);
                U* ← U;
                V* ← V;
    // Remove old clusters from C
    C ← C\{U*, V*};
    // Add merged cluster
    C ← C ∪ (U* ∪ V*);
    // Update the family F
    F[i] ← C;
return F;
```


[24] D. Arthur and S. Vassilvitskii. "How slow is the k-means method?" In: *Proceedings of the twenty-second annual symposium on Computational geometry*. ACM, June 2006. DOI: 10.1145/1137856.1137880 (cit. on p. 55).

[25] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006 (cit. on pp. 39, 41).

[26] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN: 978-0521833783 (cit. on p. 35).

[27] J. Brownlee, S. Cristina, and M. Saeed. *Calculus for Machine Learning*. 1st ed. Machine Learning Mastery, 2022 (cit. on p. 35).

[28] O. Calin. *Deep learning architectures: a mathematical approach*. Springer, 2020 (cit. on pp. 33, 35, 40, 41).

[29] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014. DOI: 10.3115/v1/d14-1179 (cit. on p. 48).

[30] B. J. Copeland. *The essential Turing*. Oxford: Clarendon Press, 2004. ISBN: 0-19-825079-7 (cit. on p. 33).

[31] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314. DOI: 10.1007/BF02551274 (cit. on p. 48).

[32] D. Defays. “An efficient algorithm for a complete link method”. In: *The Computer Journal* 20.4 (Apr. 1977), pp. 364–366. DOI: 10.1093/comjnl/20.4.364 (cit. on p. 58).

[33] M. P. Deisenroth, A. A. Faisal, and C. S. Ong. *Mathematics for machine learning*. Cambridge University Press, 2020. URL: https://mml-book.github.io (cit. on pp. 33–36).

[34] L. Deng. “The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142. DOI: 10.1109/MSP.2012.2211477 (cit. on p. 36).

[35] S. V. Dongen. “Graph Clustering Via a Discrete Uncoupling Process”. In: *SIAM Journal on Matrix Analysis and Applications* 30.1 (Jan. 2008), pp. 121–141. DOI: 10.1137/040608635 (cit. on p. 53).

[36] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. 2nd ed. John Wiley & Sons, Nov. 2012. ISBN: 111858600X (cit. on p. 57).

[37] J. C. Dunn. “A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters”. In: *Journal of Cybernetics* 3.3 (Jan. 1973), pp. 32–57. DOI: 10.1080/01969727308546046 (cit. on p. 53).

[38] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM. 1996, pp. 226–231 (cit. on p. 53).

[39] D. H. Fisher. “Knowledge acquisition via incremental conceptual clustering”. In: *Machine Learning* 2.2 (Sept. 1987), pp. 139–172. DOI: 10.1007/bf00114265 (cit. on p. 53).

[40] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org (cit. on pp. 39, 135).

[41] P. Grohs and G. Kutyniok. *Mathematical Aspects of Deep Learning*. Cambridge University Press, 2022 (cit. on p. 36).

[42] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer Series in Statistics. Springer New York, NY, 2009. ISBN: 978-0-387-21606-5. DOI: 10.1007/978-0-387-21606-5 (cit. on pp. 35, 36).

[43] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 49).

[44] J. J. Hopfield. “Neural networks and physical systems with emergent collective computational abilities”. In: *Proceedings of the National Academy of Sciences* 79.8 (1982), pp. 2554–2558. DOI: 10.1073/pnas.79.8.2554 (cit. on p. 45).

[45] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009. ISBN: 0470317485 (cit. on p. 56).

[46] L. Kaufman and P. J. Rousseeuw. "Partitioning Around Medoids (Program PAM)". In: *Finding Groups in Data*. John Wiley & Sons, Inc., 1990, pp. 68–125. DOI: 10.1002/9780470316801.ch2 (cit. on p. 53).

[47] T. Kohonen. "Self-organized formation of topologically correct feature maps". In: *Biological Cybernetics* 43.1 (1982), pp. 59–69. DOI: 10.1007/bf00337288 (cit. on p. 53).

[48] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541 (cit. on p. 45).

[49] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao. "Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00572 (cit. on p. 48).

[50] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. "Federated Learning: Challenges, Methods, and Future Directions". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60. DOI: 10.1109/MSP.2020.2975749 (cit. on p. 35).

[51] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489 (cit. on p. 52).

[52] D. J. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003. ISBN: 0-521-64298-1 (cit. on pp. 33, 35, 36).

[53] A. Ng, M. Jordan, and Y. Weiss. "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems* 14 (2001) (cit. on p. 53).

[54] R. L. Rivest et al. "Cryptography and machine learning". In: *ASIACRYPT*. 1991, pp. 427–439 (cit. on p. 35).

[55] F. Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain". In: *Psychological review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519 (cit. on p. 42).

[56] A. Shamir, I. Safran, E. Ronen, and O. Dunkelman. "A simple explanation for the existence of adversarial examples with small hamming distance". In: *arXiv preprint* (2019). ARXIV:1901.10861 (cit. on p. 35).

[57] R. Sibson. "SLINK: An optimally efficient algorithm for the single-link cluster method". In: *The Computer Journal* 16.1 (Jan. 1973), pp. 30–34. DOI: 10.1093/comjnl/16.1.30 (cit. on p. 58).

[58] R. L. Thorndike. "Who belongs in the family?" In: *Psychometrika* 18.4 (Dec. 1953), pp. 267–276. DOI: 10.1007/bf02289263 (cit. on p. 58).

[59] A. M. Turing. "Computing Machinery and Intelligence". In: *Mind* 59.236 (1950), pp. 433–460 (cit. on p. 33).

[60] G. Voronoi. “Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives parfaites.” In: *Journal für die reine und angewandte Mathematik (Crelles Journal)* 1908.133 (Jan. 1908), pp. 97–102. DOI: 10.1515/crll.1908.133.97 (cit. on p. 54).

Part II
Coloured-graphs analysis

Overview

Chapter 3
Graph contraction on attribute-based colouring

This chapter presents a novel performance-oriented algorithm for contracting graphs provided with a vertex-colouring function. The algorithm was developed in collaboration with Flavio Lombardi, researcher at "Istituto per le Applicazioni del Calcolo" *of the* "Consiglio Nazionale delle Ricerche" *(IAC-CNR). Here I harmonise the contributions from a conference paper [278], a journal paper [274], and a paper in preparation [287], hence providing a mathematical formalisation that, to the best of my knowledge, was still missing in the literature.*

Abstract Networks play a ubiquitous role in computer science and real-world applications. Using distinctive node features to group such data into clusters, hence represented by a single representative node per cluster, proves to be a valuable approach. These contracted graphs can reveal previously hidden characteristics of the original networks.
In many real-world cases, clusters can be identified by a set of connected vertices that share the result of some categorical function, *i.e.* a mapping of the vertices into some categorical representation that takes values in a finite set C. As an example, we can identify contiguous terrains with the same discrete property on a geographical map (leveraging Space Syntax) or contiguous nodes having the same discrete property in a social network. Contracting a graph allows a more scalable analysis of the interactions and structure of the network nodes.
This chapter delves into the problem of contracting possibly large coloured networks into smaller and more easily manageable representatives. It provides a rigorous mathematical definition of the problem, which, to the best of our knowledge, was missing in the existing literature. Specifically, we explore the variadic nature of the contraction operation and use it to introduce a weaker version of colour contraction. We propose a novel effective algorithm to perform colour contraction efficiently by leveraging this weaker definition. Performance plots are provided for a range of graphs and results are thoroughly analysed and discussed, aiming to provide practical use cases and application scenarios for the approach.

Keywords: Coloured networks · Graph contraction · Greedy algorithm · Graph analysis

2020 MSC: 90C35 · 94A16 · 68T09 · 05C90 · 68W10

3.1 Introduction

Networks are pervasive in computer science and in real-world applications [80]. Collecting, navigating, and extracting insights from such data and from the underlying graph structure is often challenging [62]. A widespread technique is to organise data in clusters by means of some characteristics and extract a representative element from each cluster [68]. The graph made of these representative objects is usually much smaller than the original one while it preserves many useful characteristics.

A common way to describe clusters is to provide a categorical classification of the vertices by means of some colouring function, *i.e.* a mapping of the vertices into some fixed representation. Colouring functions can be generated as the outcome of clustering techniques (see Section 1.2.7). Oftentimes, however, this kind of information is natural with the graph – like the language spoken by users in a Social Network – or can be injected as expert-knowledge – as the additional information a geologist can provide on a specific terrain map. Indeed, also continuous variables, like *e.g.* air pressure in weather forecasting maps, can be sketched as discrete if sampled in fixed ranges.

Networks can be quite large in nature, thus having an efficient way to contract them provides a useful approach to ease analysis and detection of issues. In this sense, stakeholders can be identified as the analysts that need to extract information, *e.g.* for urban planning, where city maps are contracted on neighbourhoods sharing the same thematic area. Other stakeholders are those ones analysing networks representing Unspent Transaction Output or account-based cryptocurrencies that might search, *e.g.*, for accumulation nodes or highly connected small clusters of vertices.

In the above scenarios, the vertices sharing the same categorical information can be *merged* via (graph) contraction, *i.e.* a novel vertex is generated in place of the previous ones, preserving the adjacency of the substituted vertices (see Section 1.5). Application of contraction generates novel graphs that share a number of properties with the original graph (*e.g.* connectivity), but whose size is usually much smaller (in both vertices and edges).

Graph contraction problems are an interesting set of problems on their own being a class of NP-Hard problems, *e.g.* determining whether one can obtain a tree by contracting at most k edges from a given connected graph to obtain a tree (*i.e.* Tree Contraction) is a possible example [61].

3.1.1 Related Work

Graph contraction is useful in many graph-related problems since it tackles the problem of reducing input data into a more manageable size. It is the case, for example, of image 3D curve skeleton extraction, as presented in [74]. Solutions to many well-known graph problems have been formulated by means of procedural reduction of the original graph (cf. Section 1.5); it is the case, *e.g.*, of finding the

shortest path [66, 11], of computing chromatic polynomials [86], evaluating DP colouring [79] or evaluating the number of spanning trees [69] of a simple graph.

Parallel computing, where data must be mapped to processing units via some not a priori known logic, is an interesting use case for graph contraction as well. Ponnusamy *et al.* [82] introduced an iterative parallel graph contraction algorithm to pairwise contract vertices, hence reducing the problem size before mapping data. Meyerhenke *et al.* apply a similar idea in [76], extending it in a multi-levelled way: namely, the graph is iteratively reduced while applying label propagation to obtain graphs of manageable size. An interesting review on these topics can be found in [85].

In both coloured and colourless versions, contraction provides many insights about graph connectivity-related features (see *e.g.* [84] for a fast connectivity algorithm based on graph contraction). Studying graphs and colour clusters connectivity is useful in a wide variety of applications [63], including when the graphs underlie beneath more complex processes, *e.g.* the connectivity of the basis of Markov process reveals important information about its Ergodicity (see *e.g.* later Chapter 6 for an Ergodic two-coloured graph-based Markov process).

Despite its usefulness in a wide variety of applications, to the best of our knowledge, there is not a rich research effort on colour-based graph contraction. Nevertheless, some research work can be found with applications in distinct areas.

D'Autilia and Spada [65] used colour-based graph contraction to retrieve the relationship between pedestrian, vehicular and hybrid areas in city maps to compare different mobility plans; in this context, the authors enforced colours on street junction-based graphs leveraging on Space Syntax approach [71] where a city is analysed by means of its different areas. Later, in Chapter 4, we described a procedural classification method of cascaded localisers derived by colour contraction over museums graph representations where colours are injected by expert knowledge as architectural and conceptual constraints. Crypto-currency offers some interesting use cases as well – transactions and users graphs, whose analysis can be complex (see *e.g.* [64, 73]), can benefit from a feature-preserving size reduction of the original network. Finally, other applications include complex networks such as Social Networks and web graphs, with some novel results shown and discussed below.

Nevertheless, to the best of our knowledge, an efficient parallel algorithm for coloured graph contraction is not available at present. Some early work by Miller and Reif [78] and by Philips [81], is limited to –colourless– graphs. More recent work on label propagation by Meyerhenke *et al.* [77] aims at building colour clusters instead of contracting them. This motivates our work, whose final goal is to provide a contraction algorithm effectively capable of leveraging parallel computing.

3.1.2 Chapter contribution

In this chapter, we tackle the problem of colour contraction (or γ-contraction) and we discuss it in detail providing, to the best of our knowledge, the first mathematical formalisation of the problem. We introduce a consistent notation with regular

graph connectivity and contraction and we propose a weaker formalisation of the γ-contraction under the name of β-contraction. Here, the idea is to exploit local properties of the vertices instead of applying global visits that are known to be challenging to efficiently parallelise (see [5]).

We leverage β-contraction constructive definition (see Theorem 3.1) to describe a general approach to γ-contraction problem and we refine it by introducing a novel algorithm for graph colour contraction that is fast and reliable. The algorithm is applied in various scenarios and results are shown and discussed. Although the algorithm is presented in its serial version, it is designed to be naturally parallelisable[1]. Care has been taken to incorporate important concepts of parallel computing system design (we refer the interested reader to [67] for a good review on parallel graph algorithms). In particular, the underlying data structure is optimised, yielding a computational cost that, in the average case, is proportional to the number of edges of the original graph (*i.e.* like in a regular visit) while avoiding global assumptions on the vertices (that usually force parallel versions to use barriers).

3.1.3 Chapter organisation

The rest of this chapter is organised as follows. Section 3.2 provides the mathematical formalisation of the colour contraction problem, directly extending the contents from Chapter 1. Section 3.3 introduces our novel algorithm both providing a mathematical formalisation of it (Section 3.3.1) and describing it algorithmically (Section 3.3.3); underlying data structures (Section 3.3.2) and computational complexities (Section 3.3.4) are both discussed in details. Section 3.4 shows and discusses results obtained applying the proposed algorithm to benchmark Erdös-Rényi graphs (Section 3.4.1) and to a real-world graph of Facebook pages (Section 3.4.2). Finally, Section 3.5 closes the chapter by summarising contributions and discussing future developments.

3.2 Coloured-based graph contraction

In this section, we formalise the problem of contracting a (vertex-)coloured graph. Hence, let us consider a graph $G = (V, E, \gamma)$ equipped with a vertex-colour function $\gamma = \gamma_V : V \to C, C \subsetneq N, |C| = c < +\infty$, as described in Section 1.2.7.

We first and foremost need to extend the definition of graph contraction (see Section 1.5) to a coloured graph:

Definition 3.1 (Colour-preserving contraction)
Let $G = (V, E, \gamma)$ be a vertex-coloured graph and let $u, v \in V$ be two adjacent vertices

[1] A parallel version is already available at the time of writing but statistics, comparison, and profiling for performances are incomplete yet. In order to provide a self-contained work, such results, along with the detailed implementation of the parallel version, are not discussed here.

sharing the same colour, *i.e.* $u \sim v$ and $\gamma(u) = \gamma(v)$. We define the *colour-preserving contraction* – and we denote it as $G/_\gamma\{u, v\}$ – as the graph $G' = G/\{u, v\}$ equipped with the colouring function

$$\gamma' : \begin{matrix} V' \to & C \\ z \mapsto & \gamma'(z) \end{matrix}, \qquad \text{with} \qquad \gamma'(z) = \begin{cases} \gamma(z) & \text{if } z \neq w \\ \gamma(u) & \text{if } z = w \end{cases} \tag{3.1}$$

where w is the novel vertex in G' that blends u and v.

We define the corresponding procedure as $\mathfrak{m}^\gamma_G(u, v)$ analogously to $\mathfrak{m}_G(u, v)$; $\mathfrak{m}^\gamma_G(u, v)$ outputs the blended vertex $w \notin V$, the edge set E_w joining w with $N_G(\{u, v\})$, and the colour $x_w \in C$ such that $\gamma'(w) = x_w = \gamma(u)$.

It is straightforward to consider the following

Proposition 3.1 (Properties of colour-preserving contraction)
Colour-preserving contraction $\mathfrak{m}^\gamma_G$ is commutative and associative.

Proof For what concern commutative property, let us consider $w, E_w, x_w = \mathfrak{m}^\gamma_G(u, v)$ and $w', E_{w'}, x_{w'} = \mathfrak{m}^\gamma_G(v, u)$. In particular, we already know by the definition of $\mathfrak{m}_G$ that $w = w'$ and $E_w = E_{w'}$. From (3.1), it follows $x_w = \gamma(u)$ and $x_{w'} = \gamma(v)$. Since colour-preserving contraction is solely defined on u, v such that $\gamma(u) = \gamma(v)$ we can conclude that $x_w = x_{w'}$.

Being $\mathfrak{m}_G$ associative, associative property for $\mathfrak{m}^\gamma_G$ follows analogously.

In general, we refer to $\mathfrak{m}^\gamma_G$ simply as $\mathfrak{m}$ when it is clear from the context that we are considering a coloured graph.

Definition 3.2 (γ-contraction)
Given a vertex-coloured graph $G = (V, E, \gamma)$, we define the *γ-contraction* of G, more formally the *colour contraction* of G on the colour set C induced by the colouring γ, as the procedure that applies the (commutative, associative) colour-preserving graph contraction to all the couples of adjacent vertices that shares the same colour, *i.e.* that evaluates progressively $G/_\gamma\{u, v\}, \forall u, v \in V$ such that $u \sim v$ and $\gamma(u) = \gamma(v)$. We denote such operation as G/γ.

Before proceeding, it is worthwhile for ease of notation to provide the following set of definitions on coloured graphs

Definition 3.3 (Colour neighbourhood)
Let $G = (V, E, \gamma)$ be a vertex-coloured graph. We define the *colour neighbourhood* of $v \in V$ – and we write $N_\gamma(v)$ – as the set of adjacent vertices of v that share the same colour with it

$$N_\gamma(v) = \{u \in N_G(v) \mid \gamma(u) = \gamma(v)\} . \tag{3.2}$$

Mimicking the notation for the regular neighbourhood (cf. (1.7)), we also define the colour neighbourhood of a set of vertices $U \subseteq V$ as

$$N_\gamma(U) = \bigcup_{u \in U} N_\gamma(u) \backslash U . \tag{3.3}$$

We refer to the size of the colour neighbourhood as the vertex/vertex set *colour degree*, *i.e.*

$$\partial_\gamma(u) = |N_\gamma(v)| \,, \qquad \partial_\gamma(U) = |N_\gamma(U)| \,. \tag{3.4}$$

Observation 3.1 (On colour neighbourhoods)
In the following, we assume that $\gamma(u) = \gamma(v), \forall u, v \in U$ when referring to $N_\gamma(U)$. In fact, dropping such a condition would result in having multiple colours amongst the vertices in $N_\gamma(U)$.

Definition 3.4 (Colour cluster and colour component)
Let $G = (V, E, \gamma)$ be a vertex-coloured graph. We define a *colour cluster* as a set of vertices $S \subseteq V$ such that $\forall u, v \in S$ we have $\gamma(u) = \gamma(v)$ and there exists $\Gamma_{u,v} = (w_0, \dots, w_k)$ entirely contained in S, *i.e.* $w_i \in S$, for $0 \leq i \leq k$. We denote with $\gamma(S)$ the colour of the vertices contained in S.

If a colour cluster S is *maximal* (*i.e.* $\forall S' \subset V$ colour cluster, $S \subseteq S'$ then $S = S'$) we say that S is a *colour component*.

Observation 3.2 (On the connectivity of colour clusters and colour components)
Requiring the existence between u and v of $\Gamma_{u,v}$ entirely contained in S is equivalent to requiring that $\langle S \rangle_G$ is connected (cf. Section 1.2.5). We here explicitly refer to the path Γ joining each couple of vertices in S as it is later useful in both the proof of Proposition 3.4 and in the explanation of the algorithm introduced in this chapter (cf. Section 3.3).

With the concept of colour-cluster, it is a natural consequence of Proposition 3.1 to claim the following

Proposition 3.2 (Colour contraction variadic form)
The colour-preserving contraction procedure $\mathfrak{m}_G^\gamma$ admits a variadic form on colour clusters. In other words, given $U \subseteq V$ a colour cluster, then $G/_\gamma U$ is well defined and equips the resulting graph G/U with the colouring function defined in (3.1).

Proof The proposition directly follows from the fact that colour-preserving contraction is commutative and associative, analogously to regular graph contraction (cf. Section 1.5). □

A useful remark on colour components, also highlighting the similarity with connected components from Section 1.2.5, is provided in the following

Observation 3.3 (Colour components are disjoint)
It directly follows from Definition 3.4 that, given two colour components S_1, S_2 of G, then if $v \in S_1$ and $v \in S_2$ we have $S_1 = S_2$. In other words, $S_1 \cap S_2$ is either empty or equal to S_1 (and S_2).

Definition 3.5 (Colour (sub-)partition)
Let $G = (V, E, \gamma)$ be a vertex-coloured graph. We define a *colour sub-partition* of G as a set of colour clusters $\mathcal{S}$ that forms a partition on the vertex set V. We further define a *colour partition* $\mathcal{S}^\star$ as a minimal (sized) colour sub-partition.

It is pretty straightforward to claim the following

Proposition 3.3 (The colour partition is unique)
Let $G = (V, E, \gamma)$ be a vertex-coloured graph. The colour partition $\mathcal{S}^\star$ of G – and we denote it as $C_\gamma(G)$ – is unique and it is made of all the colour components of G.

Proof The proof follows from Observation 3.3. Since each vertex is contained in a single colour component, then the set $\mathcal{S}^\star$ of all the colour components of G is unique. Furthermore, $\forall S_i, S_j \in \mathcal{S}^\star, S_i \neq S_j$, it holds $S_i \cap S_j = \emptyset$ and, since $S^\star$ collects all the colour components, then $\forall v \in V, \exists! S \in \mathcal{S}^\star$ such that $v \in S$, *i.e.* $\mathcal{S}^\star$ is a partition of V. Minimality directly follows from the fact that colour components are maximal. □

In order to build a colour partition, it is useful to characterise the colour components by means of the following

Proposition 3.4 (Colour component characterisation)
Let $G = (V, E, \gamma)$ be a vertex-coloured graph and let $S \subseteq V$ be a colour cluster of G. We have that S is a colour component $\iff N_\gamma(S) = \emptyset$.

Proof We prove one implication at a time, both *ad absurdum*.

($\Leftarrow$) Let $S \subseteq V$ be a colour cluster such that $N_\gamma(S) = \emptyset$. If *ad absurdum* $\exists S' \subseteq V$ such that S' is a colour cluster and $S \subsetneq S'$, then $\hat{S} = S' \backslash S \neq \emptyset$. In particular, let us consider $v \in \hat{S}$ and $u \in S$ then $\exists \Gamma_{u,v} = (w_0, \dots, w_\ell)$, with $w_i \in S', \forall 0 \leq i \leq \ell$ and $w_0 = u$, $w_\ell = v$. Let us denote with $\hat{i}$ the first i such that $w_i \notin S$, *i.e.*

$$\hat{i} = \min\{i \in \{0, \dots, \ell\} \mid w_i \in \hat{S}\} .$$

We have that $w_{\hat{i}-1} \in S$ (which is well-posed, in fact $\hat{i} > 0$ since $w_0 = u \in S$), hence $\{w_{\hat{i}-1}, w_{\hat{i}}\} \in E$ and $\gamma(w_{\hat{i}-1}) = \gamma(w_{\hat{i}})$ since $w_{\hat{i}-1}, w_{\hat{i}} \in S'$. It follows that $w_{\hat{i}} \in N_\gamma(w_{\hat{i}-1}) \subset N_\gamma(S) \neq \emptyset$ ($\Rightarrow\Leftarrow$).

($\Rightarrow$) Let $S \subset V$ be a colour component. If *ad absurdum* $\exists u \in N\gamma(S)$, let us prove that $S' = S \cup \{u\}$ is a colour cluster (such that $S \subsetneq S'$). By Definition 3.3 we have $\gamma(u) = \gamma(w), \forall w \in S$ and there exists $v \in S$ such that $u \sim v$. Let us call $e = \{u, v\}$. $\forall w \in S$, there exists a path $\Gamma_{w,v}$ which is entirely contained in S and which can be extended to v by e, hence forming a path $\Gamma_{w,u}$ entirely contained in S' . It follows that S' is a local colour cluster such that $S \subsetneq S'$ ($\Rightarrow\Leftarrow$). □

The proof of Proposition 3.4 highlights a constructive way to evaluate a colour component from one of its vertices $v \in V$, which is resumed in Algorithm 3. Hence, the colour partition of a graph can be determined by means of a simple yet effective procedure, as described in Algorithm 4.

We are now ready to refine Definition 3.2 under the variadic form of $\mathfrak{m}_G^\gamma$ (cf. Proposition 3.2) as follows

Definition 3.6 (γ-contraction (variadic form))
Let $G = (V, E, \gamma)$ be a vertex-coloured graph and let $\mathcal{S}^\star = C_\gamma(G)$. We define the *$\gamma$-contraction* of G – and we denote it as G/γ – as the graph obtained by applying $G/_\gamma S, \forall S \in \mathcal{S}^\star$.

Algorithm 3: `EvalColourComponent`$(v, G) \mapsto S$

Input: A vertex v of the vertex-coloured graph $G = (V, E, \gamma)$
Output: The colour component S where v belongs to

```
1 S ← ∅;
2 N ← {v};
3 repeat
4 |   S ← S ∪ N;
5 |   N ← N_γ(S);
6 until (N ≠ ∅);
7 return S;
```

Algorithm 4: `EvalColourPartition`$(G) \mapsto \mathcal{S}^\star$

Input: A vertex-coloured graph $G = (V, E, \gamma)$
Output: The colour partition $\mathcal{S}^\star = C_\gamma(G)$

```
1 S* ← ∅;
2 K ← V;
3 while (K ≠ ∅) do
4 |   Let v be a vertex from K;
5 |   S ← EvalColourComponent(v, G);
6 |   S* ← S* ∪ {S};
7 |   K ← K\S;
8 return S*;
```

Algorithm 5: `simpleGammaContraction`$(G) \mapsto G'$

Input: A vertex-coloured graph $G = (V, E, \gamma)$
Output: The corresponding γ-contraction $G' = G/\gamma$

```
1 S* ← EvalColourPartition(G);
2 foreach (S ∈ S*) do
3 |   G ← G/_γ S;
4 return G;
```

The latter definition further highlights that γ-contraction builds a novel graph $G' = (V', E', \gamma') = G/\gamma$ where each node $v_i \in V'$ is the contraction of a colour component $S_i \in C_\gamma(G)$ and it is equipped with the colour $\gamma(S_i)$. It follows that $|G'| = |C_\gamma(G)| = |C_{\gamma'}(G')|$ and, since all S_i are maximal, $\forall u', v' \in V'$ if $u' \sim v'$ then $\gamma'(u') \neq \gamma'(v')$. In other words, G' is a c-partite graph, where the partition is induced by γ' (and γ' is a proper colouring of G). Hence we can use Algorithm 4 to define a simple, intuitive way to perform γ-contraction, as resumed in Algorithm 5.

In particular, in the upcoming Section 3.3, we explain how the variadic nature of $\mathfrak{m}_G^\gamma$ is used to build our fast algorithm for G/γ evaluation.

3.3 The γ-contraction algorithm

In this section, we introduce our novel algorithm for evaluating γ-contraction of generic vertex-coloured graphs. We first provide an overview of our solution and then we delve into the technical details of the implementation that makes the algorithm efficient.

3.3.1 Algorithm overview

Our novel approach to γ-contraction mainly leverages the variadic nature of $\mathfrak{m}_G^\gamma$. The algorithm we introduce progressively determines colour sub-partitions $\mathcal{S}$ and applies $\mathfrak{m}(S_i)$ for all the $S_i \in \mathcal{S}$ concurrently. The reason why we went for this approach rather than evaluating colour partitions like in Algorithm 4 is that Algorithm 4 is hardly parallelisable since all of the operations introduced are strongly sequential. Conversely, in our algorithm, we efficiently determine a forest subgraph of G where each tree spans a (possibly maximal) colour cluster in $\mathcal{S}$. In particular, only a few iterations are typically needed to build G/γ ($\sim$ 4 iterations for $n \sim 5 \times 10^{+4}$ and $m > n \log n$, see later Section 3.4.1).

We mainly divide the procedure into two phases (to be iterated until the γ-contraction is completed): (i) evaluation of a colour sub-partition $\mathcal{S}$ and (ii) actual contraction of the colour clusters $S \in \mathcal{S}$. More formally, let $G = (V, E, \gamma)$ be the graph to be contracted (with $n = |G|$ and $m = \|G\|$), $\mathcal{S}_\beta$ be the colour sub-partition determined in the first phase, and $G' = (V', E', \gamma')$ the graph resulting from the second phase (with $n' = |G'|$ and $m' = \|G'\|$). Then, the two phases are defined as follows:

contraction mapping evaluation We build a contraction mapping $\beta : V \to V' \cong \mathcal{S}_\beta$ that identify, per each vertex $v \in V$, a novel vertex $v' \in V'$. In particular, we recall that there exists a bijection between the novel vertices $v' \in V'$ and the colour clusters $S \in \mathcal{S}_\beta$ through which G' is evaluated. In this phase, $\mathcal{S}_\beta$ is not known *a priori* and it is generated with an emerging procedure that only relies on (easily parallelisable) local properties of each vertex $v \in V$ *i.e.* its colour neighbourhood $N_\gamma(v)$. In particular, this is done in a four-step approach that: (i) identify a rooted forest where each maximal tree spans a colour cluster, (ii) builds a map π that links each node to the root of the tree it belongs to, (iii) builds a bijection α that assigns a novel vertex to each root of the forest, and (iv) builds the map β such that vertices within distinct trees are linked to the corresponding novel vertex.

contraction mapping application We evaluate $G' = G/\beta$, which is well posed since V' (the codomain of β) can be interpreted as a set of colours holding $V' \cong \{0, \dots, n' - 1\}$[2]. In other words, for each $S_i \in \mathcal{S}_\beta$ (which corre-

[2] This assumption is natural in computer science, where V' is represented as an array and a node can be identified by its index within the array.

sponds to $\beta^{-1}(v'_i), v'_i \in V'$) we evaluate *concurrently* the result of $G/_\gamma S_i$. Here, roughly speaking, concurrently means that we evaluate a "multi-contraction" $G/_\gamma\{S_1, \ldots, S_{|\mathcal{S}_\beta|}\} = G/_\gamma \mathcal{S}_\beta$ rather than evaluating $G/_\gamma S_1/_\gamma \ldots /_\gamma S_{|\mathcal{S}_\beta|}$. However, we are not mathematically rigorous on the introduction of $G/_\gamma \mathcal{S}_\beta$, since it much more represents a computer science optimisation rather than a real novel operation.

Being mainly a computational fact, we leave the precise description of the second phase to Section 3.3.3. We here solely recall that, being the colour-preserving contraction associative, if we assume S_i contracting in a novel vertex w_i then we can define G' as:

$$\begin{aligned} V' &= \left\{w_i, \forall S_i \in \mathcal{S}_\beta\right\} \\ E' &= \left\{\{w_i, w_j\} \in [V']^2 \mid N(S_i) \cap S_j \neq \emptyset\right\} \\ \gamma'(w_i) &= \gamma(S_i),\ \forall w_i \in V' \end{aligned} \tag{3.5}$$

or, analogously, we can build it from β as

$$\begin{aligned} V' &= \{\beta(v), \forall v \in V\} \\ E' &= \{\{\beta(u), \beta(v)\} \forall u, v \in V \mid u \sim_G v\} \\ \gamma'(w') &= \gamma(\beta^{-1}(w')),\ \forall w' \in V' \end{aligned} \tag{3.6}$$

where we are exploiting the fact that V' and E' are sets to get rid of duplicates.

For what concern the first phase, the underlying idea is to create a digraph D on V where each node $v \in V$ is joined to a representative node $\hat{v}$ within $N_\gamma(v)$. To provide a definition of *representative* which is suitable to work locally, we assume a total ordering on V. This is naturally done by considering the bijection between V and $\{0, \ldots, n-1\}$ as before and then defining the representative as the minimum of the colour neighbourhood. Hence we claim and prove the following

Theorem 3.1 (Creation and characterisation of D)
Let $G = (V, E, \gamma)$ be a coloured-vertex graph where vertices are enumerated in $\{0, \ldots, n-1\}$. The digraph with self loops $D = (V, B)$, where B is defined as

$$B = \{(v, \hat{v}), \forall v \in V\} \tag{3.7}$$

with

$$\hat{v} = \min\{u \in N_\gamma(v) \cup \{v\}\}\,, \qquad \forall v \in V\,. \tag{3.8}$$

is a forest. In particular, each connected component $T \in C(D)$ is a tree whose vertices are a colour cluster $S = V_T$ of G, hence $C(D)$ define a colour sub-partition $\mathcal{S}_\beta$ of G. The vertex $r_T = \min\{v \in V_T\}, \forall T \in C(D)$ is such that $\forall v \in V_T$ the single path Γ_{v,r_T} is directed and maximal (w.r.t. v), hence r_T is a natural root for T and a natural representative for $S \in \mathcal{S}_\beta$. In particular, the representative r_T also induces a natural way to enumerate the colour clusters $S \in \mathcal{S}_\beta$.

Proof Let us prove the Proposition point by point

1. Let us prove that D is a forest. We recall that, in order for D to be a forest, then given two distinct nodes $u, v \in V$ there exists at most a single path $\Gamma_{u,v}$ joining them.
   Without loss of generality, let $u > v$ and assume u, v and on the same connected component, hence $\exists \Gamma_{u,v} = (w_0, \dots, w_\ell)$ a path connecting them. We now prove that $\Gamma_{u,v}$ is unique.
   In particular, from (3.7), each node $w \in V$ has $\partial_D^+(w) = 1$, and we have $v < u, \forall u \in N^-(v)$ and $u \geq v$ for the single node $u \in N^+(v)$ (where equality holds only if $u = v$, *i.e.* v has a self loop). It follows that each path can be divided into at most two monotone paths, *i.e.* (since $u > v$) $\exists a \in \{1, \dots, \ell\}$ such that $w_0 > w_1 > \dots w_a$ and (unless $a = \ell$) $w_a < w_{a+1} < \dots w_\ell$. In fact, if more than a change in monotony exists, then there exists a node $w_{\hat{i}}$ such that $w_{\hat{i}-1} < w_{\hat{i}}$ and $w_{\hat{i}} > w_{\hat{i}+1}$, which is impossible since it would imply that $\partial^+(v) = 2$.
   In particular, the two monotone paths Γ_{u,w_a} and Γ_{v,w_a} are unique according to the same argument, hence $\Gamma_{u,v}$ is unique as well.
2. Let us prove that a connected component $T \in C(D)$ spans a colour cluster. Being D a forest, it is straightforward that T is a maximal tree in D. In particular, it follows from (3.8) that $\forall u, v \in V_T$, then $\gamma(u) = \gamma(v)$, hence $\langle T \rangle_G$ is a (connected, colour preserving) sub-graph of G whose vertices (*i.e.* V_T) form a colour cluster. Clearly, being $C(D)$ a partition, we have that T_i and T_j are disjoint $\forall T_i, T_j \in C(D)$, hence
   $$\mathcal{S}_\beta = \mathcal{S} = \{V_T, \forall T \in C(D)\} \tag{3.9}$$
   is a colour subpartition.
3. Let us prove that for any $T \in C(D)$, given $r_T = \min\{v \in V_T\}$, the unique path Γ_{v,r_T} is directed $\forall v \in V_T$ and it is maximal *w.r.t.* v. Holding $\partial_D^+(u) = 1, \forall u \in V_T$, it is easy to see that, $\forall v \in V_T$, the longest directed path $\Gamma_{v,*}$ starting from v is uniquely determined by (3.8) and, in particular, it is monotone. Therefore, $\Gamma_{v,*}$ ends in a node $r_v = \min\{z \in \Gamma_{v,*}\}$. It follows from (1) that $\forall u, v \in V_T, \exists w \in \Gamma_{u,v}$ such that the path can be split into two monotone directed paths $\Gamma_{u,w}$ and $\Gamma_{v,w}$, hence $w = \min\{z \in \Gamma_{u,v}\}$ and $w \in \Gamma_{u,*} \cap \Gamma_{v,*}$. In particular, $r_v = r_w$ and $r_u = r_w$, hence $r_u = r_v, \forall u, v \in V_T$, so $r_v = r_T, \forall v \in V_T$.
   The vertex r_T is then a natural choice for being the root of the tree T. Furthermore, if we let $R = \{r_T, \forall T \in C(D)\}$ be the set of the roots within the forest, then it is natural to consider the map $\tilde{\pi}$:
   $$\begin{array}{c} \tilde{\pi} : \mathcal{S} \rightarrow R \subseteq V \\ V_T \mapsto r_T \end{array} \tag{3.10}$$
   or, analogously, the map $\hat{\pi} : V \rightarrow R \subseteq V$ defined as
   $$\hat{\pi}(v) = \tilde{\pi}(S), \quad \forall v \in S, \quad \forall S \in \mathcal{S}\,. \tag{3.11}$$
4. We finally show the natural enumeration of colour clusters $S \subseteq \mathcal{S}$. Let $n' = |\mathcal{S}|$, then we can build the natural bijection $\hat{\alpha} : R \rightarrow \{0, \dots, n' - 1\}$ that preserve the ordering on R, *i.e.*

$$\hat{\alpha}(r_T) \leq \alpha(r_{T'}) \quad \Longleftrightarrow \quad r_T \leq r_{T'}, \qquad \forall r_T, r_{T'} \in R\,, \tag{3.12}$$

where equality holds if and only if $T = T'$. Hence, $\hat{\alpha}$ induces an enumeration $\tilde{\alpha}$ on $\mathcal{S}$ via $\tilde{\pi}$, *i.e.*

$$\begin{array}{rcl} \tilde{\alpha} : \mathcal{S} & \to & \{0, \dots, n'-1\} \\ S & \mapsto & \hat{\alpha}(\tilde{\pi}(S)) \end{array}\,. \tag{3.13}$$

The points (1)–(4) conclude the proof. □

Observation 3.4 In the proof of Theorem 3.1 we have defined constructively the colour sub-partition $\mathcal{S}_\beta$ in (3.9) and we have introduced two couples of analogous maps, whose form we recall in the following for ease of notation:

$$\begin{array}{ll} \hat{\pi} : V \to R, \text{ in (3.11)} & \hat{\alpha} : R \to \{0, \dots, n'-1\}, \text{ in (3.12)} \\ \tilde{\pi} : \mathcal{S} \to R, \text{ in (3.10)} & \tilde{\alpha} : \mathcal{S} \to \{0, \dots, n'-1\}, \text{ in (3.13)} \end{array}\,.$$

In particular, the formulation $\tilde{\alpha}$ and $\tilde{\pi}$ prove Theorem 3.1 while $\hat{\alpha}$ and $\hat{\pi}$ are used in the following to build β.

The four steps of "contraction mapping evaluation" phase follows naturally from Observation 3.4:

Creation of D We create a digraph with self loops $D = (V, B)$ with B defined as in (3.7). In particular, D is a rooted forest (with root set R) whose components in $C(D)$ define the colour sub-partition $\mathcal{S}_\beta$.

Creation of $\pi : V \to R$ The map π is already defined as $\hat{\pi}$ (3.11).

Creation of $\alpha : R \to V'$ Let us consider a set of novel vertices $V' = \{w_i, \forall i \in \{0, \dots, n'-1\}\}$ enumerated in $\{0, \dots, n'-1\}$, where $n' = |\mathcal{S}_\beta|$. Since clusters $S \in \mathcal{S}_\beta$ are enumerated in the same interval according $\hat{\alpha}$, as defined in (3.12), hence it is natural to build the bijection α between roots and novel vertices as follows:

$$\begin{array}{rcl} \alpha : R & \to & V' \\ r & \mapsto & w_{\tilde{\alpha}(v)} \end{array} \tag{3.14}$$

Creation of β The map $\beta : V \to V'$ directly follows as the composition $\alpha \circ \pi$, *i.e.*

$$\begin{array}{rcl} \beta : V & \to & V' \\ v & \mapsto & \alpha(\pi(v)) \end{array}\,. \tag{3.15}$$

An example of the described procedure can be found in Figure 3.1.

Observation 3.5 (D identifies colour clusters that might be not maximal)
It is important to notice that $S \in \mathcal{S}_\beta$, despite being a colour cluster, might or might be not a colour component depending on the chosen enumeration of vertices, *i.e.*. A simple counter-example is given by the path graph $G = P^4 = (\{v_0, \dots v_3\}, E, \gamma)$ where $E = \{\{v_0, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}\}$ and $\gamma(v) : V \to \{c\}$ which is represented in Figure 3.2. Here, we have $\hat{v}_0 = v_0$, $\hat{v}_1 = v_1$, $\hat{v}_2 = v_0$, and $\hat{v}_3 = v_1$, hence resulting in $D = (V, \{(v_2, v_0), (v_3, v_1)\})$ which identifies the two tree with vertex sets $S_0 = \{v_0, v_2\}$ and $S_1 = \{v_1, v_3\}$.

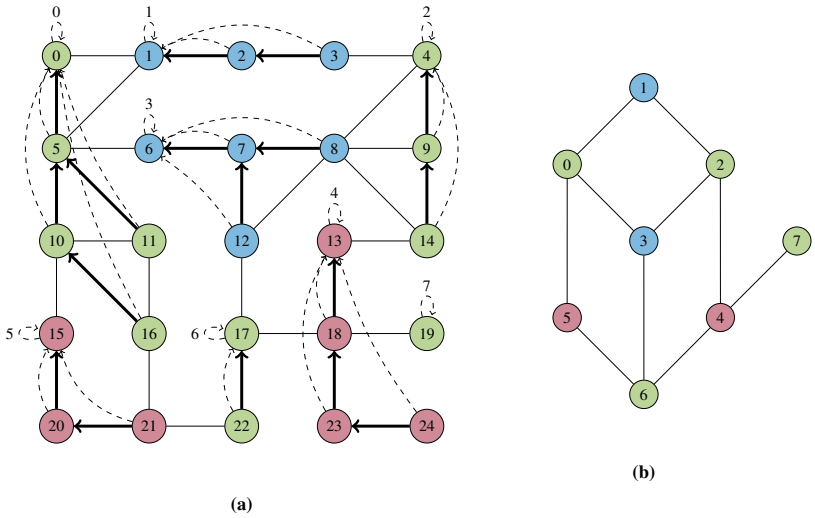


Fig. 3.1: Example of a 1-iteration β-contraction of the graph G to the graph $G' = G/\gamma$. **(a)** The graph $G = (V, E)$. The directed edges B of the digraph $D = (V, B)$ are highlighted as bold arrows over G (self loops are omitted). The map π is represented with dashed bend arrows which link each node within a colour cluster to the corresponding tree root *i.e.* nodes where π form a self loop. Numeric values of the map $\hat{\alpha}$ are reported on the π self loops. Map β is not reported for the sake of readability (see Figure 3.2). **(b)** The contracted graph $G' = (V', E') = G/\beta = G/\gamma$. Vertices in V' (*i.e.* the codomain of β) are enumerated according to values on self loops from π.

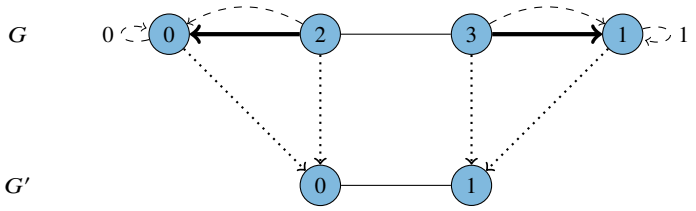


Fig. 3.2: A small example of graph $G = (V, E)$ for which β determines two distinct colour clusters instead of a single colour component. The corresponding contracted graph $G' = G/\beta = (V', E')$ is also reported. The directed edges B of the digraph $D = (V, B)$ are highlighted as bold arrows over G (self loops are omitted). The map π is represented with dashed bend arrows and numeric values of the map $\hat{\alpha}$ are reported on the π self loops. Map β is represented as dotted arrows connecting G with G'.

3.3.2 The data structures

Before discussing the actual implementation of the algorithm, it is necessary to devote a few words to the data structures involved in the contraction operation. So far, we have used a purely mathematical notation in our descriptions. However, given the implementation-oriented nature of this and the upcoming subsection, we will here use a computer-oriented notation. We will use `typewritten text` for code-related entities, with the exception of numeric variables that will be denoted with classical mathematical text weight *e.g.* $i, n, \ldots$. Unless differently specified, all the numeric variables, *i.e.* variables in $\mathbb{N}$, are assumed to be `idx_t`, a custom data type being `unsigned integer` by default. In fact, having a custom data type allows us to use the correct data type depending on the size of the graph to be stored, *e.g.* it could be `unsigned long` or `unsigned long long`. It makes exception the colour set C which typically requires much smaller values; for this reason, we refer as `col_t` to the colour type.

In particular, since our implementing language of choice is `C`, we will mainly use related notation. In particular, we talk about arrays and arrays of arrays instead of vectors and matrices and we use square brackets to identify elements, *i.e.* the i-th element of vector a is denoted as `a[`i`]`. Analogously, sets and mapping are built as arrays as well, where we have to ensure that elements of sets are disjoint. The array type is denoted by prepending a `*` character before a variable declaration, hence an array of arrays `a` of `idx_t` is denoted by `idx_t **a`. The arrays size needs to be stored manually and arrays of a given size n need to be created (or allocated, in the computer science jargon) with an allocation function `alloc`$(n,$ `d`$)$ which requires knowing the datatype `d` and is typically computationally non-negligible, *e.g.* $a \in \mathbb{N}^n$ is denoted by `a` $\leftarrow$ `alloc`$(n,$ `int`$)$. Allocated structures require explicit de-allocation via `free` when they are out of scope, *e.g.* `free(a)` must be raised when `a` is no longer needed. Along with regular data types, we use custom structures for graphs and contraction mappings (see below). The fields of a structure are accessed through the '.' operator, *e.g.* the field n of a graph `G` is denoted `G.`n. We avoid making direct use of pointers and relative operators in this description for ease of readability. For any further reference on `C`, we refer the reader to [70].

For the sake of simplicity, in the following we describe the minimal working implementation, meaning that graphs are undirected and present no weights but the vertex colour. In particular, an input graph might be a multi-graph and during the contraction procedure it eventually becomes a multi-graph anyway; however, the resulting graph will always be simple since contraction joins multiple edges connecting the same colour components and hence does our β-contraction. Vertices V of a graph $G = (V, E)$ are always assumed as $V = \{0, \ldots, |G| - 1\}$ and, since no information is held inside a vertex, vertex sets are not stored. Edges E are stored as adjacency lists (cf. Section 1.4), *i.e.* an array of arrays of `idx_t` of different sizes.

An extension of the algorithm over weighted or/and directed graphs is available at the moment of writing, however, its description is beyond the purpose of the present work since it requires many technicalities that are well out of purpose from the context of presenting the algorithm. In particular, the graph structure requires defining the

Algorithm 6: `Graph` data structure

```
1 struct Graph
2     idx_t n ; // order of the graph
3     idx_t m ; // size of the graph
4     col_t *C; // colour list, of length n
5     idx_t *Nv; // vertices degree, of length n
6     idx_t **E; // adjacency lists, for each vertex v of length Nv[v]
```

Algorithm 7: `ContractionMapping` data structure

```
1 struct ContractionMapping
2     idx_t n ; // order of the original graph
3     idx_t n' ; // order of the contracted graph, |S_β|
4     idx_t n* ; // effective size of cSize and revBecomes, either n or n'
5     idx_t *cSize; // size of the colour clusters S ∈ S, of length n*
6     idx_t *becomes; // β mapping, of length n
7     idx_t **revBecomes; // β^-1 mapping, for each vertex v of length cSize[v]
```

vertex set explicitly as an array of `vertex` and the edge set as an array of arrays of `edge`, both of which structures need to be defined properly. Furthermore, an explicit merge function is needed to combine `edge`s and `vertex`es during contraction. All of these requirements are bulky and, however, penalise the algorithm from the performances point of view and should be avoided if possible; this further motivates the discussion of the minimal working implementation (that is, however, the one used for benchmarks presented in Section 3.4.1). Furthermore, the main part of the algorithm consisting in the creation of the contraction mapping is agnostic *w.r.t.* these changes, unless considering differences also in the way that colours behave (a further different topic). However, we will point out with Footnotes 3 and 4 where the main modifications are needed for the algorithm to work on different graph structures.

More in detail, a graph G is stored as a `Graph` structure `G` which is described in Algorithm 6 and the contraction mapping β is stored, along with β^{-1} as a `ContractionMapping` structure `cMap` which is described in Algorithm 7. In particular, the various mapping described in Section 3.3.1 are not stored individually, but rather the field `becomes` and `revBecomes` are progressively updated following the four-steps pattern described.

3.3.3 Implementation of the algorithm

In the following, we provide and discuss the implementation of our contraction algorithm. At a high level, the algorithm takes in input a graph $G = (V, E, \gamma)$ and progressively evaluates the β-contraction operation $G' \leftarrow G/\beta$ in place, until G/γ is obtained (see Algorithm 8).

β-contraction

- **evaluateContractionMapping**
  - **becomesInitialisation**
    - **forall** $(v \in V)$ $\quad \texttt{becomes}[v] \leftarrow \min\left(\{v\} \cup N_\gamma(v)\right)$
  - **becomesUpdate**
    - **for** $(i = 0;\ i < n;\ i{+}{+})$ $\quad \texttt{becomes}[i] \leftarrow \texttt{becomes}[\texttt{becomes}[i]]$
  - **evaluateClusterSize**
    - **forall** $(v \in V)$ $\quad \texttt{cSize}[\texttt{becomes}[v]]{+}{+}$
  - **extractReverseBecomesMapping**
    - **let** I be a n-length array storing the first available position of $\texttt{revBecomes}[v]$
    - **forall** $(v \in V)$
      - $w \leftarrow \texttt{becomes}[v]$
      - $\texttt{revBecomes}[w][\texttt{I}[w]{+}{+}] \leftarrow v$
  - **revBecomesCompacting**
    - $i \leftarrow 0$
    - **foreach** $(v \in V)$
      - $\texttt{cSize}[i] \leftarrow \texttt{cSize}[v]$
      - $\texttt{revBecomes}[i{+}{+}] \leftarrow \texttt{revBecomes}[v]$
    - **shrink** cSize **and** revBecomes **down to** n'
  - **becomesCompacting**
    - **forall** $(v' \in V')$
      - **for** $(i = 0;\ i < \texttt{cSize}[v'];\ i{+}{+})$
        - $\texttt{becomes}[\texttt{revBecomes}[v'][i]] \leftarrow v'$
- **applyGraphContraction**
  - **edgesDestinationUpdate**
    - **forall** $(v \in V)$
      - **for** $(i = 0;\ i < \texttt{Nv}[v];\ i{+}{+})$
        - $\texttt{E}[v][i] \leftarrow \texttt{becomes}[\texttt{E}[v][i]]$
  - **colourClusterMerge**
    - **let** a be a n'-length bit-array repr. edges endpoints
    - **for** $(v' \in V')$
      - **reinitialise** a to false
      - **for** $(i = 0;\ i < \texttt{cSize}[v'];\ i{+}{+})$
        - $w \leftarrow \texttt{revBecomes}[v'][i]$
        - **for** $(j = 0;\ j < \texttt{Nv}[w];\ j{+}{+})$
          - $\texttt{a}[\texttt{E}[w][j]] \leftarrow \texttt{true}$
      - $\texttt{E}[v'] \leftarrow$ **convert** a in adjacency list
      - $\texttt{Nv}[v'] \leftarrow$ **count** true **in** a
      - **Shrink** E **and** $\texttt{Nv}[v']$ **down to** n'
  - **vertexContraction**
    - **Re-build** C according revBecomes

Fig. 3.3: Description of the β-contraction algorithm, *i.e.* an iteration of our novel γ-contraction algorithm, sketched as a function tree. Each leaf summarises a different phase by means of a simplified math-oriented pseudo-code.

Algorithm 8: `GraphColourContraction(G)`$\mapsto$`G'`

Input: The `Graph G` to be contracted
Output: The `Graph G'`, colour contraction of `G`

```
1 cMap ← evaluateContractionMapping(G);
2 while (cMap.n ≠ cMap.n') do
3 |   applyGraphContraction(G, cMap);
4 |   freeCMap(cMap); // The function opportunely frees allocated elements in cMap
5 |   cMap ← evaluateContractionMapping(G);
6 return G;
```

In the rest of this section, we focus on the structure of a single iteration of G/β. Let $G = (V, E, \gamma)$ and $G' = (V', E', \gamma') = G/\beta$ and let $n = |G|$, $m = \|G\|$, $n' = |G'|$, and $m' = \|G'\|$. We recall that here $V = \{0, \dots, n-1\}$ and $V' = \{0, \dots, n'-1\}$, hence, when we write $\forall v \in V$, we mean that a variable `idx_t` v is ranging in the interval $0 \leq v < n$. Then, β-contraction is made of two phases: the generation of the contraction mapping `cMap` (see Algorithm 9) and its application (see Algorithm 10). A summary of the whole procedure is sketched as a function calls tree in Figure 3.3 with a mathematic-friendly notation.

In the pseudo-codes that follow (Algorithm 8–20), we keep naming consistency as much as possible, hence we here list a few variable names that always play the same role and that we are not re-defining in each pseudo-code for sake of simplicity.

- `Graph *G` is the graph $G = (V, E)$ as defined in Algorithm 6.
- `Graph *G'` is the contracted graph $G/\beta = (V', E')$. Do note that the algorithms described work in-place to avoid multiple allocations, hence `G'` is never formally allocated.
- `ContractionMapping *cMap` is the contraction mapping as defined in Algorithm 7.
- `idx_t` i, j, k are (indices of) vertices within V, ranging in $[0, n)$.
- `idx_t` i', j', k' are (indices of) vertices within V', ranging in $[0, n')$.
- `idx_t` $i_{\text{IDX}}, j_{\text{IDX}}, k_{\text{IDX}}$ define (indices of) vertices within `E`$[\cdot]$ (*i.e.* edges), ranging in $[0,$ `Nv`$[\cdot])$.
- `idx_t` $i_{\text{BDX}}, j_{\text{BDX}}, k_{\text{BDX}}$ define (indices of) vertices within `revBecomes`$[\cdot]$, ranging in $[0,$ `cSize`$[\cdot])$.

Contraction mapping creation

For what concern the evaluation of the contraction mapping, the four steps described in Section 3.3.1 – *i.e.* the creation of (i) D, (ii) π, (iii) α, and (iv) β – are entirely managed through the sole map `becomes`, which is progressively updated until the map β is created; this is done in seven steps that involve also the creation of `revBecomes`, *i.e.* the inverse non-surjective map of `becomes`, and `cSize`, *i.e.* the sizes set $\{|S|, \forall S \in \mathcal{S}_\beta\}$. Do note that, since within the algorithm we consider

Algorithm 9: `evaluateContractionMapping(G)↦cMap`

Input: The `Graph G`
Output: The corresponding `ContractionMapping` holding β and β^{-1}

```
1 ContractionMapping cMap;
2 cMap ← contractionMappingAllocation(G.n);
3 becomesInitialisation(G, cMap);
4 becomesUpdate(cMap);
5 evaluateClusterSize(cMap);
6 extractReverseBecomesMapping(cMap);
7 revBecomesCompacting(cMap);
8 becomesCompacting(cMap);
9 return cMap;
```

Algorithm 10: `applyGraphContraction(G, cMap)`

Input: The `Graph G` to be contracted with the β-map `cMap`
Output: Nothing, contraction is performed in place

```
1 edgesDestinationUpdate(G, cMap);
2 colourClusterMerge(G, cMap);
3 vertexContraction(G, cMap);
```

$V' = \{0, \dots, n' - 1\}$, then the two maps $\hat{\alpha} : R \to \{0, \dots, n' - 1\}$ and $\alpha : R \to V'$ are equivalent.

More in detail, we have

1. The `ContractionMapping cMap` is allocated and initialised according to the size n of the graph G so that `becomes`$[v] = v, \forall v \in V$. In particular, `cMap`$.n^* \leftarrow n$ while the vectors in `revBecomes` are left non-allocated since their size is still not known. The total computational time of this phase is $O(n)$, just like the computational space (see Algorithm 11).
2. `becomes` is initialised so that it reflects the edges set B of the digraph D (see Algorithm 12). This is a greedy procedure that requires for each vertex to look at its whole neighbourhood and hence it takes $\sum_{v \in V} \partial(v) = 2m$ comparison operations, *i.e.* a total time of $O(m)$.
3. As we have abundantly discussed for D, `becomes` now defines a set of rooted trees (variable in height) such that each node within a tree is connected to its root via a directed path. Consequently, in order to make `becomes` behaving like $\hat{\pi} : V \to R$ (with R the set of roots), then $\forall v \in V$ it is sufficient to repeatedly update `becomes`$[v] \leftarrow$ `becomes`[`becomes`$[v]$] until a fixed point is reached (since a root $r \in R$ is such that `becomes`$[r] = r$). In particular, since by construction `becomes`$[v] \leq v, \forall v \in V$, then, if the operation is performed in-order *w.r.t.* v, we have that `becomes`$[v]$ is always a root and the update converges in one step.
   We can easily prove the correctness of this claim by induction on the size $|T|$ of each tree T; let $V_T = \{v_0, \dots, v_i, \dots\} \subseteq V$, then

base, $|T| = 1$ It is trivial that $\texttt{becomes}[v_0] = v_0$ and therefore, v_0 is mapped to the root of the tree it belongs to.

step, $|T| = l$ Let us consider the induction hypothesis that $\texttt{becomes}[v_i] = v_0, \forall i < l$. Then $\texttt{becomes}[v_l] = v_{\hat{i}}$ for some $\hat{i} < l$. Therefore

$$\texttt{becomes}[\texttt{becomes}[v_l]] \leftarrow \texttt{becomes}[v_{\hat{i}}] = v_0 \ .$$

Conversely, if the operation is not done in-order (*e.g.* in parallel implementations), in the worst case it requires $\log(\ell)$ iterations, where ℓ is the height of the deepest tree. In fact, it is rather clear that, in the worst case, the procedure halves the height of a tree at each iteration. The proof is pretty straightforward and it is out of context for the present work since we only discuss here a serial implementation.

The total cost of this step (see Algorithm 13) is of a few operations per node and, hence, it is bound by $O(n)$.

4. Once the map $\hat{\pi}$ is stored in `becomes`, it is rather simple to evaluate the size of each colour cluster in the vector `cSize` as $\texttt{cSize}[v'] \leftarrow |\{v \in V \mid \texttt{becomes}[v] = v'\}, \forall v' \in R$ (see Algorithm 14). This is necessary since, in order to efficiently apply first α over `becomes` without taking into account the order of the operation and then β over G, we first need to build $\hat{\pi}^{-1}$ (see next step) which requires allocating a proper-sized `revBecomes`. The time spent for this operation is negligible being $O(n)$.
5. With a suitably built `cSize` array, we can efficiently evaluate in `revBecomes` the map $\hat{\pi}^{-1} : R \to V$. In fact, it is rather simple to build it by progressively traversing `becomes` while keeping the last used position in an array of indices $\texttt{I}_{\text{BDXS}}$ (see Algorithm 15). Since the operation is performed in a single visit of `becomes`, then it takes $O(n)$ steps. From the computational space point of view, we require the allocation of $\texttt{I}_{\text{BDXS}}$ of size n' and of the vectors within `revBecomes` which are in number n'; however, the total space allocated for `revBecomes` is still bounded by $O(n)$, being equal to the number of vertices in V.
6. We can now apply the map $\alpha : R \to V'$ over the domain of $\hat{\pi}^{-1} : R \to V$, hence obtaining the map $\beta^{-1} : V' \to V$. To do so, we rearrange `revBecomes` and `cSize` in two novel arrays `revBecomes'` and `cSize'` of size n' (respectively of arrays and of vertices). As a consequence, n^* is now set to n'. The operation is completed in linear time $O(n)$ (see Algorithm 16).
7. Finally, we are ready to update `becomes` from representing $\hat{\pi}$ to representing β. This can be trivially done by a single traverse of `revBecomes` in linear time $O(n)$ (see Algorithm 17).

Contraction mapping application

We already mentioned in Section 3.3.1, despite non being strictly formal, that the application of the contraction mapping mainly consists of concurrently evaluating

Algorithm 11: `contractionMappingAllocation`$(n)\mapsto$`cMap`

```
1  ContractionMapping cMap;
2  cMap ← alloc(1, ContractionMapping);
3  cMap.n ← n;
4  cMap.n′ ← 0;
5  cMap.n* ← n;
6  cMap.becomes ← alloc(n, idx_t);
7  cMap.revBecomes ← alloc(n, idx_t*);
8  cMap.cSize ← alloc(n, idx_t);
9  for (i ∈ {0, ..., n − 1})
10     cMap.becomes[i] ← i;
```

Algorithm 12: `becomesInitialisation`(`G`, `cMap`)

```
1  col_t c; // colour of current vertex
2  idx_t b; // current best choice for becomes
3  foreach (i ∈ {0, ..., G.n − 1}) do
4      c ← G.C[i];
5      b ← i;
6      foreach (j_IDX ∈ {0, ..., G.Nv[i] − 1}) do
7          j ← G.E[i][j_IDX];
8          if (j < b ∧ G.C[j] = c)
9              b ← j;
10     cMap.becomes[i] ← b;
```

Algorithm 13: `becomesUpdate`(`cMap`)

```
   // This for-loop must be executed in order to work correctly
1  foreach (i ∈ {1, ..., cMap.n − 1) do
2      cMap.becomes[i] ← cMap.becomes[cMap.becomes[i]];
```

Algorithm 14: `evaluateClusterSize`(`cMap`)

```
1  foreach (i ∈ {0, ..., cMap.n − 1}) do
2      cMap.cSize[cMap.becomes[i]]++;
3  foreach (i ∈ {0, ..., cMap.n − 1}) do
4      if (cMap.cSize[i] > 0)
5          cMap.n′++;
```

Algorithm 15: `extractReverseBecomesMapping(cMap)`

```
1  idx_t *I_BDXS; // counter for j_BDX of revBecomes[becomes[i]]
2  I_BDXS ← alloc(cMap.n, idx_t);
3  foreach (i ∈ {0, ..., cMap.n − 1}) do
4      if (cMap.cSize[i] > 0)
5          cMap.revBecomes[i] ← alloc(cMap.cSize[i], idx_t);
6      else
7          cMap.revBecomes[i] ← NULL;
8  foreach (i ∈ {0, ..., cMap.n − 1}) do
9      i' ← cMap.becomes[i];
10     j_BDX ← I_BDXS[i']++;
11     cMap.revBecomes[i'][j_BDX] ← i;
12 free(I_BDXS);
```

Algorithm 16: `revBecomesCompacting(cMap)`

```
1  idx_t i' ← 0; // explicit assignment needed since it is used as a counter
2  idx_t **revBecomes', *cSize'; // compact version of revBecomes and cSize
3  cSize' ← alloc(cMap.n', idx_t);
4  revBecomes' ← alloc(cMap.n', idx_t*);
   // This for-loop should be performed in order for α̂ to respect the order of the roots in R
5  foreach (i ∈ {0, ..., cMap.n − 1}) do
6      if (cMap.revBecomes[i] ≠ NULL)
7          cSize'[i'] ← cMap.cSize[i];
8          revBecomes'[i'] ← cMap.revBecomes[i];
9          i'++;
10 free(cMap.cSize);
11 cMap.cSize ← cSize';
   // We only free revBecomes array, since revBecomes[·] are still used in revBecomes'
12 free(cMap.revBecomes);
13 cMap.revBecomes ← revBecomes';
14 cMap.n* ← cMap.n';
```

Algorithm 17: `becomesCompacting(cMap)`

```
1  foreach (i' ∈ {0, ..., cMap.n' − 1}) do
2      foreach (j_IDX ∈ {0, ..., cMap.cSize[i'] − 1}) do
3          cMap.becomes[cMap.revBecomes[i'][j_IDX]] ← i';
```

Algorithm 18: `edgesDestinationUpdate(G, cMap)`

```
1 foreach (i ∈ {0, ..., G.n − 1}) do
2     foreach (j_idx ∈ {0, ..., G.Nv[i]}) do
3         G.E[i][j_idx] ← cMap.becomes[G.E[i][j_idx]];
```

the graph G/β. Here concurrently means that we perform the contraction of all the colour clusters at the same time rather than evaluating the graph $G_1 = G/S_0$, $G_2 = G_1/S_1, \ldots, G' = G_{n'}/S_{n'}$. In particular, we complete this task in three distinct steps, as highlighted in Algorithm 10:

1. First and foremost, we apply $\beta : V \to V'$ (stored in `becomes` array) to the destination of the edges stored in $\mathtt{E}[\cdot]$; this can be done by simply traversing the edge adjacency list, hence tacking $O(m)$ operations (see Algorithm 18). Clearly, this lets `E` in an inconsistent state, since now maps sources in V to destinations in V', but prepares them for the next step.
2. We are now ready to effectively merge the adjacency lists according to β. In other words, given $S \in \mathcal{S}_\beta$, we want to build the adjacency list for $u' = \tilde{\alpha}(S) \in V'$ as $\{\{u', w'\} \mid w' \in \mathtt{E}[v], \text{ for some } v \in S\}$. In fact, after the previous step, we have $\mathtt{E}[v] \subseteq V'$. In particular, this can be done efficiently with the aid of an n'-length bit-array `a` that holds a dense representation of the adjacency list we are building; in fact, $v \in S$ we can traverse the edge list $\mathtt{E}[v]$ of v and set to true each position corresponding to a valid destination, *i.e.* $\mathtt{a}[z] \leftarrow \mathtt{true}, \forall z \in \mathtt{E}[v]$. This naturally gets rid of any duplicate edge, both formed during β-contraction and within the original graph[3]. Algorithm 19 shows the complete procedure for performing the merge.
   Only a single array `a` is required at a time, hence bounding the space complexity to $O(n')$ (plus the space occupied by the contracted graph). From the computational complexity point of view, this phase is the heaviest of the whole β-contraction. In fact, filling `a` $\forall v' \in V'$ requires in total $O(m)$ steps. Conversely, converting back `a` to a sparse representation requires $O(n')$ steps, which are executed $\forall v \in V'$, hence costing $O(n'^2)$. This bounds the complexity to $O(n'^2 + m)$.
3. Finally, vertices should be merged as well. In particular, since in our simplified representation we only keep track of the colours in a separate array `C` within the `Graph` structure, then we solely need to build a novel n'-length array `C'`, which can be simply done in $O(n)$ steps (see Algorithm 20)[4].

[3] The bit-array should be adequately substituted if more complex merge operations are required, like in the case of edge-weighted graphs. As an example, we can keep track of how many edges have been merged together by implementing a suitable function here.

[4] A merge function for vertices should be properly defined if more complex graphs are considered, like in the case of vertex-weighted graphs. As an example, we can keep track of how many nodes or edges have been merged in a vertex by implementing a suitable function here.

Algorithm 19: `colourClusterMerge(G, cMap)`

```
1  idx_t a_IDX; // index of a
2  idx_t d; // temporary value for Nv'[i']
3  idx_t *e; // temporary reference to E'[i'], i.e. sparse version of a
4  bool *a; // temporary dense n'-length bit-array for the adjacency list e
5  idx_t m' ← 0; // value m for G'
6  idx_t *Nv'; // neighbourhood degrees Nv for G', of size n'
7  idx_t **E'; // adjacency lists E for G', of size n'
8  Nv' ← alloc(cMap.n', idx_t);
9  E' ← alloc(cMap.n', idx_t*);
10 a ← alloc(cMap.n', bool);
11 foreach (i' ∈ {0, ..., cMap.n' − 1}) do
       // Reset a
12     foreach (a_IDX ∈ {0, ..., cMap.n' − 1}) do
13         a[a_IDX] ← false;
14     d ← 0;
       // fill a and deallocate old adjacency vectors
       // cycle over revBecomes[i'] vertices j
15     foreach (j_BDX ∈ {0, ..., cMap.cSize[i'] − 1) do
16         j ← cMap.revBecomes[i'][j_BDX];
           // cycle in the neighbourhood of j
17         foreach (k_IDX ∈ {0, ..., G.Nv[j] − 1}) do
18             k' ← G.E[j][k_IDX];
               // The condition i' ≠ k' can be dropped if self loops are allowed in G/β
19             if (i' ≠ k' ∧ ¬a[k'])
20                 a[k'] ← true;
21                 d++;
22         free(G.E[j]);
       // convert from bit-array a to adjacency list e
23     e ← alloc(d, idx_t);
24     k_IDX ← 0;
25     foreach (a_IDX ∈ {0, ..., cMap.n' − 1}) do
26         if (a[a_IDX])
27             e[k_IDX++] ← a_IDX;
28     Nv'[i'] ← d;
29     E[i'] ← e;
30     m' ← m' + d;
31 free(a);
32 free(G.Nv);
33 G.Nv ← Nv';
34 free(G.E);
35 G.E ← E';
36 G.m ← m';
```

Algorithm 20: `vertexContraction(G, cMap)`

```
1 col_t *C'; // Colour array C for G', of size n'
2 C' ← alloc(cMap.n', col_t);
3 foreach (i' ∈ {0, ..., cMap.n' − 1}) do
4     C'[i'] ← G.C[cMap.revBecomes[i'][0]];
5 free(G.C);
6 G.C ← C';
7 G.n ← cMap.n';
```

3.3.4 Space and time complexity

We mainly discussed space and time complexities bounds for β-contraction during the description of the algorithm phases in Section 3.3.3.

In particular, we have seen that, from the time-complexity point of view, the creation of the contraction mapping requires traversing a constant number of times the vertices and a single time the edges of the input graph, yielding a total complexity of $O(n+m)$ which is optimal. From the computational space point of view, the contraction mapping requires exactly 3 `idx_t` for the constants, n `idx_t` and one pointer for the `becomes`, n `idx_t` (n' after compaction) and one pointer for the `cSize`, and n `idx_t` and $n+1$ pointers ($n'+1$ after compaction) for the `revBecomes`, yielding a total of $3n+n'+3$ `idx_t` and $n+n'+5$ pointers during compaction. If we compare this number to the graph occupancy of $n+3$ pointer and $m+2n+2$ `idx_t`, we can safely say that its computational space is nearly negligible since, typically, $n = O(m)$. Only a few more `idx_t` variables are required (mostly for $\mathtt{I}_{\text{BDXS}}$) throughout the phase, bounding the total space cost to $O(n)$.

For what concerns the contraction mapping application, we have seen that the computational time cost of `colourClusterMerge` dominates over the entire algorithm. In fact, our solution with bit-array requires a $O(n')$-step elaboration for each of the n' contracted vertices; furthermore, all the edges need to be elaborated once again, hence yielding a total cost of $O(n'^2+m)$. While the m term is clearly optimal, the n'^2 term can certainly rise the computational cost up to $O(n^2)$ if the graph is only partially contracting (think *e.g.* a graph where all nodes are of different colours). However, it is important to point out that the same cost would be yielded from any kind of algorithm that elaborates the graph the way ours does (*e.g.* cf. Algorithm 5). Moreover, careful heuristics can (and should) be used a priori to determine whether it is convenient to contract the graph: we recall that our algorithm is motivated by the need of generating small representatives of big graphs, hence the assumption $n' \ll n$ usually holds true. Finally, for what concerns the computational space cost, we notice that most of the requirements are once again related to the `colourClusterMerge` algorithm; however, they are bounded by the creation of `a` requiring n' `bool` and the allocation of the novel components of the contracted graph. If we consider the latter, we obtain a space bound of $O(n'+m')$, however, as far as the compacted graph is

	Time complexity	Space complexity
`contractionMappingAllocation`	$O(n)$	$O(n)$
`becomesInitialisation`	$O(m)$	$O(1)$
`becomesUpdate`	$O(n)$	$O(1)$
`evaluateClusterSize`	$O(n)$	$O(1)$
`extractReverseBecomesMapping`	$O(n)$	$O(n)$
`revBecomesCompacting`	$O(n)$	$O(n')$
`becomesCompacting`	$O(n)$	$O(1)$
`evaluateContractionMapping`	$O(n+m)$	$O(n)$
`edgesDestinationUpdate`	$O(m)$	$O(1)$
`colourClusterMerge`	$O(n'^2+m)$	$O(n')^{\dagger}$
`vertexContraction`	$O(n)$	$O(1)^{\ddagger}$
`applyGraphContraction`	$O(n'^2+m)$	$O(n')^{\dagger}$
Total	$O(n'^2+m)$	$O(n)^{\star}$

Table 3.1: Computational time- and space cost bounds for the proposed β-contraction algorithm. If we consider the allocation of the contracted graph, space costs are: † $O(n'+m')$, ‡ $O(n')$, and * $O(n+m')$.

allocated, they are unavoidable. Hence it makes much more sense to exclude them from the bound, consequently yielding a space complexity bound of $O(n')$.

In particular, we report the computational costs related to Algorithms 9–20 in Table 3.1.

As we stated in Section 3.3.1 and showed with Observation 3.5, the local greedy approach used for building D in the first place does not guarantee the optimality for $\mathcal{S}_\beta$. One possible solution (as displayed in Algorithm 8) is to execute the complete algorithm more than once until convergence is reached. However, different solutions (possibly used in an adaptive way) can be devised; as an example, instead of iterating over the whole contraction, it might be convenient to iterate over `becomesInitialisation` assigning iteratively

$$\texttt{becomes}[v] \leftarrow \min\left(\texttt{becomes}[w] \;\middle|\; w \in \{v \cup E_\gamma[v]\}\right). \tag{3.16}$$

However, do mind that, while successive iterations of contraction are typically performed on much smaller graphs (hence requiring a limited time *w.r.t.* the first contraction), iteration over (3.16) works over the complete graph and might take up to n iterations (it is the case of a line graph where nodes are connected as $0 \sim n \sim n-1 \sim \cdots \sim 1$).

We stress out that particularly unfair and pathological graphs (*e.g.* graphs where $^n/_2$ nodes contract and the remaining $^n/_2$ does not) might yield unsatisfactory execution times. Nevertheless, particular structures like those are rare and can be spotted a priori by simple graph measures so that different approaches can be employed. As we see in the upcoming Section 3.4.1, in random graphs it is for example usual that

the algorithm takes a few iterations to stop and that the sizes involved sink fast, making the dimensions of n and m decrease by an order of two or more magnitudes as early as the first iteration, actually making the computational time of the upcoming phases negligible.

3.4 Benchmarks and results

We open this section with a discussion of a benchmark campaign over Erdös-Rényi random graphs. In particular, we evaluate execution times, contraction rate and β-contraction iteration over (randomly coloured) graphs of order up to $n = 5 \times 10^{+4}$ built under the binomial ER model with three different values for $p = p(n, c)$.

The rest of this section is then devoted to discussing a real-world use case, *i.e.* a web-page network that categorises Facebook pages according to four different owners: politicians, governmental organisations, television shows, and companies.

3.4.1 Application to random graphs

In order to provide useful benchmarks, we performed an experimental campaign over randomly generated graphs. We adopted the binomial Erdös-Rényi model discussed in Section 1.6.1 with variable sizes in terms of nodes n and probability p.

In particular, we recall that (for n large) the expected behaviour in terms of the number and the size of the connected components of a graph $\mathrm{ER}_{n,p}$ is strictly related to the two thresholds

$$p_1 = \frac{1}{n} \qquad \text{and} \qquad p_2 = \frac{\log(n)}{n} \,. \tag{3.17}$$

In particular, for $p < p_1$ no connected components of size larger than $\log(n)$ are expected, for $p_1 < p < p_2$ a few components, one of which is giant, are expected, and for $p > p_2$ the graph is expected to be connected.

It is worthwhile to point out that, for what concerns our analysis, values $p \in [p_1, p_2]$ are the most significant since $p \geq p_1$ yields graphs that contracts on a small number of nodes n' (*i.e.* $n' \ll n$) while $p \leq p_2$ generates graphs with a non-trivial number of connected components ($n' > 1$).

For what concerns colours, we decided to assign a vertex colouring uniformly at random from a given colour set C of c colours. It is easy to see that, if each vertex has probability $1/c$ to be of a given colour, then only n/c vertices are expected to share the same colour. As a consequence, it is mandatory to rescale the probability thresholds from (3.17) properly, hence obtaining two new thresholds

$$p^- = \frac{c}{n} \qquad \text{and} \qquad p^+ = \frac{\log(n) - \log(c)}{n} \cdot c \tag{3.18}$$

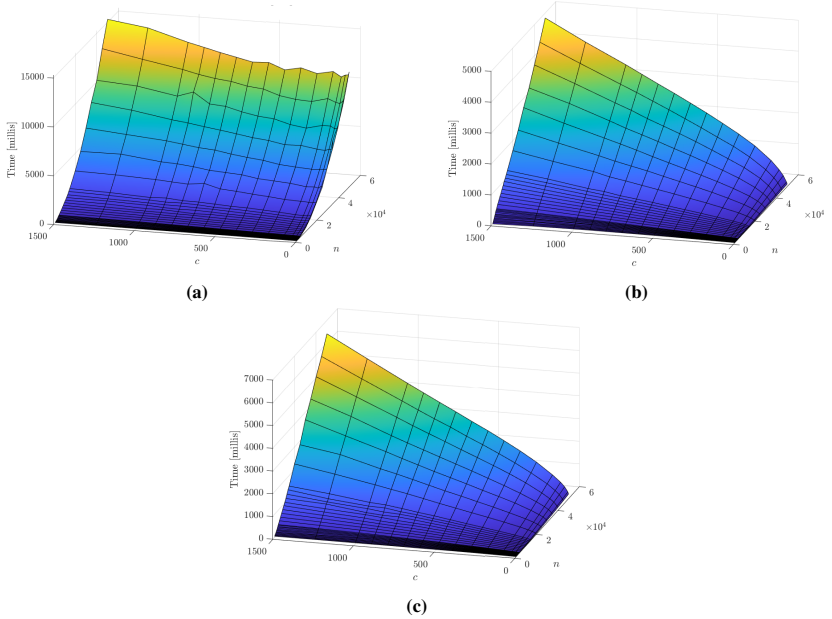


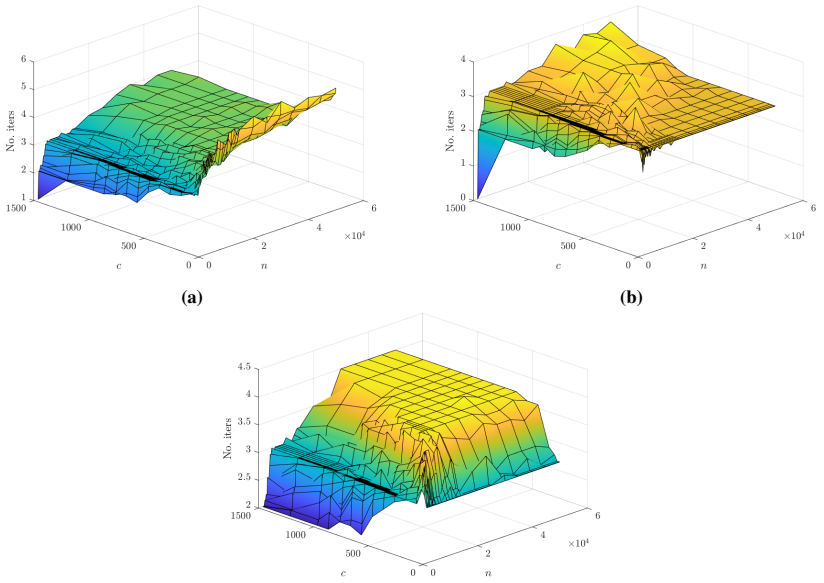
Fig. 3.4: Average contraction time obtained on c-coloured $\mathrm{ER}_{n,p}$ graphs at varying of c and n. Do note that z-axis scales are different. **(a)** $p = p^-$, **(b)** $p = p^+$, and **(c)** $p = p^*$.

respectively in place of p_1 and p_2. This naturally extends the ER model to consider colours since the expected behaviour (in terms of colour clusters) of a c-coloured graph $\mathrm{ER}_{cn,p}$ is the same as c single-colour graphs of the form $\mathrm{ER}_{n,p}$ then randomly connected between them (*i.e.* thresholds should be evaluated *w.r.t.* n/c nodes).

We performed a benchmark campaign evaluating different aspects of γ- and β-contraction at the varying of $n \in [1500, 50000]$, $c \in [1, 1500]$ and $p \in \{p^-, p^+, p^* = \frac{p^- + p^+}{2}\}$. In particular, we repeated multiple experiments per triple (n, c, p) scattered with different resolutions and we collected the resulting statistics. Averaged results in terms of computational execution time, number of β-contraction iterations before convergence, and the inverse ratio of contraction n'/n are reported respectively in Figures 3.4, 3.5, and 3.6.

Do note that $c > n$ makes little sense for the analysis since the expected number of vertices per colour would be less than one (also p^+ would be negative, hence generating disconnected vertices only). This is the reason why we kept c and n running over two quasi-disjoint consecutive ranges, being $n = 1500 = c$ the only intersection (yielding, as expected, 0 iterations on p^+).

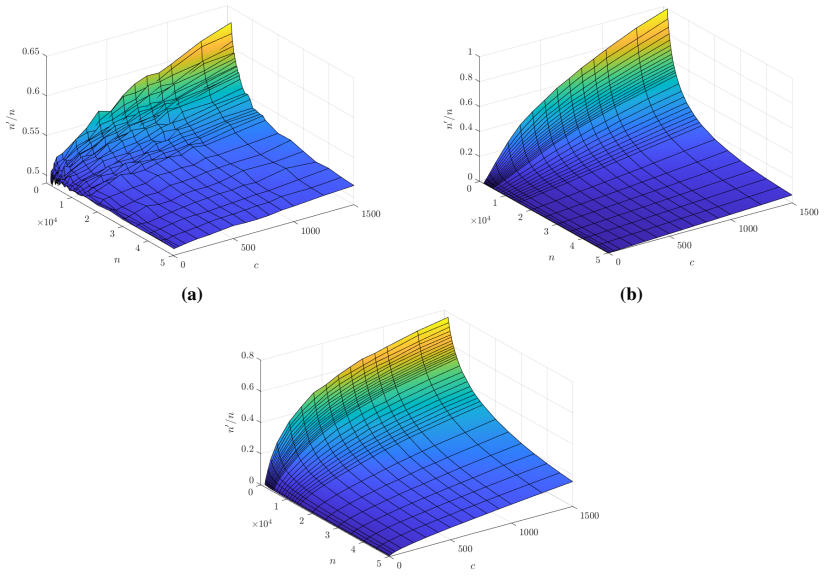
The thresholds in the ER model are sharp when $n \to \infty$, hence it makes sense that the most interesting conditions are when $n/c \gg 1$; when $n/c \sim 1$, in fact, the graph hardly contracts, actually causing little or no iterations. When n grows, contraction is much more effective, actually leading to much fewer colour clusters (as can be seen in Figure 3.6); this behaviour is less significant in the case $p = p^-$ since we expect



(a) **(b)**

(c)

Fig. 3.5: Average number of required β-contraction iterations on c-coloured $\mathrm{ER}_{n,p}$ graphs at varying of c and n. Do note that z-axis scales are different. **(a)** $p = p^-$, **(b)** $p = p^+$, and **(c)** $p = p^*$.



(a) **(b)**

(c)

Fig. 3.6: Average contracted vertices ratio n'/n obtained on c-coloured $\mathrm{ER}_{n,p}$ graphs at varying of c and n. Do note that z-axis scales are different. **(a)** $p = p^-$, **(b)** $p = p^+$, and **(c)** $p = p^*$.

to have a big number of small connected components, compared to $p = p^+$, where we expect to have only a few of them – actually $n' \sim c$. Following this behaviour, Figure 3.4 shows that the more the graph is connected (according to the choice of p), the less it takes to contract (despite m being larger). Such a conclusion seems to be counter-intuitive, however, it totally makes sense since the less the graph is connected, (i) the higher it will be n', causing each further iteration to be more complex and (ii) the higher the chances the local minima are not the global ones, hence requiring more β-contraction iterations to correctly obtain the γ-contraction, as can be seen in Figure 3.5. In fact, having a larger number of iterations over larger amounts of vertices makes the algorithm more computationally expensive.

3.4.2 A real-world use case: Facebook

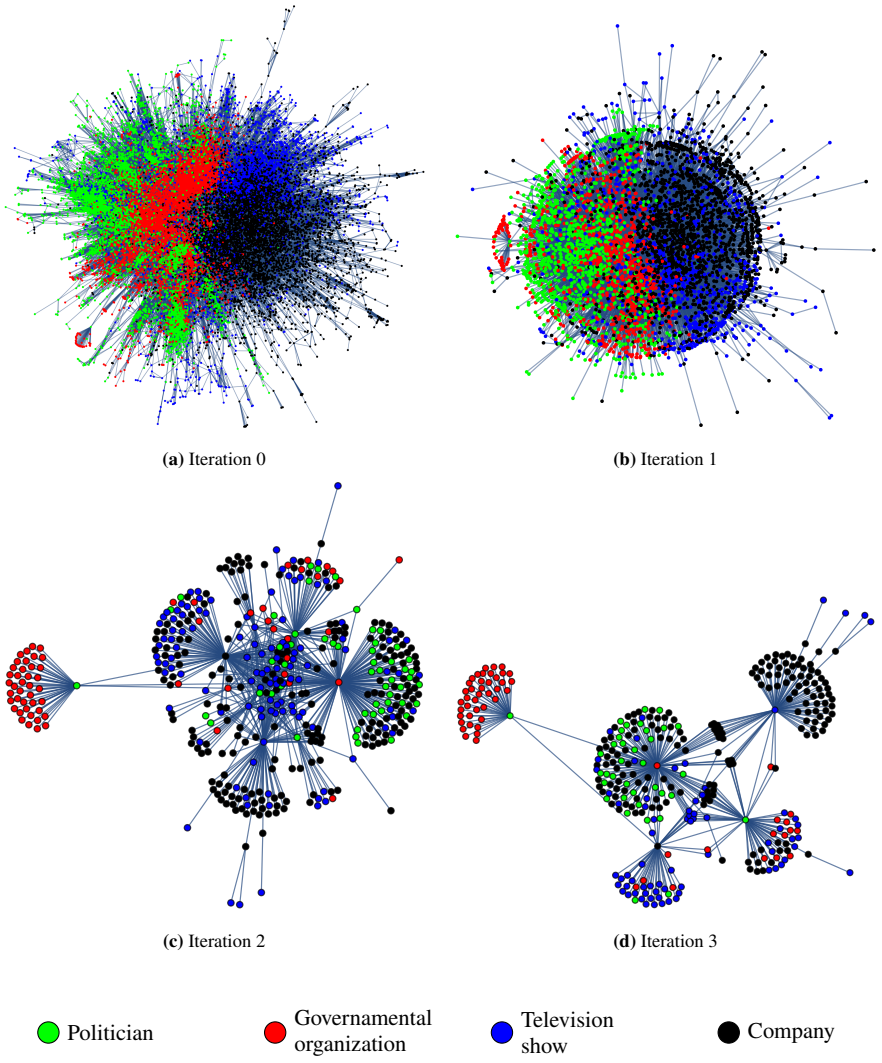
In order to provide a sample yet interesting use case for graph contraction, we have applied our algorithm to a real-world graph from the SNAP datasets [75]: the Facebook web-pages network $G_{\text{FB}} = (V_{\text{FB}}, E_{\text{FB}})$ collected in 2017 by [83].

V_{FB} represents $n_{\text{FB}} = 22470$ Facebook official pages organised in 4 categories (*i.e.* $c = 4$ colours) depending on their owner, namely: (i) politicians, (ii) governmental organisations, (iii) television shows and (iv) companies. V_{FB} nodes are linked via $m_{\text{FB}} = 170823$ undirected edges E_{FB} representing mutual likes between the pages. It is worth noticing that the numbers n_{FB}, m_{FB}, and c are in line with the E-R graphs we have used as benchmarks. In fact, considering an E-R graph with $n = 22470$ and $c = 4$, the expected number of edges ranges between $m^- = {}^{n^2}\!/_2 \cdot p^- = {}^{n \cdot c}\!/_2 = 44540$ and $m^+ = {}^{n^2}\!/_2 \cdot p^+ = m^- \cdot (\ln(n) - \ln(c)) = 387996$.

The contraction converges in three iterations (plus a contraction mapping evaluation to check convergency), and the running time is approximately $61ms$ on the test machine. The original graph and the three contraction iterations are pictorially displayed in Figure 3.7, where the graph is rendered using Wolfram Mathematica [72].

The intense computational effort lies in the first β-contraction iteration that takes $\sim 95\%$ of the total time, actually contracting vertices with a ratio $\sim 5 : 1$ and edges with a ratio $\sim 3.5 : 1$. Conversely, the second iteration is the one contracting more, with a ratio of $\sim 10 : 1$ and $\sim 35 : 1$ for vertices and edges respectively; despite this fact, it takes significantly less time than the first step given that the size of the graph has already been significantly reduced. The third and last β-contraction iteration contracts the last few local minima, actually reducing vertices and edges with a negligible ratio of $\sim 1.3 : 1$ and $\sim 1.8 : 1$ respectively.

By observing the contracted graph (Figure 3.7(**d**)) we can identify one major cluster per colour but "Politician" pages having two of them. "Company" pages first and "Television Show" pages second, represent the main glue among the central clusters, while "Politician" and "Governmental" pages are either part of the central clusters or nearly-isolated vertices. Furthermore, the "Politician" pages appear to be the only connection to many isolated "Governmental" pages; a similar behaviour can



(a) Iteration 0 **(b)** Iteration 1

(c) Iteration 2 **(d)** Iteration 3

Fig. 3.7: (a) The original page-page Facebook graph G_{FB} with self loops elided; $n = 22470$, $m = 170823$. **(b)** The first β-contraction iteration took 59ms, yielding $n = 4441$, $m = 48178$. **(c)** The second β-contraction iteration took 1ms, yielding $n = 437$, $m = 1380$. **(d)** The third and last β-iteration took less than 1ms, yielding $n = 334$, $m = 782$.

be identified for a big number of nearly-isolated "Company" pages that are connected to the "Television show" cluster only.

Overall, the above results show how useful network contraction is in trying to evince meaningful characteristics and behaviour from large datasets. A field expert studying a possibly very large network can benefit from the compact representation of the structure under observation offered by the contracted graph.

3.5 Conclusions

In this chapter, to the best of our knowledge, we provided the first formal characterisation of γ-contraction, which incorporates concepts related to graph connectivity, clustering and coloured representation. We explored the variadic nature of γ-contraction and proposed a weakened version under the name of β-contraction. Additionally, we have introduced a generic yet effective approach for β-contraction, providing both formal and algorithmic descriptions in its serial version. We have evaluated its complexity in detail and conducted a performance measurement campaign on random graphs. Furthermore, we have demonstrated real-world applications of the implemented β-contraction algorithm. Results are detailed and discussed, offering diverse use cases and heterogeneous application scenarios.

The results presented in this chapter demonstrate the utility and feasibility of coloured graph contraction in visualising and identifying issues and features related to large coloured networks, which are prevalent in today's world.

Future work will involve the introduction and evaluation of a parallel implementation of the methodology here proposed, leveraging the formal definition of the problem presented here. Additionally, further exploration of practical scenarios and more extensive performance evaluations on benchmark networks will help assess the advantages and limitations of the proposed approach. In this direction, the analysis of Polkadot transaction graph will also be the object of a future paper [283].

References

[61] T. Asano and T. Hirata. "Edge-contraction problems". In: *Journal of Computer and System Sciences* 26.2 (1983), pp. 197–208. DOI: 10.1016/0022-0000(83)90012-0 (cit. on p. 66).

[62] M. Bernaschi, A. Celestini, M. Cianfriglia, S. Guarino, F. Lombardi, and E. Mastrostefano. "Onion under Microscope: An in-depth analysis of the Tor Web". In: *World Wide Web* 25.3 (Mar. 2022), pp. 1287–1313. ISSN: 1573-1413. DOI: 10.1007/s11280-022-01044-z (cit. on p. 66).

[63] G. Blelloch and M. Reid-Miller. "Graph Contraction and Connectivity". In: *Lecture notes in Parallel and Sequential Data Structures and Algorithms*

(15-210) (Mar. 2022). Ed. by S. of Computer Science of Carnegie Mellon University. [PDF] (cit. on p. 67).

[64] M. Caprolu, M. Pontecorvi, M. Signorini, C. Segarra, and R. Di Pietro. "Analysis and Patterns of Unknown Transactions in Bitcoin". In: *IEEE Intl. Conf. on Blockchain*. 2021, pp. 170–179. DOI: 10.1109/Blockchain53845.2021.00031 (cit. on p. 67).

[65] R. D'Autilia and M. Spada. "Extension of Space Syntax Methods to Generic Urban Variables". In: *Urban Science* 2.3 (2018). ISSN: 2413-8851. DOI: 10.3390/urbansci2030082 (cit. on p. 67).

[66] D. Delling, A. V. Goldberg, A. Nowatzyk, and R. F. Werneck. "PHAST: Hardware-accelerated shortest path trees". In: *Journal of Parallel and Distributed Computing* 73.7 (2013), pp. 940–952. ISSN: 0743-7315. DOI: 10.1016/j.jpdc.2012.02.007 (cit. on p. 67).

[67] K. Erciyes. "Parallel Graph Algorithms". In: *Guide to Graph Algorithms: Sequential, Parallel and Distributed*. Cham: Springer Intl. Pub., 2018. Chap. 3, pp. 77–115. ISBN: 978-3-319-73235-0. DOI: 10.1007/978-3-319-73235-0_4 (cit. on p. 68).

[68] A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, and A. A. Akinyelue. "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects". In: *Engineering Applications of Artificial Intelligence* 110 (2022), p. 104743. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2022.104743 (cit. on p. 66).

[69] J. L. Gross, J. Yellen, and M. Anderson. *Graph theory and its applications*. Chapman and Hall/CRC, 2018. ISBN: 978-0429757099 (cit. on p. 67).

[11] S. Guattery and G. L. Miller. "A Contraction Procedure for Planar Directed Graphs". In: *Proc. 4th ACM Symp. on Parallel Algorithms and Architectures*. SPAA '92. San Diego, California, USA: ACM, 1992, pp. 431–441. ISBN: 089791483X. DOI: 10.1145/140901.141935 (cit. on pp. 24, 67).

[70] B. Harwani. *Practical C Programming: Solutions for modern C developers to create efficient and well-structured programs*. Packt Publishing, 2020. ISBN: 9781838647988 (cit. on p. 78).

[71] B. Hillier and J. Hanson. *The Social Logic of Space*. Cambridge University Press, 1984. DOI: 10.1017/CBO9780511597237 (cit. on p. 67).

[72] W. R. Inc. *Mathematica, Version 13.1*. Champaign, IL, 2022. URL: wolfram.com/mathematica (cit. on p. 93).

[73] Y. Issaoui, A. Khiat, A. Bahnasse, and H. Ouajji. "Smart Logistics: Blockchain trends and applications". In: *Journal of Ubiquitous Systems & Pervasive Networks* 12.2 (Mar. 2020), pp. 09–15. DOI: 10.5383/juspn.12.02.002 (cit. on p. 67).

[74] W. Jiang, K. Xu, Z.-Q. Cheng, R. R. Martin, and G. Dang. "Curve Skeleton Extraction by Graph Contraction". In: *Proc. First Intl. Conf. on Computational Visual Media*. CVM'12. Beijing, China: Springer-Verlag, 2012, pp. 178–185. ISBN: 9783642342622. DOI: 10.1007/978-3-642-34263-9_23 (cit. on p. 66).

[75] J. Leskovec and A. Krevl. *SNAP Datasets: Stanford Large Network Dataset Collection*. URL: snap.stanford.edu/data. June 2014 (cit. on p. 93).

[76] H. Meyerhenke, P. Sanders, and C. Schulz. "Partitioning Complex Networks via Size-Constrained Clustering". In: *Experimental Algorithms*. Feb. 2014, pp. 351–363. DOI: 10.1007/978-3-319-07959-2_30 (cit. on p. 67).

[77] H. Meyerhenke, P. Sanders, and C. Schulz. "Parallel Graph Partitioning for Complex Networks". In: *IEEE Transactions on Parallel and Distributed Systems* 28.9 (2017), pp. 2625–2638. DOI: 10.1109/TPDS.2017.2671868 (cit. on p. 67).

[78] G. L. Miller and J. H. Reif. "Parallel tree contraction part 2: Further applications". In: *SIAM Journal on Computing* 20.6 (1991), pp. 1128–1147. DOI: 10.1137/0220070 (cit. on p. 67).

[79] J. A. Mudrock. "A deletion-contraction relation for the DP color function". In: *arXiv preprint* (2021). arXiv:2107.08154 (cit. on p. 67).

[80] V. Patel, D. Shah, and N. Doshi. "Emerging Technologies and Applications for Smart Cities". In: *Journal of Ubiquitous Systems and Pervasive Networks* 15.02 (Mar. 2021), pp. 19–24. DOI: 10.5383/juspn.15.02.003 (cit. on p. 66).

[81] C. A. Philips. "Parallel Graph Contraction". In: *Proc. 1st ACM Symp. on Parallel Algorithms and Architectures*. SPAA '89. Santa Fe, New Mexico, USA, 1989, pp. 148–157. ISBN: 089791323X. DOI: 10.1145/72935.72952 (cit. on p. 67).

[82] R. Ponnusamy, N. Mansour, A. Choudhary, and G. C. Fox. "Graph Contraction for Mapping Data on Parallel Computers: A Quality–Cost Tradeoff". In: *Scientific Programming* 3.1 (1994), pp. 73–82. DOI: 10.1155/1994/715918 (cit. on p. 67).

[83] B. Rozemberczki, C. Allen, and R. Sarkar. *Multi-scale Attributed Node Embedding*. arXiv:1909.13021, github:benedekrozemberczki/MUSAE, snap:data/facebook-large-page-page-network.html. 2019 (cit. on p. 93).

[84] J. Soman, K. Kishore, and P. Narayanan. "A fast GPU algorithm for graph connectivity". In: *2010 IEEE Intl. Symp. on Par. & Distr. Processing, Workshops and Phd Forum (IPDPSW)*. IEEE. 2010, pp. 1–8. DOI: 10.1109/IPDPSW.2010.5470817 (cit. on p. 67).

[85] A. Valejo, V. Ferreira, R. Fabbri, M. C. F. d. Oliveira, and A. d. A. Lopes. "A Critical Survey of the Multilevel Method in Complex Networks". In: *ACM Comput. Surv.* 53.2 (Apr. 2020). ISSN: 0360-0300. DOI: 10.1145/3379347 (cit. on p. 67).

[86] D. B. West. *Introduction to graph theory*. Vol. 2. Prentice Hall, 2001. ISBN: 978-0130144003 (cit. on p. 67).

Part III
Pedestrian & vehicular traffic

Overview

Chapter 4
Managing crowded museums: visitors' flow measurements, analysis, modelling, and optimisation

This chapter covers a line of research on pedestrian tracking related to cultural heritage. The research was conducted between 2018 and 2023 in collaboration with Emiliano Cristiani, researcher at the "Istituto per le Applicazioni del Calcolo" *of the* "Consiglio Nazionale delle Ricerche" *(IAC-CNR), Alessandro Corbetta, assistant professor at the Department of Applied Physics of the Eindhoven University of Technology, and Pietro Centorrino, independent researcher and professor. Here I harmonise the contributions from two conference proceedings [276, 280], one journal paper [271], and a technical report [289].*

Abstract We present an all-around study of the visitors' flow in crowded museums: a combination of Lagrangian field measurements and statistical analyses enables us to create stochastic digital twins of the guest dynamics, unlocking comfort- and safety-driven optimisations.
We specifically present a graphs encoding for museum-like environments, where expert knowledge is injected as a total colouring. This encoding allows us to extract useful information and representations, including a natural notion of room metric.
We describe two cheap visitor tracking systems: (i) a simple manual Eulerian approach based on a self-devised tally counter and (ii) a more complex yet cheap semi-automatic Lagrangian IoT-based visitor tracking system relying on Raspberry Pi receivers, displaced in fixed positions throughout the museum rooms, and on portable Bluetooth Low Energy beacons handed over to the visitors. The signal intensity provides a proxy for the distance to the antennas and thus indicative positioning. However, RSSI signals are well-known to be noisy, even in ideal conditions (high antenna density, absence of obstacles, absence of crowd, etc.). Consequently, we present three approaches allowing to filter the beacons RSSI and accurately reconstruct visitor trajectories at room scale: a sliding window-based statistical analysis, a vanilla MLP neural network reconstruction, and a cascaded AI classifier based on the graph encoding of the museum.
Via a clustering analysis, hinged on an original Wasserstein-like trajectory-space metric, we analyse the visitors' paths to get behavioural insights, including the most common flow patterns. On these bases, we build the transition matrix describing, in probability, the room-scale visitors' flows. Such a matrix is the cornerstone of a stochastic model capable of generating visitor trajectories *in silico*.
We conclude by employing the simulator to enhance the museum fruition while respecting numerous logistic and safety constraints. This is possible thanks to opti-

mised ticketing and new entrance/exit management.
Our case studies are the museum of Galleria Borghese in Rome and the Peggy Guggenheim Collection in Venice (Italy), in which we performed multiple real-life data acquisition campaigns.

Keywords: IoT · Machine learning · Clustering · Tracking system · Museum simulator · Museum optimisation · Pedestrian dynamics · Total-coloured graph analysis

4.1 Introduction

The analysis of the behaviour of museum visitors has a long-standing tradition [108, 137] and underlies the capacity of profiling exhibitions, increasing visitors' comfort and safety, improving public reception, increasing the number of sold tickets, and enhancing artworks preservation [154]. Its feasibility has significantly grown during the last decades thanks, particularly, to the diffusion of Internet-of-Things (IoT) technologies [96, 107, 150], which enabled individual tracking needlessly of invasive structural modifications (*e.g.* as it happens with overhead optical tracking sensors [136]). The outstanding issue of visitors' management demands multidisciplinary skills connected to, amongst others, psychology, computer science, statistics, physics of complex systems as well as modelling and optimisation theory.

Museums curators are expected to achieve three complex and seemingly contradictory objectives: increasing the visitors' number, enhancing the experience quality, and preserving the artworks [154]. Accurately measuring and analysing the visitors' trajectories is an essential component towards these objectives and, specifically, when aiming at an efficient organisation of the exhibits [93, 141], the determination of adequate ticketing strategies, and also to verify if visitors' experience complies with managers' intents [143].

A complete workflow enabling the full control of visitors in a museum consists of several challenging steps, that we here summarise.

Visitors tracking The first goal is to understand the behaviour of visitors in terms of paths followed in the museum. Not all museums have predefined paths and sometimes more than one choice is possible [120]. Moreover, in large museums, it is rare that visitors see the whole exhibition [130]. A number of technologies exist for indoor tracking that is characterised by a trade-off between deployment complexity, invasiveness, and accuracy. Radio-based approaches, as considered in this work, enable room-level positioning accuracy: visitors' trajectories are rendered into sequences of visited rooms and related permanence times. At the price of more invasive and complex deployments, sometimes impossible in the context of cultural heritage, centimetre-level individual positioning can be also accomplished, e.g. via distributed grids of 3D scanners or video cameras.
Besides, psychological and sociological variables can be observed on side of paths, such as heart rate, skin conductance, emotional and aesthetic evaluations

of specific artworks [116, 141, 142], interactions with group-mates, degree of attention, boredom or fatigue.

Automatic systems could be complemented with manual activities, like paper-and-pencil annotations and questionnaires [117, 127, 141]. From questionnaires, one can estimate demographic-related and museum visit-related features [127] like age, gender, educational level, number of visits per year to museums, etc. After the visit, one can measure the degree of satisfaction, the relationship between perceived and real time spent in the museum [92], etc.

Behaviour understanding A number of variables can be estimated from visitors' trajectories: busy hours, movement patterns, length of visits, permanence times in each room, and number of stops. Two indicators are generally considered to quantify the importance of a specific exhibit, the *attraction power* (relative amount of people who have stopped in front of an artwork during their visit) and the *holding power* (average time spent in front of an artwork) [120].

Clustering and AI-based algorithms can be used for inferring, from the whole trajectories dataset, the *typical paths* or, equivalently, the *typical individual behaviours* inside the museum. Another interesting question regards the predictability of visitors' behaviour [100, 118, 125]: can a person who starts visiting the museum in a certain manner be immediately labelled as a visitor of a certain type?

Social behaviour can be observed too. For example, one can wonder, e.g., if people belonging to the same group follow the same path or they split, or whether individuals are attracted or repelled by crowding.

Museums digital twin Once statistics about visitors' trajectories and behaviour are available, it is possible to create an algorithm capable of generating real-like visit paths in the museum [88]. This is done by reproducing the movements of people from one room to another, duly determining their transition probability. Moreover, herding behaviour in social groups or the response to congestion and fatigue could be taken into account. A digital twin is able to reproduce virtual visitors moving in the museum with realistic behaviour, possibly in new (*i.e.* inexperienced) conditions. It is also possible to forecast the visitors' flow from some initial conditions, like, e.g. the visitor inflow at a given time.

Visitors' flow optimisation In order to use the museum digital twin as a managing tool, curators and organisers have to identify relevant *control variables* and *objectives*: regarding the former, one can, e.g., regulate the entrance flows, limit the maximum occupancy of selected rooms, increase the number of entrances or exits, set a maximal duration of the visit. The ticket price can obviously be controlled too.

Regarding objectives instead, one can aim at maximising the number of visitors, the pleasantness of the visit, the amount of information conveyed, or keeping the environmental parameters (e.g., temperature and humidity) in a given range, for the best conservation of the collection.

Once this is done, a museum digital twin can be profitably used to simulate different scenarios, aiming at matching the objectives while varying the control

variables. Here, optimisation algorithms like particle swarm and gradient-based methods can be used to automatise the search for a solution.

4.1.1 Relevant literature

The first step (visitors' tracking) is the one that has received the most attention in the literature, as it relates to pedestrian dynamics in general, i.e. beyond the museum context.

Focusing on (indoor) tracking systems, all kinds of technologies have been exploited, such as RFID [120], Wi-Fi [109, 114], Bluetooth [95, 97, 102, 125, 129, 131, 132, 133, 134, 146, 152, 154], video cameras [123], 3D scanners [98, 140]. An exhaustive review of these methods is out of the scope of the paper; we refer the interested reader to the papers [110, 129] for more references.

Different technologies require different degrees of visitors' involvement. For example, video cameras, 3D scanners, Wi-Fi or Bluetooth mass scans require no collaboration, while Bluetooth-based apps and RFID tags usually require some degree of interaction with the visitor. Measuring personal data like heart rate or skin conductance requires instead total involvement [116, 142]. Moreover, convincing people to participate in an experiment, for example by downloading and installing a smartphone app, can be difficult and time-consuming [134]. Sometimes free tickets could yield a good incentive [155].

Our contribution falls on the Bluetooth-based approach, with portable IoT devices broadcasting periodically their identities on the Bluetooth and/or Wi-fi networks. Measuring the strength of these signals, the so-called RSSI (i.e. the Received Signal Strength indication), with antennas deployed at different locations yields a mechanism to perform localisation and tracking (cf. also reviews [101, 138]). Such an approach has been employed in different fields, allowing tracking in healthcare facilities [115] and smart buildings [144].

In principle, high antennas densities could also allow precise localisation through signal tri/multi-lateration [148, 149, 151]. However, in historic buildings, a frequent location for museums, massive antennas deployments are impossible due to architectural constraints, while room-level tracking allows sufficient insights. Additionally, even in optimal conditions (e.g. line of sight to the beacon, absence of radio interference) RSSI values typically suffer from high fluctuations [89]. Literature provides useful filtering techniques like Bayesian [124] and Kalman [156] filtering, capable to improve distance readings up to 40% in accuracy. Nevertheless, museums often feature complex geometries rich in obstacles and – especially in old constructions – a wide mixture of thick and thin walls with narrow and wide doors. In combination with crowding, these yield even noisier RSSI signals with a quick decay, causing ambiguous or even void positioning readings. These constraints likewise jeopardise the success of any approach based on instantaneously "maximum RSSI" readings (argMax), even when the ambition is the sole room-level localisation.

The second step (Behaviour understanding) has also been investigated in great detail in connection with museums. Regarding *individual behaviour*, the predominant idea is to classify visitors into four categories based on the way they interact with the artworks: 'Ants' (tend to follow a specific path and observe extensively almost all the exhibits); 'Butterflies' (do not follow a specific path but are guided by the physical orientation of the exhibits; stop frequently to acquire more information); 'Fish' (most of the time move around in the centre of the room and usually avoid looking at exhibits details); 'Grasshoppers' (seem to have a specific preference for some pre-selected exhibits and focus their time on them, while tending to ignore the rest), see, e.g., [118] or [145] for the origin of this taxonomy.

Regarding *social behaviour* instead, the idea is to label visitors in six categories based on how they interact with group mates: 'Doves' (interested in other visitors while ignoring the environment); 'Meerkats' (stand side by side, expressing great interest in the exhibits); 'Parrots' (share their attention between exhibits and group members); 'Geese' (advance together with one visitor appears in the lead); 'Lone wolves' (enter the museum together and then separate); 'Penguins' (cross the space together while ignoring the exhibits), see, e.g., [103, 120].

Clustering techniques (e.g., k-means, hierarchical clustering, sequence alignment) have been used to assign every visitor trajectory (spanning from room-scale to continuum) to one of the groups described above, or to some given typical movement patterns [95, 102, 103, 118, 125, 127, 129, 132, 133, 153, 155]. This enables one to quantify the percentage of visitors belonging to each group. Note that typically the number of clusters is assigned *a priori* and this can be an important limitation. One crucial point for cluster investigation is the definition of a suitable *metric*, to measure the distance between trajectories, and aggregate (cluster) trajectories close to each other. Examples of such metrics devised at room scale can be found in [95, 125, 132]. In particular, [95, 132] propose a combination of well-known metrics defined in the space of characters strings (as trajectories can be suitably represented as sequences of characters), which is further corrected to take into account the differences in time of permanence in each room.

Regarding trajectory comparisons, let us mention also two other papers: [153] compares measured trajectories with those coming from a random walk simulator in order to understand which kind of visitors exhibits stronger patterns. The work [119] compares trajectories of visitors with and without audio guides in order to measure the impact of the transmitted information.

The third and fourth steps are also related to the rich pedestrian flow modelling literature: if one considers the museum as a continuous space as in [122], one can refer to differential (agent-based, kinetic, fluid-dynamic) or non-differential (discrete choice, cellular automata) models. See, e.g., [90, 99, 104, 105, 106, 113, 126] for some reviews, books chapters and books about this topic.

If one instead considers the museum as a graph – where the nodes represent the rooms of the museum and the edges represent connections amongst rooms – one can refer to some classical tools like transition matrices and deterministic/stochastic Markov chains with/without memory [114, 131, 147] in order to simulate a room-level walk in the museum (i.e. a trajectory on the graph).

Although many mathematical tools are available, examples of actual museums digital twins developed with the aim of reproducing, understanding and optimising visitors' behaviour are largely missing. This fact holds despite the fact that the path followed by visitors is evidently conditioned by the design of the exhibition galleries [93, 141]. An interesting attempt can be found in [111, 112]: the author describes a museum simulator and uses it to show that changes in the layout design of an exhibition result in different visitor circulation patterns. Unfortunately, that simulator can be hardly used in a museum with a very high density of artworks exposed, since it requires a complex calibration of many artwork-scale parameters which usually show a high variance between visitors. See also [88] for a rudimentary graph-based simulator and [135] for a simulator developed under the NetLogo software environment.

4.1.2 Chapter contributions

In this line of research, we perform an all-around investigation which includes contributions to all the four steps described above. Covering the whole process allows us to reach an unprecedented level of understanding and control of the museum, which unleashes the capability of improving deep modifications to the ticketing strategy as well as to the museum access management. Our results are validated on real visitors data acquired in the museum of Galleria Borghese (Rome, Italy) and the Peggy Guggenheim Collection (Venice, Italy). In more detail, the research unfolds along the following lines:

1. We propose a room-scale graph-structured representation of museums capable of capturing museum topology and expert knowledge. In practice, we equip both vertices and edges (representing rooms and connections respectively) with different colourings to represent architectural and conceptual constraints. We also show how to use this representation to extract compact representations of the museum via graph colour contraction, which will be essential for trajectory reconstruction.
2. We describe a cheap and easily reproducible semi-automatic data collection system, hinged on an IoT-based room-scale Lagrangian tracking system for museum visitors. From an operational point of view, one provides (consenting) individuals (or groups) with a portable IoT portable Bluetooth Low Energy (BLE) *beacon* (note that personal mobile phones, modulo privacy and randomisation issues, could be used likewise [154]). Antennas preemptively deployed (here realised by means of common Raspberry Pis) measure the RSSI of the periodic advertisements of each beacon. We also describe a simpler manual data collection system for Eulerian measurements of museum visitors. We employ these systems for multiple extended data collection campaigns which provide the high statistics measurements here employed.
3. We employ and filter the RSSI of the beacons to reconstruct individual visitors' trajectories. Due to the restricted space and the numerous architectural and

historical constraints each beacon is often captured by multiple antennas at the same time reporting highly fluctuating readings. Accurately reconstructing the trajectories in this setting defines a challenge *per se*. We propose a simple ML approach which outperforms standard sliding window processing, especially, when it comes to estimating the correct time of permanence in rooms. We further enhance this result by introducing an original cascaded methodology that combines an ensemble of trained classifiers, henceforth referred to as *localisers*, with the coloured-graph representation of the museum. This approach replaces a "complex" predictor with multiple simpler ones obtained by the injection of expert knowledge, which operates at different space- and time scales.

4. The total-coloured graph representation of a museum enforces a room-scale metric $\mathcal{D}$ and, consequently, a (room-level) trajectory-scale Wasserstein-inspired[1] distance $\mathcal{W}$. We employ such metrics as indicators to compare reconstructed trajectories with the corresponding ground truth, providing a more insightful performance analysis than the simple binary accuracy.
5. In order to get insights about visitors' behaviour, including the most common movement patterns, we analyse trajectories via statistical and clustering techniques. The previously introduced notion of distance further enforces an *ad hoc* trajectory clustering metric capable of capturing the geometrical properties of the museum. By using agglomerative hierarchical clustering analysis, only based on such metric (and with no *a priori* hypothesis on the number of clusters nor on their size), we can automatically unveil hard-to-see movement patterns that go well beyond the standard animal-inspired classification (see Section 4.1.1). As a by-product, we can also identify anomalous behaviours.
6. We employ statistical tools to build a probability transition matrix amongst museum rooms, which provides us with building blocks for a model capable of simulating *in silico* the museum visits. In particular, this enables us to forecast the path of visitors entering the museum from any room. Unlike the simulator presented in [111, 112], our simulator leverages the measured permanence time in each room on side of the probability of transition from one room to any other. This results in a tool easier to calibrate.
7. Finally, we employ the simulator to significantly increase the efficiency of the ticketing strategy and entrance/exit management. Our results suggest a way to enhance the museum fruition from both visitors' and curators' points of view while keeping the numerous constraints within the limits. We propose a suggested visiting path capable of significantly reducing the flux crossings and we start unveiling quantitatively how artwork positioning impact on their fruition.

[1] The Wasserstein distance was first introduced by Kantorovich in 1942 and then rediscovered many times. Nowadays, it is also known as Lip′-norm, earth mover's distance, $\bar{d}$-metric, Mallows distance. An important characterisation is also given by the Kantorovich–Rubinstein duality theorem.

4.1.3 Chapter organisation

In the rest of this chapter, we present our methods and original contributions alongside our field activity, hence describing methods and results as a whole. For this reason, we firstly describe in Section 4.2 our case studies, introducing the Galleria Borghese first (Section 4.2.1) – with its two-floor ring-structure and its peculiar original ticketing strategy – and the Peggy Guggenheim Collection then (Section 4.2.2) – with its problematic entrance system and its floor plan divided into two buildings and multiple areas. In Section 4.3, we introduce a topological representation for museum-like environments based on total-coloured (di)graphs that allow expert knowledge to be injected as a piece of typological information, or colour (Section 4.3.1); we further analyse how this pieces of information can be used to provide a coarse-grained representation of the museum (Section 4.3.2) and how a metric definition emerges (Section 4.3.3). In Section 4.4, we describe our IoT-based tracking system for Lagrangian datasets (4.4.1), our manual Eulerian data gathering approach (4.4.2), as well as the structure of our case studies and the datasets we collected (4.4.3). In Section 4.5 we discuss the trajectory reconstruction methods, ranging from the simplest sliding window approach (Section 4.5.1), passing on an intermediate multi-layer perceptron approach (Section 4.5.2), up to the more complicate cascaded selector (Section 4.5.3), our innovative high-level approach that builds over the coloured representation of the museum. We conclude the section by discussing the problem of missing data in beacon readings (Section 4.5.4) and comparing the result of our novel cascaded method under various implementation strategies (Section 4.5.5). In Section 4.6 we analyse the trajectories collected (Section 4.6.1), also fitting their statistical observable with known distributions, we discuss the dataset variability (Section 4.6.2), and we introduce a clustering approach and its results in the Galleria Borghese case study (Section 4.6.3). We further show how the coarse-scale representation of the museum can provide clustering on multiple levels of details and we adopt the Peggy Guggenheim Collection as an explanatory example (Section 4.6.4). In Section 4.7 we introduce the model which allows us to create a complete digital twin of the museum (Section 4.7.1) and simulate *in silico* the visitors' flow (Section 4.7.2); the resulting trajectories are discussed in the setting of Galleria Borghese case study (Section 4.7.3). In Section 4.8 we discuss a sample of model parameters and objectives in the Galleria Borghese case study and we employ them to find optimal strategies for the entrance/ticketing system (Section 4.8.1); we further discuss the possibility of removing the finite time horizon in visits (Section 4.8.2) and the changes, along with their impact, that the curator decided to apply (Section 4.8.3). In Section 4.9 we discuss analogous yet different results of our study in the Peggy Guggenheim Collection mainly focussing on suggesting preferential visiting paths to avoid flux crossing. In particular, we discuss the preliminary changes and their impact on the fruition of the museum and we close the section by describing the future changes that curators are currently planning (at the time of writing). The final discussion in Section 4.10 closes the chapter.

(a) **(b)**

Fig. 4.1: The two largest rooms in Galleria Borghese. **(a)** *Ratto di Proserpina* (room IV) located on the first floor. **(b)** Main area on the second floor (room XIV), part of the Pinacoteque. The room enumeration is given in Figure 4.2.

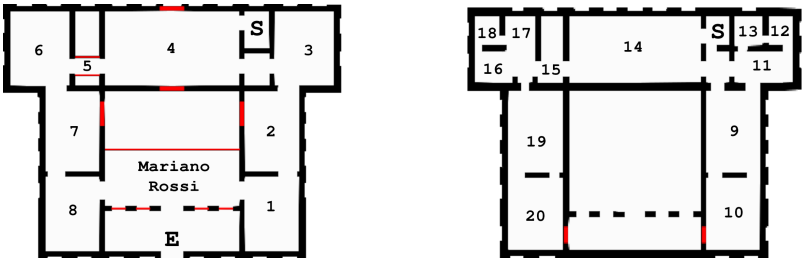
4.2 Case study description

In the following, we describe the structure and the peculiarities of our case-study museums. Despite the methodologies introduced in this chapter being mostly general (with some exceptions that are clearly pointed out)[2], we decide to present them along with their application and validation for the sake of readability. Consider, *e.g.*, the task of sketching a graph representation of a museum (see later Section 4.3): we do believe that the abstract description of the procedure is highly enhanced by our two well-grounded case-study applications.

4.2.1 The museum of Galleria Borghese in Rome

The world-renowned museum of Galleria Borghese (Rome, Italy), is a relatively small, two-floor museum with 3 entrances and 21 exhibition areas. Its sculptures and paintings attract visitors from all over the world, see Figure 4.1. On the main floor, the exhibition area is circular, while the second floor (Pinacoteque) is U-shaped, see Figure 4.2. Rooms are numbered but no obligatory exhibition path is assigned, so many people do not visit the rooms in their natural order. Moreover, the density of exhibits is so high that people often come back to already visited rooms multiple times to admire artworks missed during the previous passages. Congestion is frequent in some rooms, like the one which hosts Caravaggio's paintings. Audio guides are available on-demand and guided tours are allowed but subject to quota (both in number and size).

[2] Mathematical and numerical methods presented here can be used whenever one has to track people moving in a built environment through a non-predefined sequence of rooms. In addition, control and optimisation techniques are suitable whenever the flow of people can be controlled in some way, *e.g.* changing entrance doors and/or changing the visiting path dynamically.



(a) Floor 1 **(b)** Floor 2

Fig. 4.2: Floor plan of Galleria Borghese. The ticket office is located on floor -1 (not depicted) while floors 1 and 2 host the exhibition. The entrances/exits to the exhibition area are marked with E (main entrance, from outside the building) and with S (stairs, accessible from the ticket office). Lines in red represent inaccessible passages.

To cope with the many historical, artistic and architectural constraints, museum curators established in 1996 scheduled visits: tickets must be booked in advance and give access to the museum for a slot of 2h. Five slots per day are granted. The maximum number of visitors allowed in each slot is 360. Additionally, 30 tickets, called "last-minute", are sold 30 minutes after the beginning of each time slot. People can also decide which floor to start the visit from, within some limits[3]. At the end of each time slot, people are invited to leave, and the museum empties completely. Let us also note that many visitors enter without their smartphones since they must leave their personal bags in the wardrobe.

It is plain that the Galleria Borghese has specific peculiarities which make it rather unique in the world. Let us mention, in particular, the entrance system with quota and the fact that visitors often return to already visited rooms.

4.2.2 The Peggy Guggenheim Collection in Venice

The Peggy Guggenheim Collection is amongst the most important Italian museums for European and North American art of the first half of the 20th century, see Figure 4.3. It is located in Peggy Guggenheim's former home, Palazzo Venier dei Leoni, on the Grand Canal in Venice. It is part of the Solomon R. Guggenheim Foundation, whose constellation includes the Solomon R. Guggenheim Museum, New York, the Guggenheim Museum Bilbao, and the future Guggenheim Abu Dhabi.

The museum holds Peggy Guggenheim's personal collection, masterpieces from the Hannelore B. and Rudolph B. Schulhof collection, a sculpture garden as well as temporary exhibitions.

[3] The strategy here described is referred to the ante-pandemic situation, before the SARS-CoV-2 outbreak and the consequent limitations introduced to face Covid-19 disease.

(a) **(b)**

Fig. 4.3: Two of the most visited areas of the Peggy Guggenheim Collection. **(a)** the Calder Hall, main entrance to the permanent exhibition that connects it with both the Sculpture Garden and the Grand Canal terrace. **(b)** One of the areas of the Sculpture Garden.

The museum itself is located in two distinct buildings divided by the sculpture garden areas (see Figure 4.4). The permanent collection, which holds masterpieces by Jackson Pollock and Pablo Picasso amongst the others, is divided into 12 exhibition spaces arranged like an eight-like shape and shares the building with the Schulhof collection (6 exhibition spaces). The temporary exhibitions are located in a different building made of 14 rooms displaced in a circular fashion. The same building also holds a bookshop and a veranda, where visitors can take a break, charge their electronic devices and peek at the garden beneath. The two buildings are separated by the Sculpture Garden, a large open green area divided into 4 exhibition spaces from which visitors can reach a cafeteria as well. Finally, a terrace over the Grand Canal of Venice is reachable by the central room of Peggy's collection.

Conversely to the museum of Galleria Borghese, there are no fixed time slots and visitors can book tickets in advance choosing between six entrance times per hour. Audio guides are available on-demand and guided tours are allowed with limitation in number.

Curators of the museum cyclically change the disposition of the pieces of art, also replacing artworks frequently (about every three months) and a discrete number of temporary exhibitions are set up yearly. As a consequence, this offers a peculiar characteristic of the museum: visitor membership, a yearly-based subscription that allows members to freely access the museum and participate in dedicated events. Such a feature enforces interesting patterns in visiting habits since members typically know well the museum and tend to focus on specific rooms instead of visiting the entire exhibition area.

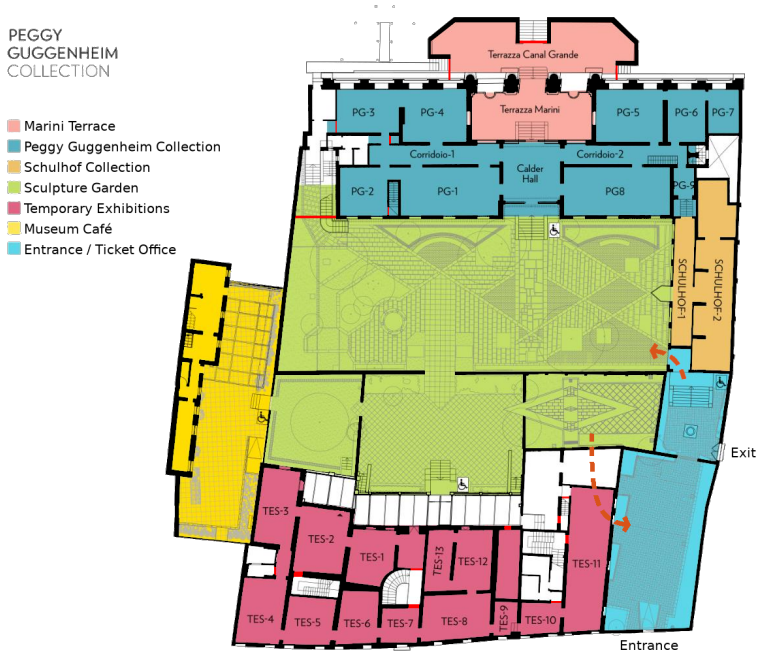


Fig. 4.4: Floor plan of the Peggy Guggenheim Collection. The entrance and the exit to/from the exhibition area are depicted with orange arrows. Red lines represent closed passages.

4.3 Representing museums as total-coloured (di)graphs

We consider here a generic museum divided into multiple floors; usually, a visitor explores one floor at a time instead of going back and forth between them repeatedly. Two kinds of connections arise from this example: same and different floor links. However, we can think of other different architectural constraints that may influence a visit, *e.g.*, different buildings. Similarly, we can provide conceptual subdivisions that concern rooms: *e.g.* if the exhibition is organised in different historical time periods, the typical visitor will likely explore a thematic area at a time instead of traversing them randomly.

In the following, we formalise this approach by first sketching the museum as a total-coloured graph and then extracting its emerging clustering. We also provide a metric definition to make useful insights available when it comes to comparing trajectory reconstruction methods.

4.3.1 Extraction of a total-coloured graph

In this section we provide, by means of graph theory (cf. Chapter 1), a natural formalisation of a museum as a graph equipped with different colourings to represent its architectural and conceptual constraints (cf. Figure 4.5).

Let $G = (V, E)$ be a graph, where $V = \{r_1, r_2, \ldots, r_n\}$ represents museum *virtual* rooms and E represents connections between them (*e.g.*, doors, staircases, paths, ...). Here, we adopt the term *virtual* room with respect to the physical room to cope with different data granularity: it might be beneficial to consider multiple virtual rooms inside a single big room (*e.g.* east and west part of a room) or a single virtual room representing many physical rooms (*e.g.* different gardens might be sketched as a single open area).

It is worth pointing out that, despite being unusual for cultural heritage applications, also one-way connections may occur (cf. Figure 4.5**(c)**), actually making G a digraph.

Once a room-wise digraph is designed, we can enforce conceptual subdivisions of the rooms and architectural constraints of the edges by means of a set of *typological* information sketched as a vertex-colouring γ_{V} and an edge-colouring γ_{E}, hence obtaining a total coloured digraph (cf. Section 1.2.7). More formally, we have

$$\gamma_{\mathrm{V}} : V \to C_{\mathrm{V}}, \qquad \gamma_{\mathrm{E}} : E \to C_{\mathrm{E}}\,, \tag{4.1}$$

where $C_{\mathrm{V}}, C_{\mathrm{E}} \subset \mathbb{N}$ represent a set of colours for vertices and edges respectively.

We add (if not present in the reconstruction) additional artificial rooms, marked with a dedicated colour, to represent the entrances and the exits of the museum: *ticket offices*, *bookshops*, or *outside areas* compose great examples. These rooms will serve in the trajectory reconstruction process as placeholders for the visitors before the visit begins, after it ends, and, possibly, whenever no signal is detected. Conceptually, these rooms shall be sources or sinks in the digraph, namely nodes with no incoming or outgoing edges, respectively.

4.3.2 Room clustering via colour contraction

We now consider the *colour contraction* of the digraph G (cf. Section 4.3), allowing us to extract different kinds of room clustering. The resulting clusters infer insights from the typological information stored as colours in vertices and edges.

In particular, we slightly extend the notion of colour contraction from Section 3.2, where only vertex colouring was considered. Depending on which colouring we rely on, we can in fact build different partitions of V by joining two rooms r_i and r_j in the same cluster as follows:

vertex-colouring If r_i and r_j are connected and share the same colour, then they are in the same cluster, say K_l, that is:

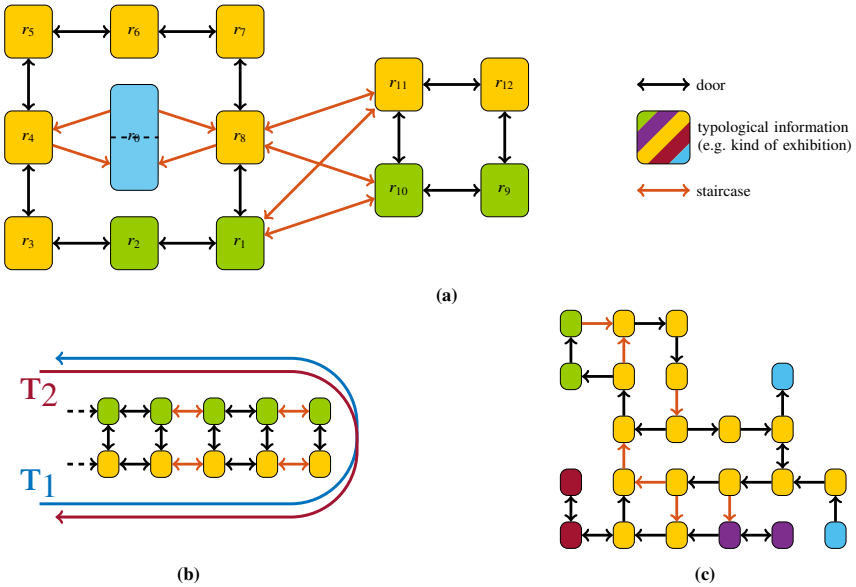


Fig. 4.5: Use cases for total-coloured graphs in three fictitious museums. **(a)** A museum inspired by the Galleria Borghese with three floors and two kinds of exhibitions. **(b)** A segment of a museum where all three floors host two kinds of exhibitions. Without considering the vertex-colour information, the two trajectories τ_1 and τ_2 would be very similar, however, they clearly are opposite and therefore different. **(c)** An example of a directed museum with multiple kinds of exhibitions based on Vatican Museums (Rome, Italy). Entrance and exit artificial rooms are at different locations.

$$\text{if} \quad (r_i, r_j) \in E \quad \text{and} \quad \gamma_{\text{V}}(r_i) = \gamma_{\text{V}}(r_j) \quad \text{then} \quad r_i, r_j \in K_l \,. \tag{4.2}$$

Note that two rooms do not need to be directly connected to belong to the same cluster. Hence, we aggregate all the rooms sharing the same conceptual information which are reachable one from the other (cf. 4.6**(a)**). This corresponds to regular colour contraction G/γ, deeply discussed in Chapter 3.

edge-colouring If r_i and r_j are connected by an edge of some specific colour $C' \subsetneq C_{\text{E}}$, then they are in the same cluster, say K_l, that is:

$$\text{if} \quad e = (r_i, r_j) \in E \quad \text{and} \quad \gamma(e) \in C' \quad \text{then} \quad r_i, r_j \in K_l \,. \tag{4.3}$$

As an example, by selecting door connections, this technique contracts all the rooms within the same floor of the museum in single clusters (cf. 4.6**(b)**).

multi-colouring We can also combine the two previous methodologies, therefore obtaining a clustering of the rooms based on both architectural and conceptual constraints (cf. 4.6**(c)**).

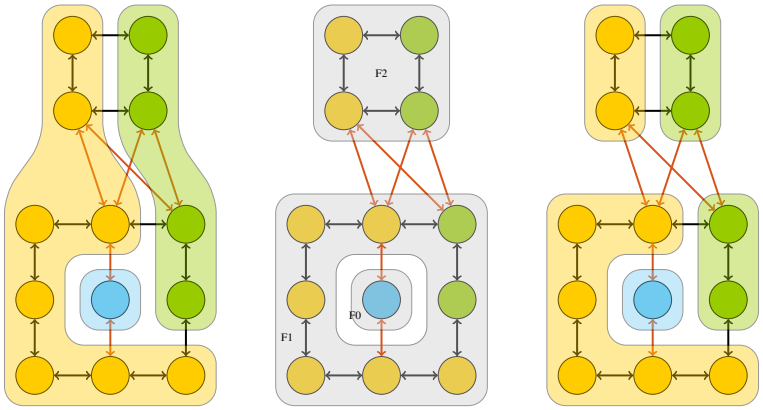


Fig. 4.6: Colour contractions applied to the graph representation in Figure 4.5(**a**). **(a)** vertex-colouring: the two kinds of exhibitions join in one single cluster each, suggesting that visitors explore them independently. **(b)** edge-colouring: by contracting over door edges ($\{\leftrightarrow\}$) we obtain a floor-based representation, prompting the idea that visitors would rarely deal with stairs if it is not needed. **(c)** total-colouring: contracting over both vertex-colouring and edge-colouring summaries the hints from the clustering **(a)** and **(b)**, detecting four areas, likely visited independently.

4.3.3 Room metric enforced by the total-coloured graph

The definition of a graph related to museum exhibition spaces naturally yields a concept of distance $\mathcal{D}$ between the nodes. A "vanilla" distance between two rooms for a colourless graph can be in fact defined by the least number of edges traversed to reach one room from the other (cf. Section 1.2.5). Determining such a distance corresponds to the well-known *shortest path problem*, which naturally extends to include edges weighing a non-unit distance (i.e. weighted edges) [87].

In the setting of (typically edge) coloured graphs, the colours themselves can be interpreted as weights, e.g. the weight of a colour can be proportional to the time employed to traverse that specific kind of edge. Therefore, the assignment of a weight $w(\gamma(\cdot))$ to each colour C_{E}, *i.e.* a mapping:

$$w : C_{\text{E}} \to \mathbb{N}^{+} \qquad (\mathbb{R}^{+}, \text{ more in general}) \,, \tag{4.4}$$

enforces also a distance notion between rooms[4].

Whenever two rooms differ in colour, since their typological information differs (cf. Figure 4.5(**b**)), a penalty β is applied to their mutual distance. Note, whenever $\beta > 0$, the metric $\mathcal{D}$ does not satisfy the relaxation principle [6] (which can be still applied by adding β *a posteriori*).

[4] As an additional feature, it can be useful to reduce the cost of the first door transition, actually decreasing the weight of short transitions that can occur if visitors stand still by the entrance door.

	r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}
r_0	0	11.5	10.5	11.5	12.5	12.5	11.5	10.5	11.5	20.5	20.5	21.5	21.5
r_1	11.5	0	1.5	2.5	3.5	1	2.5	3.5	4.5	10	10.5	11.5	11
r_2	10.5	1.5	0	1	2	2.5	3	4	3	10.5	10	11	11.5
r_3	11.5	2.5	1	0	1	3.5	4	3	2	11.5	11	12	12.5
r_4	12.5	3.5	2	1	0	4.5	3	2	1	12.5	12	13	13.5
r_5	12.5	1	2.5	3.5	4.5	0	1.5	2.5	3.5	11	11.5	12.5	12
r_6	11.5	2.5	3	4	3	1.5	0	1	2	12.5	12	13	13.5
r_7	10.5	3.5	4	3	2	2.5	1	0	1	13.5	13	14	14.5
r_8	11.5	4.5	3	2	1	3.5	2	1	0	13.5	13	14	14.5
r_9	20.5	10	10.5	11.5	12.5	11	12.5	13.5	13.5	0	1.5	2.5	1
r_{10}	20.5	10.5	10	11	12	11.5	12	13	13	1.5	0	1	2.5
r_{11}	21.5	11.5	11	12	13	12.5	13	14	14	2.5	1	0	1.5
r_{12}	21.5	11	11.5	12.5	13.5	12	13.5	14.5	14.5	1	2.5	1.5	0

Table 4.1: Distance matrix $\mathcal{D}$ from the total-coloured graph in Figure 4.5**(a)**. We weigh '1' the door connections ($w(\leftrightarrow) = 1$), '10' the staircase links ($w(\leftrightarrow) = 10$), and $\beta = +0.5$ the distance between two rooms not sharing the same room-colour.

The distance $\mathcal{D}$ is discrete in the vertices of the graph, therefore it can be represented as a $n \times n$ matrix. If the connections in the underlying graph are bi-directional (G is undirected) the distance is commutative; however, since in general G is a digraph, no assurance can be made *a priori* on the symmetry of $\mathcal{D}$. Formally, we have:

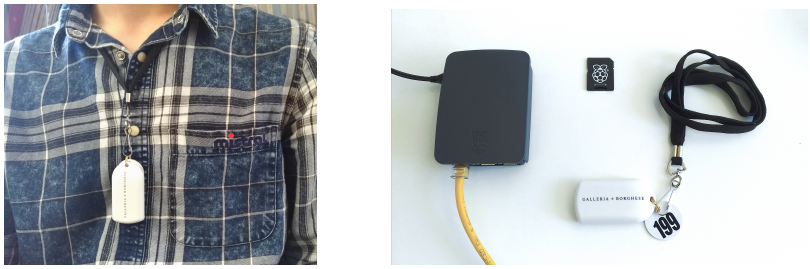
$$\mathcal{D}_{i,j} \quad = \quad \mathcal{D}(r_i, r_j) \quad = \quad \beta \cdot \mathbb{1}_{\{\gamma_v(r_i) \neq \gamma_v(r_j)\}} \quad + \quad \sum_{\substack{e \in \Gamma \\ \Gamma \text{ shortest path} \\ \text{between } r_i \text{ and } r_j}} w(\gamma(e)) \,, \tag{4.5}$$

where $\mathbb{1}_{\{\epsilon\}}$ is the indicator of the event ϵ, i.e. $\mathbb{1}_{\{\gamma_v(r_i) \neq \gamma_v(r_j)\}} = 1$ if $\gamma_v(r_i) \neq \gamma_v(r_j)$ and 0 otherwise.

As an example, we report in Table 4.1 the distance matrix $\mathcal{D}$ associated with the graph introduced in Figure 4.5**(a)**.

4.4 Data collection

In this section, we tackle the problem of collecting data. We introduce first the main source of data, our original IoT tracking system that allows us to gather individual



(a) (b)

Fig. 4.7: Main components of the IoT tracking system **(a)** Sample visitor wearing the BLE beacon. **(b)** Raspberry Pi used as Bluetooth antenna to receive beacon signals and measure their RSSI.

visitor trajectories. Secondly, we describe our complementary dataset of room-wise crowd statistics. In numerical simulations, the first method is often referred to as *Lagrangian*, recalling the Lagrangian approach from fluid dynamics which deals with *individual particles* and calculates the trajectory of each particle separately. The second method, on the other hand, is often referred to as *Eulerian*, since it deals with *concentration of particles* and evaluates their overall diffusion.

4.4.1 IoT visitors tracking system

Cultural heritage sites worldwide typically have heavy limitations in the possibility of displacing (electrical) devices. It is the case, *e.g.*, of the Galleria Borghese, which is covered by frescoes and paintings, and the Peggy Guggenheim Collection, with its minimalist design and no places to hide large devices. To cope with these historical and architectural constraints, we developed a non-invasive radio-based IoT measurement solution delivering (virtual) *room-level* visitor trajectories. In this sense, we define a trajectory as a finite sequence

$$\mathrm{t} = (\mathrm{t}_0, \mathrm{t}_1, \ldots, \mathrm{t}_{T-1}), \tag{4.6}$$

where t_t states the room (within a finite virtual room set $\{r_1, r_2, \ldots, r_n\}$) in which the visitor is located at discrete time t, considering a regular time sampling $t \in 0, 1, \ldots, T-1$.

Figure 4.7 shows the main components of the tracking system, which consists of:

Transmitters We employed small BLE beacons as transmitters due to their cheapness and their compactness. We provide a beacon to each (consenting) visitor to track their position inside the museum after a small briefing about the experiment, see Figure 4.7**(a)**. The beacon transmitted a signal at +4dB with iBeacon encoding [128], which carries a unique identifier (UUID), a major, and a minor (for filtering purposes).

Fleet of receiving antennas We employed Raspberry Pi 3B+ (RPi) as receivers due to their affordable cost and limited power consumption. An RPi is a single-board computer with embedded Bluetooth and Wi-Fi modules that can be powered via regular commercial power banks, see Figure 4.7**(b)**. RPis were located along the museum in fixed positions and labelled with convenient names, see later Figure 4.12**(a)** and Figure 4.15**(a)**. A high-level Python3 service running on the RPis was used to scan continuously the surrounding area, searching for beacons signals, and packing data by aggregating multiple readings from the same beacon.

Data packets Data packets are created to transmit/store readings aggregated on a Δt time interval ($\Delta t = 5s$ in our experiments). Aggregation is useful to limit the amount of data to be transmitted and stored. Each data packet corresponds to the readings of a single couple RPi–beacon in a Δt time interval and it is made of (i) Beacon identity, (ii) RPi identity, (iii) timestamp, (iv) number of readings within the interval, and (v) average or the RSSI readings[5]. We recall that the RSSI, or *received signal strength intensity*, is a relative index in arbitrary units representing the intensity of the beacon signal strength compared to the ground noise. Do also note that no unified standards are designed to what concerns RSSI values, hence we employed the most commonly used: values are strictly negative and the stronger signals, the closer they are to zero (cf. Figure 4.8).

Central server We employed a proprietary server to receive data packets in real time coming from all RPis via an internet connection. The server then stored data packets in a central (Postgre)SQL database along with the reception timestamp and other information from the museum and the RPi position. In the unfair situation where no internet connection is available at reading time, data packets are saved on RPis and later imported into the database. This situation is, in practice, very common in old cultural heritage sites where no Wi-Fi or LAN is available in each and every room; it was *e.g.* the case of Galleria Borghese at the time the readings were taken. However, it can be typically bypassed by setting up a custom internet connection system by means of commercial routers and Wi-Fi extenders.

Such a system proved to be reliable also with dozens of beacons broadcasting at the same time within a small area.

A data sample is reported in Figure 4.8, showing the history of the readings of a single beacon during a visit. The analysis of the raw data immediately confirms that the RSSI signal suffers from high fluctuations (cf. [89]). This means that a beacon fixed in the middle of a room is not received with a constant RSSI, and the RSSI of two equidistant beacons might not be the same. In addition, two other important difficulties arise from data analysis:

[5] It is worthwhile to point out that this structure was improved over time; *e.g.*, during our first campaign in Galleria Borghese only the first RSSI reading was stored and no aggregation procedures were taken into account. For this and many other practical reasons, also an identification of the code version by means of the commit hash was added as a piece of information within the data packet.

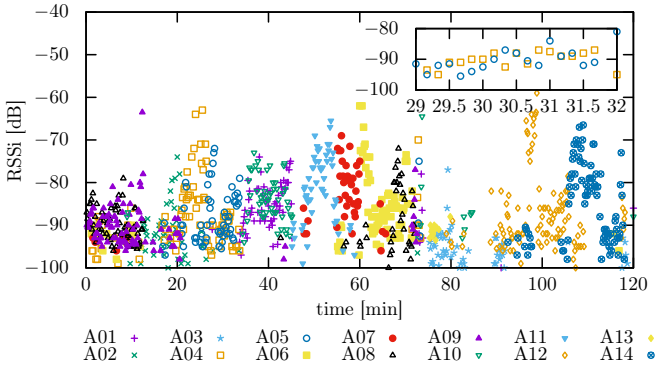


Fig. 4.8: Typical raw RSSI data throughout a 120-minute visit as recorded by the 14 RPis receiving antennas in Galleria Borghese. Data are re-sampled every $\Delta t = 10\,\mathrm{s}$ and RPis antennas are distinguished by markers. Inset: between minutes 29 and 32 the visitor is detected by both A04 and A05, and the maximal RSSI strongly oscillates between the two. As a consequence, the signal strength is insufficient to associate unambiguously a visitor to a location.

1. A single beacon can be detected by multiple antennas at the same time. RSSI is used to resolve the ambiguity but high fluctuations make such task rather hard.
2. Some areas of the museum could not be covered at all (*e.g.*, due to the impossibility to deploy antennas, as it can happen in narrow corridors or staircases).

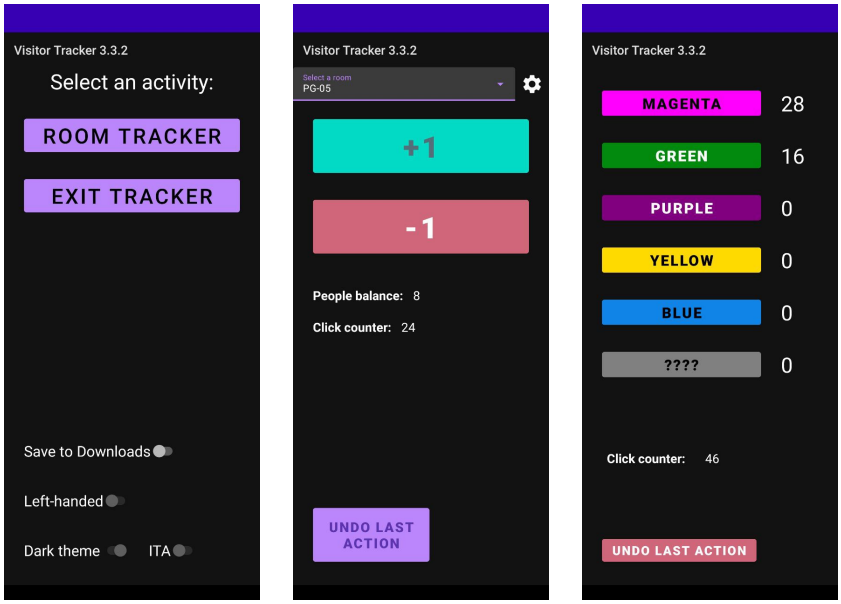
Finally, let us also mention the possibility – which we consider very rare – that visitors wearing beacons could be influenced by the fact that they feel tracked, cf. [141, 154].

In the upcoming section, we describe how the raw RSSI data from the transmitters is processed to estimate individual trajectories.

4.4.2 Eulerian data gathering

Concurrently with the (semi-automatic) Lagrangian data gathering technique introduced in the previous section, we developed a simple yet useful smartphone application to perform Eulerian data gathering (manually).

The application allows an operator within a room to report time-by-time the visitors' presence inside the room by the use of two simple buttons: '+1' and '-1' (cf. Figure 4.9). Writing such an application from scratch (instead of using one of the many tally counters already available online) allowed us to customise it and, in particular, store each button press along with the timestamp of the tap. Figure 4.10 reports a typical measurement of the crowd level inside a room in the Peggy Guggenheim Collection on a weekday. Do note that larger rooms can easily be covered and surveyed by multiple operators working together (*e.g.*, positioned at



(a) **(b)** **(c)**

Fig. 4.9: Visitor Tracker application. **(a)** Main menu of the application that allows to set preferences and choose amongst different activities (here only two are reported for the sake of simplicity). **(b)** Main activity of the application, *i.e.* counting the number of visitors inside a given room. The current sum and a tap counter are provided for the sake of verifiability. **(c)** Secondary activity of the application developed specifically for measuring changes in Galleria Borghese, *i.e.* counting the number of visitors with a given coloured stamp (see later Section 4.8.3 for more details). The current number of counted visitors per colour and a tap counter are provided for the sake of verifiability.

each access to the room). In a similar fashion the entire museum or part of it (*e.g.* a wing) can be tracked as well, here with the downside that no intermediate error checks can be properly made.

Furthermore, the application can be used also to track other visitors' habits, like the natural emerging trend of going left/right after entering an exhibition area and how this changes in the presence of obstacles or signage (see later Section 4.6.3 for a discussion of visiting pattern in the Peggy Guggenheim Collection and Section 4.9 for an example of signage and its impact).

Such a solution, despite being unfeasible on a large scale (due to the high requirement in manpower), typically produces high-quality data that can serve both *per se* for fine statistics purposes and as a benchmark for the coarser grain data that can be extracted from the Lagrangian measurements. In fact, providing all the visitor trajectories are available, the occupancy of a room can be evaluated time-by-time by simply counting how many trajectories are in that room at each time; however, in our solution this is typically not the case, since only a fraction of visitors is provided with beacons, hence making such a reconstruction much noisier.

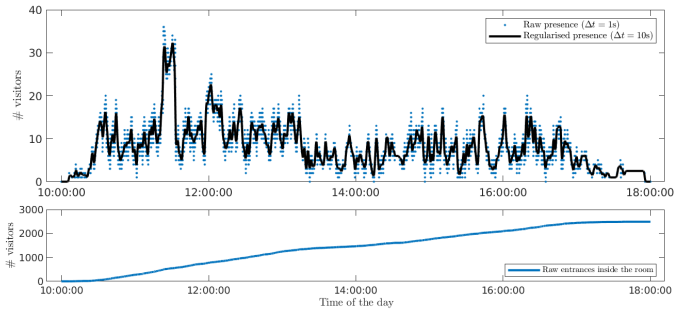


Fig. 4.10: Sample of Eulerian data gathered in the Peggy Guggenheim Collection (PG-6) during a day-long measurement. The top panel displays the crowd level with a resolution of $\Delta t = 1$s; a regularisation Gaussian filter is applied over a downsampling to $\Delta t = 10$s to better highlight the trend. The bottom panel displays how many people enter the room during the same day: as can be seen, the in-flux is rather constant and slows down around lunchtime. Do note that a visitor re-entering is counted twice since Eulerian measurements do not follow individuals.

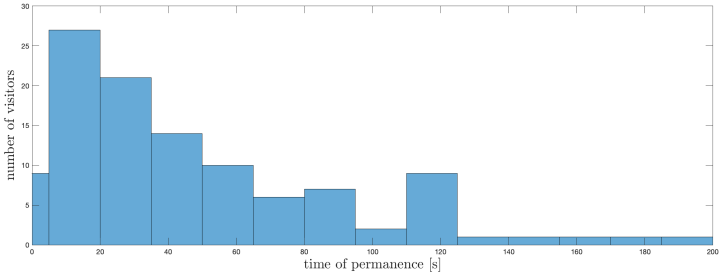


Fig. 4.11: Sample of visitors' time of permanence in front of a single artwork gathered in the Peggy Guggenheim Collection. The artwork is one of the most renowned by Jackson Pollock, "Alchemy". Do note that the first time-bin highlights visitors that observed the artwork for less than 5 seconds (14%). Mean time of permanence is 48s (45s without audio guide, 115s with audio guide), standard deviation is 46s, and most of the visitors (73%) leave within one minute.

Manual measurements can also be made on specific aspects of the museum. It is the case, *e.g.*, of measuring the time of permanence in front of single artworks. A sample measurement of this kind is depicted in Figure 4.11 and allows the gathering of highly detailed insights on specific artworks. In particular, it is strictly related to the *holding power* of each piece of art and can prompt interesting suggestions on possible rearrangements of a collection (see also [112]). Furthermore, it can be correlated with other information, like the usage of an audio guide, the visitors' age, sex, and ethnic group to perform in-depth studies on the single artwork (which is out of the scope of this work).

Number	Room nickname	Antenna
r_1	Paolina	a_2
r_2	David	a_4
r_3	Apollo e Dafne	a_5
r_4	Ratto di Proserpina	a_1, a_{10}
r_5	Portico	a_8, a_9
r_6	Enea e Anchise	a_{11}
r_7	Satiro su delfino	a_7
r_8	Caravaggio	a_6
r_9	Pinacoteque	a_3, a_{12}, a_{13}, a_{14}

Table 4.2: Match amongst the $R = 9$ rooms (r_x) and the $A = 14$ RPi antennas (a_{xy}) in the Galleria Borghese.

4.4.3 Analysis of the case studies and corresponding datasets

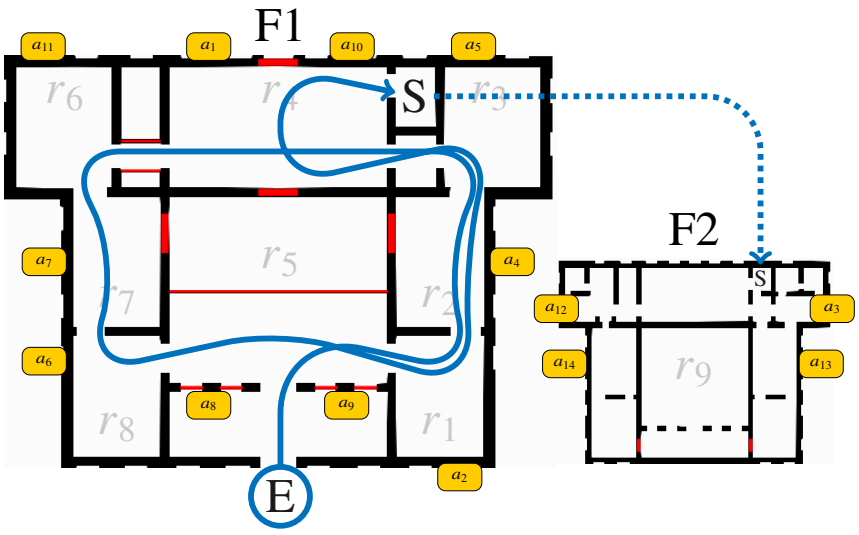
In the following, we describe the tracking system details of our two case studies along with the two corresponding datasets we were able to put together from our measurements.

Galleria Borghese

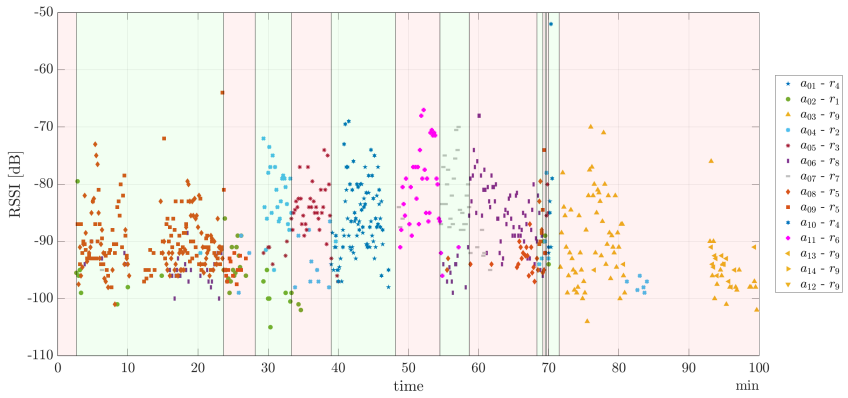
The Lagrangian data gathered in the Galleria Borghese case study come from a measurement campaign that lasted between June and August 2019. The central SQL server received $1,308,617$ records corresponding to 900 visitor trajectories surveyed during 13 2h-long visit slots. The percentage of tracked visitors w.r.t. the total number was about 1:5. As it usually happens, the vast majority of visitors came in groups (family, friends, guided tours, etc.). In this case, apart from a few exceptions, we tracked only one member of each group, thus losing the ability to detect the interactions within social groups.

Due to the architectural constraints, we were able to monitor the 21 exhibition areas of the museum with $A = 14$ antennas only, hence we decided to group the physical rooms into $R = 9$ areas (virtual rooms). Table 4.2 reports the antenna-room assignments, while Figure 4.12**(a)** display the antennas position on the museum map and the virtual room definition, along with a sample ground-truth trajectory associated with data from Figure 4.12**(b)**, here conveniently represented with room transition highlighted and with colours matching the virtual room of the reading.

According to the disposition of antennas within the different exhibition areas, we chose to reconstruct the two visiting floors with different resolutions, building trajectories at room-scale on the first floor while having a single comprehensive room on the second one. We assigned a single colour to the so-formed $R = 9$ rooms. We include an entrance/exit artificial room connected to the three entry rooms of the museum (r_4, r_5 and r_9) representing the ticket office (where we also provided

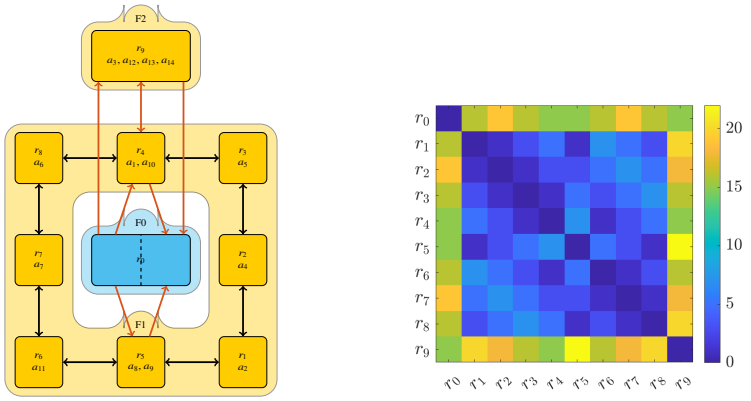


(a)



(b)

Fig. 4.12: (a) The Galleria Borghese floor plan. We report the position of antennas (a_{xy}) and virtual rooms (r_x). Red lines represent closed doors. We include in blue a sketch of the trajectory whose RSSI measurements are in panel **(b)**. The trajectory begins on the first floor (F1) and follows a counter-clockwise path before reaching the second floor (F2). **(b)** RSSI measurements for a real visitor trajectory (sketched in panel **(a)**). The readings from the $A = 14$ antennas are coloured depending on the virtual room (thus, measurements taken by any of the antennas on the second floor come with the same colour, yellow triangles). Room transitions are represented by the alternating background colour.



(a) **(b)**

Fig. 4.13: **(a)** Graph representation of the Galleria Borghese (cf. 4.12**(a)**) with the corresponding edge-clustering enforced by the orange connections (↔). Do note the addition of a fictitious room r_0 to serve as entrance/exit. **(b)** The distance matrix obtained weighting '2' the door connections ($w(\leftrightarrow) = 2$), '15' the staircases connection ($w(\leftrightarrow) = 15$) and '0' the different type of exhibitions ($\beta = 0$). A discount factor of '$^1/_2$' is applied for the first door transition (cf. Footnote 4).

visitors with beacons), assigning to it a different colour. The point of interest for the clustering phase consists, however, of the kinds of connections here represented in black (doors) and orange (stairs). The coloured-graph representation of the museum with its edge-colouring contraction, along with the corresponding distance matrix is also reported in Figure 4.13.

The Eulerian data gathered come from multiple measurement campaigns that occurred during 2019, 2020 and 2021 and mainly serve here as benchmarks for the construction of the methodologies introduced in the rest of this chapter. In particular, data for the occupancy of the whole museum, reported in Figure 4.14, are worth of mention. It clearly highlights two of the most important criticalities that first motivated the study here proposed: (i) the museum fully empties between the time slot due to exit/enter procedures and (ii) the museum mean occupancy on the second hour of each turn is below 50%, hence being the occupancy trend highly sub-optimal. Furthermore, they can be used to study entrance and exit rates from the museum.

Along with them, we gathered single room occupancy, and we measured different aspects regarding the visitors' choices: the chosen entrance and the first choice between going left or right upon entrance are two examples. The second measurement, in particular, is correlated with the "clockwisety" of the visit, *i.e.* if the main floor is mainly visited in clockwise or anti-clockwise order, being circular in shape.

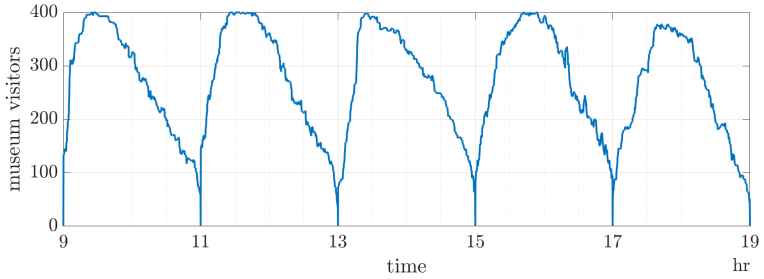


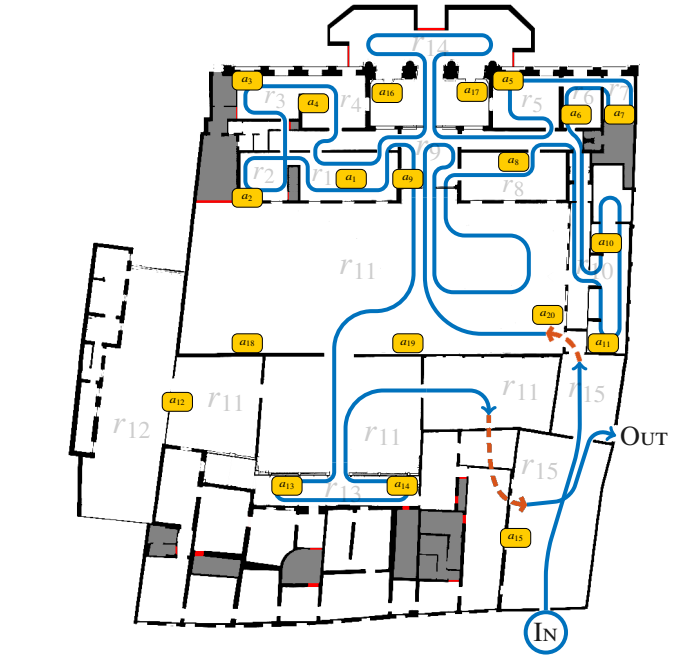
Fig. 4.14: Example of occupancy of the whole museum in the Galleria Borghese case study. It is clear that the decision of having non-overlapping time slots heavily penalises the fruition of the museum since the museum is overcrowded in the first hour and poorly populated during the last half-hour.

Peggy Guggenheim Collection

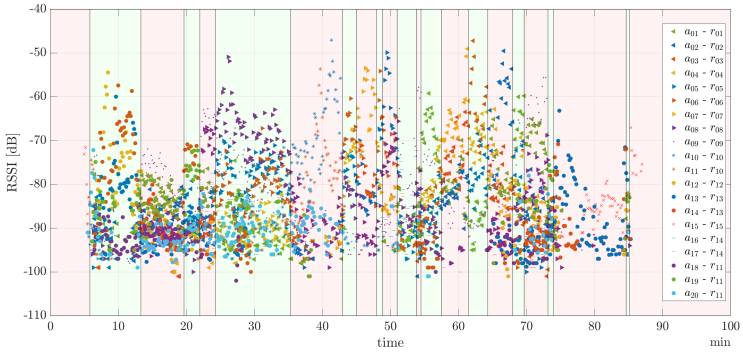
The Lagrangian data gathered in the Peggy Guggenheim Collection case study come from a measurement campaign that lasted between December 2022 and January 2023. The central SQL server received $1,377,293$ records corresponding to 524 visitors' trajectories, ranging in time from 10 up to 310 minutes (5 hours and 10 minutes). Conversely to the Galleria Borghese measurements, the management of beacons delivery and pick-up operations was entrusted to museum interns, hence to simplify the procedures, beacons were only delivered to visitors who left items in the wardrobe.

At the time of Lagrangian measurements, no temporary exhibitions were held, hence we were able to monitor the 18 exhibition areas of the museum, along with the terrace, the cafeteria, the 4 areas of the sculpture garden, the veranda, and the ticket office with $A = 20$ antennas. We deployed mainly one antenna per room in the permanent exhibition while we were able to place only two antennas in the Schulhof collection. Being mainly outdoor areas, we were able to deploy a limited number of antennas in the garden (3), the ticket office area (1), the terrace (2), and the bar (1). Table 4.3 reports the antenna-room assignments, while Figure 4.15**(a)** display the antennas position on the museum map and the virtual room definition, along with a sample ground-truth trajectory associated with data from Figure 4.15**(b)**.

We decided to reconstruct room-wise the permanent exhibition area but PG-9 and the corridors that, despite holding artworks, were too narrow to correctly deploy antennas. We provided the so-formed 9 virtual rooms with a single colour and we linked them with two kinds of connections to represent the following conceptual constraint: visitors typically visit one side of the permanent exhibition before the other (this was originally suggested by the curator and further confirmed by data, see later Section 4.6.3). Due to the limited number of antennas, we decided to reconstruct the rest of the areas by grouping them up in a single virtual room each, each one with a different colour, as suggested in Figure 4.4. Connections between exhibition areas and the garden are classified by a third colour representing the



(a)



(b)

Fig. 4.15: (a) the Peggy Guggenheim Collection map. We report the position of antennas (a_{xy}) and virtual rooms (r_{xy}). Red lines represent closed doors and greyed areas represent inaccessible zones. We include in blue a sketch of the trajectory whose RSSI measurements are in panel **(b)**. The trajectory starts the visit from Calder Hall (r_{09}) by crossing the sculpture garden (r_{11}). It visit the left wing first and the right wing of the permanent exhibition after. Later, it goes to the Schulhof collection (r_{10}) before moving to the garden and the veranda (r_{12}) and, finally, leaving the museum. **(b)** RSSI measurements for a real visitor trajectory (sketched in panel **(a)**). The readings from the $A = 19$ antennas are shaped depending on the virtual room. Room transitions are represented by the alternating background colour.

Number	Room name	Antenna
r_{01}	PG-1	a_{01}
r_{02}	PG-2	a_{02}
r_{03}	PG-3	a_{03}
r_{04}	PG-4	a_{04}
r_{05}	PG-5	a_{05}
r_{06}	PG-6	a_{06}
r_{07}	PG-7	a_{07}
r_{08}	PG-8	a_{08}
r_{09}	Calder Hall	a_{09}
r_{10}	Schulhof Collection	a_{10}, a_{11}
r_{11}	Sculpture garden	a_{18}, a_{19}, a_{20}
r_{12}	Cafeteria	a_{12}
r_{13}	Veranda	a_{13}, a_{14}
r_{14}	Terrace	a_{16}, a_{17}
r_{15}	Ticket office	a_{15}

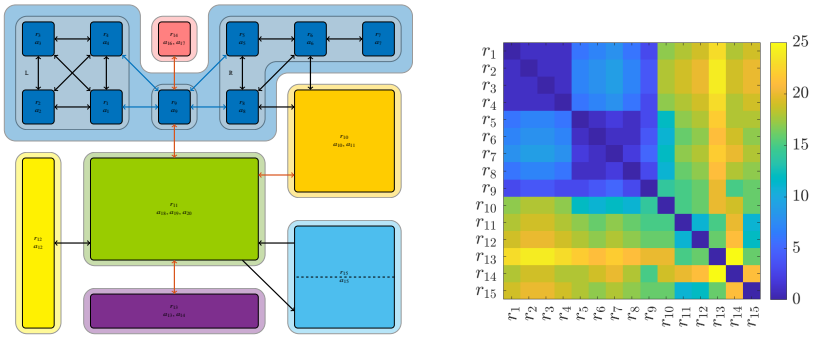
Table 4.3: Match amongst the $R = 15$ rooms (r_{xy}) and the $A = 20$ RPi antennas (a_{xy}) in the Peggy Guggenheim Collection.

transition from indoor to outdoor. The reconstruction yields a total of $R = 15$ virtual rooms (in 7 colours) connected by 22 edges (in 3 colours). The coloured-graph representation of the museum with its multi-step multi-colouring contraction, along with the corresponding distance matrix is reported in Figure 4.16. The multi-step contraction highlights a finer representation with 9 areas and a coarser representation with 7 areas, being the difference in the reconstruction of the permanent exhibition that is divided into left-, centre-, and right wing (L, r_9, and R).

The Eulerian data gathered come from multiple measurement campaigns that occurred during June, July, and December 2022, and January, March, and May 2023 and mainly serve as a source of information for the final discussion in Section 4.9.

Conversely to the Lagrangian dataset, some of the Eulerian measurements were gathered while the temporary exhibition "Surrealism and Magic: Enchanted Modernity" was held. For this reason, we were able to track the number of visitors inside the two main exhibition areas during June and July 2022. The data, reported in Figure 4.17, mainly show that the first opening hours are the least crowded and a pike in presence can be located before lunch. Furthermore, the presence in the temporary exhibition is constantly lower than the correspondent presence in the main exhibition. This latter fact can be attributed to three main reasons: (i) visitors approach the temporary exhibition after the permanent one and could be more fatigued, (ii) the building structure of the temporary exhibition is less appealing compared to the main exhibition (being hold in the original Peggy Guggenheim's home), and (iii) the artworks held in permanent exhibition are of higher artistic and historical value.

Like in the Galleria Borghese case study, we gathered single-room occupancy to be used as benchmarks and we measured the ratio of first choice between going left wing L or right wing R upon entrance in Calder Hall (r_9). Of particular interest, is



(a) (b)

Fig. 4.16: **(a)** Graph representation of the Peggy Guggenheim Collection (cf. 4.15**(a)**) with the corresponding multi-colouring contraction: edge-colouring enforced by the blue connections ($\leftrightarrow$) for wings and vertex-colouring for thematic areas. Do note that here no additional room for entrance/exit is needed since its role is already handled by the ticket office (r_{15}). **(b)** The distance matrix obtained weighting '1' the door connections ($w(\leftrightarrow) = 1$), '3' the wing connection ($w(\leftrightarrow) = 3$), '5' the inside-outside transition ($w(\leftrightarrow) = 5$) and '10' the different type of exhibitions ($\beta = 10$). A discount factor of '1' is applied for the first door transition (cf. Footnote 4).

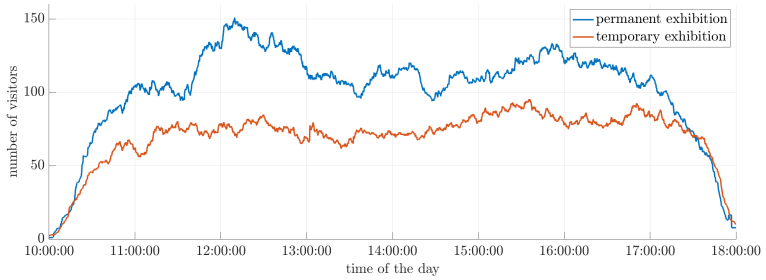


Fig. 4.17: Mean museum occupancy in the Peggy Guggenheim Collection where visitors are distributed amongst temporary and permanent exhibitions.

the fact that visitors are evenly split between the two choices L and R, hence later causing a flux cross once they get back to Calder Hall (r_9).

Finally, we also measured the time of permanence in front of a number of artworks. A simple yet effective result of this analysis is later discussed in Section 4.9.

4.5 Trajectory reconstruction and filtering

We use the raw data collected by the tracking systems to reconstruct trajectories $\mathrm{t} = (\mathrm{t}_0, \mathrm{t}_1, \ldots, \mathrm{t}_{T-1})$, which represent the sequence of visited rooms and the time of permanence in each room. These reconstructed trajectories are then *compared*

with ground-truth data, which can be obtained either through field observations or manual extraction from the data.

Naive methods for comparing a reconstructed trajectory with its ground truth include binary accuracy, which can be used as an initial indicator. Given two trajectories $\mathfrak{s}$ and $\mathfrak{t}$ spanning T discrete time bins (eventually padded to match in length), we recall the binary accuracy as follows:

$$\mathrm{Acc}(\mathfrak{s}, \mathfrak{t}) = \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{1}_{\mathfrak{s}_t = \mathfrak{t}_t} \ . \tag{4.7}$$

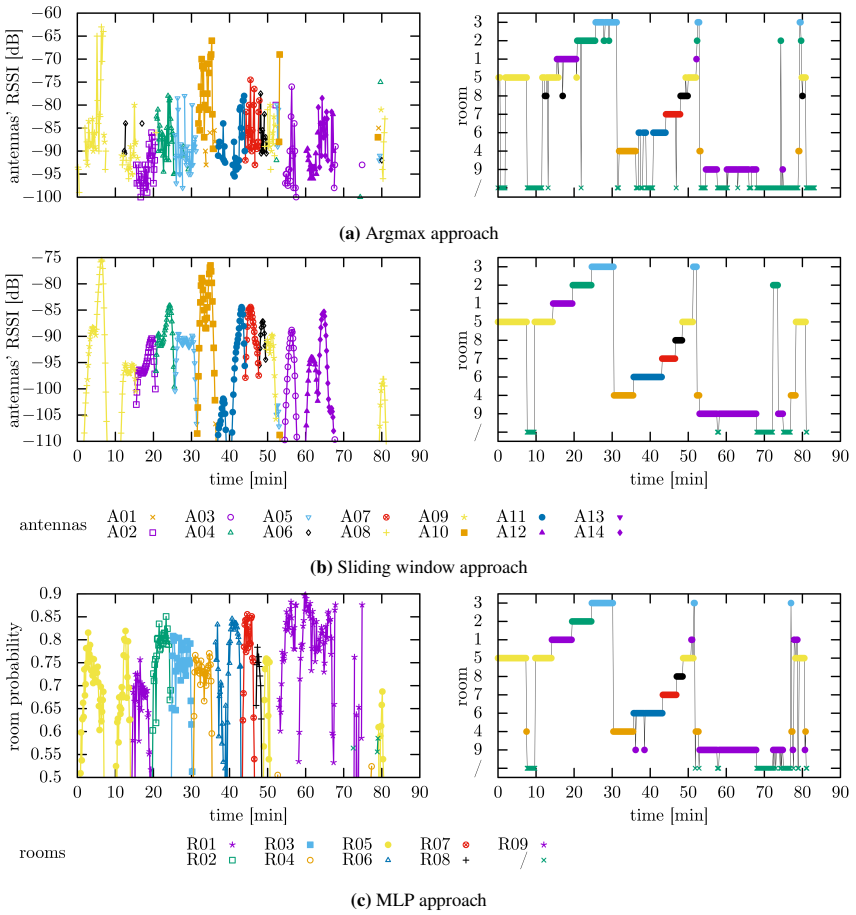
However, the coloured graph representation and, in particular, the metric (4.5), offer the opportunity to develop a more comprehensive set of indicators. We can adopt the room-level metric $\mathcal{D}$ as is to evaluate single room displacements and we can extend it to a Wasserstein-inspired trajectory-level metric $\mathcal{W}$ by summing the component-wise distances between instantaneous locations, that is:

$$\mathcal{W}(\mathfrak{s}, \mathfrak{t}) = \sum_{t=0}^{T-1} \mathcal{D}(\mathfrak{s}_t, \mathfrak{t}_t) \ . \tag{4.8}$$

The collected RSSI data does not have uniform temporal sampling, therefore, as a first step, we apply a re-sampling procedure to the signal with a fixed time step of $\Delta t = 10\,\mathrm{s}$ (cf. Figure 4.8). The choice of Δt involves a trade-off between signal granularity and desired resolution; in particular, using $\Delta t = 10\,\mathrm{s}$ results in six samples per minute, which is sufficiently small compared to the typical duration of room visits and traversal times, while also allowing redundant antenna advertisement signals (RPi transmission frequency in our model is 5 seconds). We employed a threshold of -120 dB for antennas that fail to detect the beacon. By applying this procedure to a single beacon, we obtain an $A \times T$ matrix $\mathcal{R}$ (where T is the visit duration divided by Δt). In other words, for a given beacon, the element $\mathcal{R}_{a,t}$ represents the mean RSSI of the signal received by the a-th antenna in the t-th time bin, or -120 if the a-th antenna never detects the beacon during that time bin.

The most straightforward way to reconstruct visitor trajectories is to compute the argmax of the RSSI history for each beacon: at each time bin, we retain the antenna that receives the highest RSSI value. The current location of the visitor is then determined based on the room associated with the antenna. A sample result of this procedure is shown in Figure 4.18**(a)**. Unfortunately, when a visitor is equidistant from two or more antennas, the maximal RSSI value oscillates rapidly amongst the antennas. In signal terms, the visitor appears to perform extremely rapid and unrealistic room changes.

Therefore, we present three data refinement methods that progressively increase in complexity and reliability. The first method adopts a sliding window approach, the second method is based on a simple multi-layer perceptron (MLP) model, and the third method relies on a cascaded localiser constructed using the contracted-graph representation of the museum.



(a) Argmax approach

(b) Sliding window approach

(c) MLP approach

Fig. 4.18: A sample beacon RSSI elaborated by **(a)** argmax, **(b)** sliding window, and **(c)** multi-layer perceptron approaches ($\delta = 6$). The left column reports the max of RSSI for the argmax and sliding window approaches (antennas located in the same room are labelled with the same colour but different markers), and the maximum amongst the rooms probabilities for the MLP approach. The right column reports the corresponding reconstructed trajectories as sequences of rooms. Not-detected statuses are marked by green crosses (×).

4.5.1 Sliding window approach

The first method aims to reduce noise in the RSSI data by applying a low-pass filter, which is implemented as a *weighted moving average*, and a *normalisation*. RSSIs gathered close in time should have close values; besides, the closer the bins, the higher the correlation.

To achieve this, we convolve the RSSI signals obtained from each antenna (*i.e.*, the rows $\mathcal{R}_{\cdot,t}$) of the RSSI matrix with a (symmetric triangular) kernel of size $2\delta + 1$

and weights $w_0, w_1, \ldots, w_{2\delta}$. Formally, this convolution generates a new matrix $\tilde{\mathcal{R}}$ defined as follows:

$$\tilde{\mathcal{R}}_{a,t} = \sum_{d=t-\delta}^{t+\delta} \mathcal{R}_{a,d} \cdot w_{\delta-t+d}, \quad 0 \leq a < A, \quad \delta \leq t < T - \delta \,. \tag{4.9}$$

Next, we apply a z-score normalisation across the signals acquired by the different antennas to make them comparable. This results in a third matrix $\bar{\mathcal{R}}$ given by:

$$\bar{\mathcal{R}}_{a,t} = \frac{\tilde{\mathcal{R}}_{a,t} - \mu_t}{\sigma_t}, \qquad 0 \leq a < A, \quad \delta \leq t < T - \delta \,, \tag{4.10}$$

where μ_t and σ_t are, respectively, the mean and standard deviation of $\bar{\mathcal{R}}$ at each time bin (*i.e.*, column-wise). In other words, $\mu_t = \mu(\bar{\mathcal{R}}_{\cdot,t})$ and $\sigma_t = \sigma(\bar{\mathcal{R}}_{\cdot,t})$.

Figure 4.18**(b)** illustrates a sample outcome of this procedure.

4.5.2 Multi-layer perceptron approach

The second method lays its foundation in the fact that a trained human is typically able to reconstruct the trajectory by looking at the plotted dataset (cf. Figure 4.8), hence making neural networks suitable for this purpose. Both as an improvement in performances compared to the sliding window method and as a baseline for the third methodology, we propose a trajectory reconstruction approach based on a vanilla MLP (see 2.3.2).

At any time bin t, we cast the localisation of a visitor in one amongst the R rooms as a classification problem, hence requiring the MLP to process the $\mathcal{R}$ matrix (in time windows) and to return the probability vector whose r-th component is the probability that the visitor is located in the room r. For the sake of simplicity (and the lowest possible data requirements for training), we built a single hidden layer (hence a 3-layer MLP) and we adopt the sigmoid as activation function.

At each time t, we considered a symmetrical time window of $(2\delta + 1)$, hence generating $(2\delta + 1)A$ input features obtained by restricting the matrix $\mathcal{R}$ to the δ columns before and after t (analogous notation to Section 4.5.1), *i.e.* the column block $\mathcal{R}_{\cdot,t-\delta\cdots t+\delta}$. The single hidden layer is to be hyper-tuned in the number of neurons; however, we find it suitable to consider $4 \times A$ neurons, as a trade-off between the input and the output layer sizes. The network output layer is made of R neurons that we interpret – through a softmax normalisation – as the instantaneous probability of being in the r-th room.

According to the validation performed on our datasets, almost always the MLP outputs a classification with an overall majority (probability > 0.5, see Figure 4.18**(c)**). However, it can happen – particularly during a room transition – that no class reaches such a majority. In such a case, we proceed by removing probability values smaller than a fixed threshold $\epsilon = 0.15$ and those yielding unfeasible tran-

sitions according to our graph representation. After re-normalising the remaining, in the unlikely event that still no room discloses an overall majority, we assume the visitor standing still.

Comparing with the sliding window approach

Looking at Figure 4.18**(b)** and Figure 4.18**(c)** we observe two interesting features: the sliding window approach often prevents spikes from arising (see minutes $\sim 35, 53$), yielding cleaner trajectories that make it easier to enumerate room transitions. On the other hand, it tends to smooth trajectories excessively, ignoring fast transitions (see minute ~ 52).

The first phenomenon is simply explained by the pass-filter nature of the sliding window approach: it typically creates smooth signal curves that hardly intersect in more than one point per room transition.

The latter phenomenon, conversely, can be verified if we consider the bin-by-bin accuracy and it is explained by the major flexibility of the MLP that learns over the training set.

The accuracy achieved over the Galleria Borghese test set (20% of the entire dataset) by the MLP is in fact 0.858, compared to an accuracy of 0.734 obtained by the sliding windows approach. It is doubtless, however, that both approaches overcome the results obtained via the argmax approach, which has an accuracy of 0.547.

4.5.3 Cascaded trajectory reconstruction based on colour-clustering

The third and last method mainly finds its advantage in the expert-knowledge information injected in the museum graph and its contraction (cf. Section 4.3). The idea is to use the different clustering of museum rooms we obtained from the total-coloured graph (cf. Figure 4.6) to build a series of cascaded classifiers that improve room-level localisation accuracy over the other methods.

Our clustering creates a partition of V in aggregates of rooms $K_1, K_2, \ldots$ (clusters) that are, by construction, typically explored by visitors one at a time. Thus, our localisation approach first identifies the visitor in one amongst the different room clusters, say K_i, and, only afterwards, in a specific room within K_i. From an operational point of view, this corresponds to building one "large-scale" localiser that classifies at cluster-level $K_1, K_2, \ldots$, and multiple "fine-scale" localisers, each operating within a single cluster, and returning a specific room. Each localiser, at either large- or small-scale, operates on all or a subset of $A' \leq A$ RSSI signals, possibly down-sampled in time. As RSSI are highly fluctuating, analysing data from a time window of, say, $T' \leq T$ samples, instead of the single measurements is mandatory. We refer to the time window as symmetric if data from both the future and the past is used, and asymmetric otherwise (necessary for online trajectory reconstruction). In

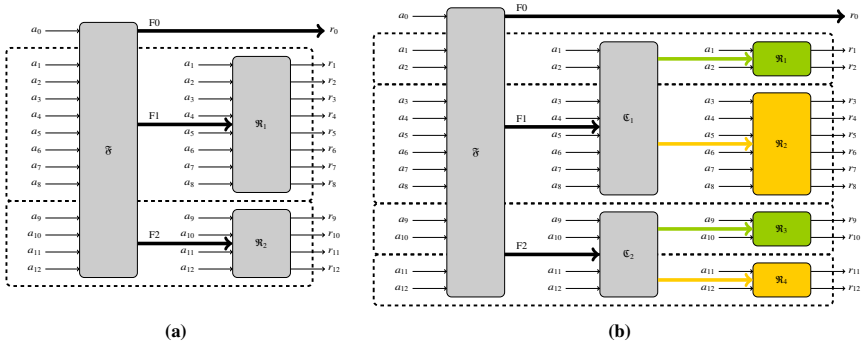


Fig. 4.19: Two possible designs for a cascaded selector built upon the museum in Figure 4.5**(a)**. A direct match between antennas and room is assumed (*i.e.* a_i is deployed in room r_i, with $0 \leq i \leq 12$). **(a)** A two-layer cascaded selector as enforced by the clustering in Figure 4.6**(b)**. **(b)** A two-layer cascaded selector as enforced by the clustering in Figure 4.6**(c)**.

the case of the *large-scale* classifier, considering a time window including a down-sampled version of the RSSI signal yields better performance since it limits the input dimension of the classifier and prevented noisy inputs.

It is clear that the task for the coarse-grain large-scale localiser is simpler *w.r.t.* the task of directly classifying the beacon in the correct room, hence we expect it to achieve a better accuracy (later cf. Table 4.4). It follows that, since locating the beacon in the correct area of the museum provides an upper bound for the misplacing, such an approach guarantees an improvement in the overall misplacement measured by $\mathcal{W}$.

Considering the example in Figure 4.6**(b)**, we would employ three localisers: a floor selector $\mathfrak{F}$ fed with the signal from all the antennas that outputs one amongst the three areas (F0, F1 or F2) of the museum and two room selectors $\mathfrak{R}_1$ and $\mathfrak{R}_2$ (to be mutually employed depending on the output of $\mathfrak{F}$) that use the signals from the receivers on their respective floor to output the most probable room within that floor. Do note that area F0 has a single room associated with it, hence making it pointless to build a selector $\mathfrak{R}_0$.

The described method naturally extends to consider multiple layers of localisers, coarser to finer, simply considering multiple progressive contractions; it is the case, *e.g.* of what happens if we consider contracting progressively over multiple kinds of edges or over edges and vertices (multi-contraction). Consider, *e.g.*, the example in Figure 4.6**(c)**, we can build a three steps mechanism based on seven localisers: the initial structure would be the same as before, with a floor localiser $\mathfrak{F}$, followed by two colour localisers $\mathfrak{C}_1$ and $\mathfrak{C}_2$, each one trained on a different floor; four fine-grained room localisers $\{\mathfrak{R}_i\}_{i=1}^{4}$ close the structure.

A pictorial representation of the described selectors can be found in Figure 4.19.

The choices for each individual localiser are also critical for the success of the method. Depending on the amount of data available one could opt for heuristics or data-driven approaches.

- Heuristic algorithms heavily depend on the museum topology. Convolutional kernel approaches (ConvKer) yield a reference baseline, so we used our sliding windows approach as a benchmark.
- Many options are available for classification algorithms trained with ground-truth data [40]. We specifically consider three of them, widely used in the literature: (i) the MLP, built as in the Section 4.5.2 as an example of FNN, (ii) the LSTM as an example of RNN, better suited than regular ANN, and (iii) the random forest (RF), capable of accomplish classification tasks efficiently.

MLP and LSTM are described respectively in Section 2.3.2 and 2.3.3. Conversely, RFs are predictors consisting of an ensemble of randomised decision trees [91], *i.e.* weaker tree classifiers where each internal node represents a query over the data features and each leaf is a classification label. The RF classifies following the majority vote over the tree ensemble, following the concept that weaker classifiers trained on different subsets of data learn various aspects from the features.

4.5.4 Handling not-detected beacons

Due to (small) areas uncovered by antennas or because of random signal losses, it may happen that a beacon remains not detected. This is also what (correctly) happens before and after a visit or when the visitor leaves the exhibition areas during the visit, for example to reach the toilet.

Before performing statistical analyses, we amend for not-detected statuses whenever this can be done unambiguously. Although we notice that such a process might require mainly museum-specific solutions, we report two corrections which we deem of general interest.

1. If the *blind period* is less than a few minutes (typically 3 minutes is suitable, with some exceptions, *e.g.* 10 minutes for the Pinacoteque of Galleria Borghese due to the low antennas presence), and the visitor is detected in the same room before and after the blind period, the visitor is associated with that room for the whole period.
2. If the blind period is less than a few time bins (typically 3, equals to 30 seconds with $\Delta t = 10\,\mathrm{s}$), and the visitor is detected in two different rooms before and after the blind period, the visitor is supposed to be in one between the two rooms (at random).

Whenever the overall not-detected status exceeds 25 minutes, we decided to trash such a trajectory, hence removing it from the dataset.

4.5.5 Trajectory reconstruction results in case studies

In this section, we discuss the performance of our cascaded methodology on the basis of Lagrangian experimental data gathered in our measurement campaigns. We design the cascaded localisers relying on the clustering built in Section 4.4.3 and we present the performances of the methods by means of four indicators based on the metric enforced by the graph.

The metrics considered are:

Acc The vanilla accuracy, given by the fraction of correct predictions over the total number of samples.

$\overline{\mathcal{D}}$ Mean sample displacement, given by the average distance $\mathcal{D}$ between the sample, *instantaneous*, predictions and the corresponding ground truth.

$\overline{\mathcal{D}}^*$ The mean sample displacement error, given by the average distance $\mathcal{D}$ restricted to wrongly labelled samples, and the corresponding ground truth.

$\overline{\mathcal{W}}$ The mean trajectory displacement, given by the average distance $\mathcal{W}$ between the reconstructed trajectory and the corresponding ground truth.

Building upon the emerging clustering from Figure 4.13**(a)**, we develop a two-layer cascaded localiser built of two predictors: (i) a floor predictor $\mathfrak{F}$, whose inputs are the data from all the A antennas; (ii) a sculpture floor (F1) room predictor $\mathfrak{R}$, whose input are the data from all the $A' \subset A$ antennas displaced in the rooms from F1. The structure of the localiser is reported in Figure 4.20**(a)**.

Conversely, building upon the emerging clustering from Figure 4.16**(a)**, we develop a three-layer cascaded localiser built of four predictors: (i) a zone predictor $\mathfrak{Z}$, whose inputs are the data from all the A antennas and the output is one of the seven colour clusters; (ii) a wing predictor $\mathfrak{W}$, whose input are the data from all the $A' \subset A$ antennas displaced in the rooms from the main exhibition area and the terrace and the output is either the left, the right or the central area of the collection; (iii-iv) two room localisers $\mathfrak{R}_{\text{L}}$ and $\mathfrak{R}_{\text{R}}$ working respectively on the two wings of the museum. The structure of the localiser is reported in Figure 4.20**(b)**.

As a sample benchmark, in Table 4.4, we compare the localisation accuracy we achieve employing different methods for building the single localisers for the Galleria Borghese cascaded selector, considering both online and offline perspectives (*i.e.* involving asymmetric resp. symmetric time windows). Additionally, we report the results obtained via a single classifier (cf. Section 4.5.2) as a benchmark.

The metrics are elaborated on a test set, $\mathcal{T}$, disjoint from the training set, consisting of about 20% of the labelled data. We employed the matrix from Figure 4.13**(b)** as room-distance matrix $\mathcal{D}$.

The proposed method proves not only to increase the accuracy but also to commit much smaller errors when a room is misclassified. In particular, when adopting $\mathfrak{F}$ = RF and $\mathfrak{R}$ = LSTM, the mean error committed by the method is $\overline{\mathcal{D}} \approx 0.08$ compared to $\overline{\mathcal{D}} \approx 0.57$ by a single LSTM localiser. This corresponds to an effective displacement restricted to misclassifications of $\overline{\mathcal{D}^*} \approx 2.5$, where 1 means the visitor is positioned in an adjacent room and 3 means the visitor is located 2 rooms away

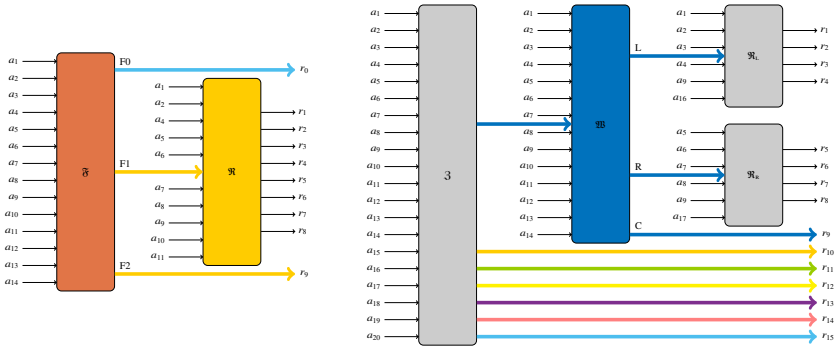


Fig. 4.20: Structure of the cascaded selectors based on the clustering associated with **(a)** The Galleria Borghese (cf. Figure 4.13(**a**)) and **(b)** the Peggy Guggenheim Collection (cf. Figure 4.16(**a**)). Do note that localisers $\mathfrak{W}$, $\mathfrak{R}_{\mathrm{L}}$, and $\mathfrak{R}_{\mathrm{R}}$ also take as input the signal from antennas a_9, a_{16}, and a_{17} due to their position inside the museum (cf. Figure 4.15(**a**)).

Symmetric time window $[-6, +6]$

Methodology	Accuracy	$\overline{\mathcal{D}}$	$\overline{\mathcal{D}^*}$	$\overline{\mathcal{W}}$
	Single Localiser			
ArgMax	0.6562	4.3755	12.727	3150.3
ConvKer	0.7645	3.1693	13.458	2281.9
MLP	0.8679	1.4402	10.903	1037.0
LSTM	0.9267	0.5761	7.8605	414.84
RF	0.8900	1.2150	11.046	874.84
	Floor Localiser			
ArgMax	0.8673	1.9905	15	–
ConvKer	0.9005	1.4925	15	–
MLP	0.9045	1.4325	15	–
LSTM	0.9650	0.5250	15	–
RF	0.9386	0.9210	15	–
	Floor Localiser with downsampling (Δt = 5min)			
ArgMax	0.9148	1.2780	15	–
ConvKer	0.8201	2.6984	15	–
MLP	0.9949	0.0764	15	–
LSTM	0.9940	0.0900	15	–
RF	0.9966	0.0509	15	–
	Sculpture floor Localiser			
ArgMax	0.8111	0.3314	1.7546	–
ConvKer	0.8974	0.1327	1.2938	–
MLP	0.9493	0.0593	1.1714	–
LSTM	0.9507	0.0521	1.0588	–
RF	0.9435	0.0709	1.2564	–
	Multiple Localiser (Floor + Sculpture)			
RF – LSTM	**0.9677**	**0.0816**	2.5267	**58.760**

Asymmetric time window $[-6, 0]$

Methodology	Accuracy	$\overline{\mathcal{D}}$	$\overline{\mathcal{D}^*}$	$\overline{\mathcal{W}}$
	Single Localiser			
ArgMax	0.6562	4.3755	12.727	3150.3
ConvKer	0.7105	3.4537	11.930	2486.6
MLP	0.8525	1.5483	10.497	1114.7
LSTM	0.8721	1.1459	8.9600	825.10
RF	0.8662	1.3524	10.108	973.76
	Floor Localiser			
ArgMax	0.8673	1.9905	15	–
ConvKer	0.8986	1.5210	15	–
MLP	0.9011	0.9480	15	–
LSTM	0.9480	0.7800	15	–
RF	0.9258	1.1130	15	–
	Floor Localiser with downsampling (Δt = 5min)			
ArgMax	0.9148	1.2780	15	–
ConvKer	0.8283	2.5754	15	–
MLP	0.9932	0.1020	15	–
LSTM	0.9966	0.0509	15	–
RF	0.9957	0.0644	15	–
	Sculpture floor Localiser			
ArgMax	0.8111	0.3314	1.7546	–
ConvKer	0.8258	0.2165	1.2429	–
MLP	0.9043	0.1072	1.1212	–
LSTM	0.9058	0.1260	1.3385	–
RF	0.8928	0.1390	1.2973	–
	Multiple Localiser (Floor + Sculpture)			
LSTM – LSTM	0.9413	0.1250	2.1311	90.068
LSTM – RF	0.9337	0.1326	**2.0008**	95.510

Table 4.4: Localisation performances achieved by employing a combination of selectors on experimental data collected at the Galleria Borghese. Best results are highlighted in yellow. Performances with single localisers are reported as a reference. All the localisers have been trained on a desktop computer in less than five minutes, being the RF the fastest in training ($^1/_{30}$ of MLP training) and $LSTM$ the slowest (three times the MLP training). The ConvKer employs the sliding window approach (see Section 4.5.1).

from the ground truth. We note that this major increase is due to the "large-scale" classifier generally labelling correctly (accuracy $> 99.5\%$) the floor a visitor is at. $\mathfrak{R}$ = LSTM proves to be the best sculpture floor selector, and, overall, using a symmetric time window provides higher results than the online approach. In the case of the floor selector $\mathfrak{F}$, online and offline approaches based on RF and LSTM delivered comparable results. This aspect is probably due to the fact that the floor classification problem is more regular and simpler than the room classification.

4.6 Trajectory analysis and clustering

In this section, we first present a few basic statistics for analysing the Lagrangian dataset that are useful both *per se* and for the creation of the digital twin later. We employ our case-study datasets to explain their usage and impact and we show how the metric enforced by the total coloured graph can be used to perform preliminary analysis on the trajectory dataset. Later, we introduce the core of this section, being the unsupervised clustering of the trajectories. Closes the section a discussion over case-study dataset clusterisation.

4.6.1 Basic statistics

Let us consider a dataset of N trajectories $\mathfrak{t}^1, \dots, \mathfrak{t}^N$ of maximum length T. We consider three basic illustrative statistics that we also employ in Section 4.7 to calibrate our digital twin:

Time of Permanence We denote by $\text{ToP}(v, r)$ the total time spent by trajectory $\mathfrak{t}^v$ in room $r \in \{1, \dots, R\}$ during its visit, with $v \in \{1, \dots N\}$. It is important not to confuse ToP with the Eulerian measurements: in fact, even if ToP is a room-wise measurement, it still is dependent on the trajectory $\mathfrak{t}^v$.

Returning visitors We denote by $\text{Ret}(v, r)$ the number of times the trajectory $\mathfrak{t}^v$ *stopped* by room r. Being a Lagrangian measurement, Ret provides insights on museum navigability since, in principle, reducing overcrowding requires avoiding flow intersections and, hence, reducing backtracking.

People per Room We denote by $\text{PpR}(r, t)$ the number of visitors in room $r \in \{1, \dots, R\}$ during the time bin $t \in \{1, \dots, T\}$. It is important to note that *this* measure is the analogous to the Eulerian measurement of a room. The idea of extracting it from the Lagrangian dataset is that the Eulerian dataset can be here used to validate trajectory reconstruction and sampling. Furthermore, being able to evaluate it from the Lagrangian dataset will unleash more potential from the trajectory-wise digital twin we introduce later in Section 4.7

Time of Permanence

Given a trajectory t^v of a visitor $v \in \{1, \ldots, N\}$ it is straightforward to evaluate the corresponding time of permanence in room $r \equiv r_i, i \in \{1, \ldots, R\}$, being

$$\mathrm{ToP}(v, r) = \sum_{t=1}^{T} \mathbb{1}_{\mathrm{t}_t^v = r} \ . \tag{4.11}$$

Given the set $\mathrm{ToP}(\cdot, r) = \{\mathrm{ToP}(v, r)\}_{v=1}^{N}$, it is worthwhile studying an approximation of its empirical distribution by means of some statistical distribution. To this scope, many functions can be used to reproduce ToP including, *e.g.*, Gaussian, Poisson, or Gamma distributions; however a Weibull distribution with parameters $\lambda = \lambda(r)$ and $k = k(r)$ depending on room r proved to fit well[6] in most of the cases. The Weibull distribution is in fact related to the *time-to-failure* of a system, which, in our context, is to be interpreted as the *time-to-exit* a room (more precisely as the *time-to-exit-and-do-not-return*, since we consider the ToP as the total time spent in a room). The parameter λ (characteristic time of visit) gives information about the room holding power, while parameter k (Weibull slope) characterises the decision to leave the room. We recall the probability density function (PDF) of the Weibull distribution being

$$f(x; \lambda, k) = \begin{cases} \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \ . \tag{4.12}$$

In Section 4.7, the survival and the hazard function associated with the Weibull distributions will be used as building blocks of the proposed museum digital twin.

Figure 4.21 reports the ToP empirical distributions and their Weibull fit for individual sample rooms as well as for the whole Galleria Borghese museum. In particular, for all rooms of the Galleria Borghese museum, $k > 1$ holds, meaning that the exit (failure) rate increases with time ($k = 1$ indicates that the exit rate is constant over time while $k < 1$ indicates a decreasing-in-time exit rate). In practice, we deem that visitors find all the rooms worthy of attention, and it happens rarely that visitors leave a room immediately.

Returning visitors

Museums in old historical buildings typically have an entangled structure that can lead the visitor to get lost. Furthermore, in museums, visitors are typically inclined to return to previously visited rooms to admire once again the most impactful artworks or to analyse them with a second eye.

For this reason, it is useful to quantify RET, a yet simple task that consists in evaluating

[6] The best solution according to the Akaike information criterion, in our experiments.

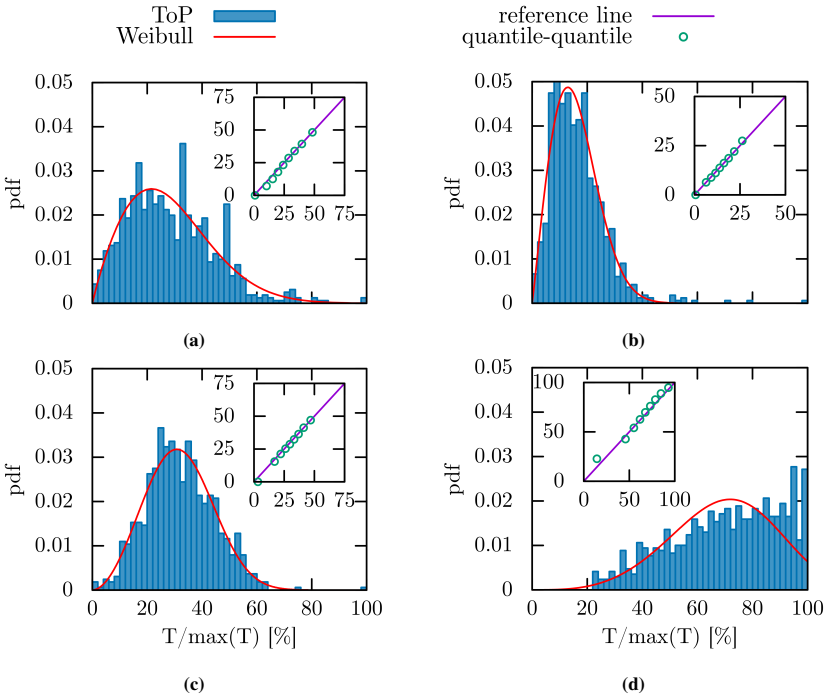


Fig. 4.21: Four ToP distributions and their Weibull fit. Inset: related Quantile-Quantile plots that depict the Real vs. Weibull quantile relation. **(a)** *Satiro su delfino* (r_7, $k = 1.8$, $\lambda = 17$). **(b)** *Apollo e Dafne* (r_3, $k = 2$, $\lambda = 36$). **(c)** Pinacoteque (r_9, $k = 2.8$, $\lambda = 221$). **(d)** Whole museum ($k = 4.1$, $\lambda = 572$). In the last case, the Weibull distribution does not fit correctly due to the forced exit after 2h. This problem will be solved later in Section 4.8.2, by censoring the last 5 minutes of the visit.

$$\textsc{Ret}(v, r) = \sum_{t=1}^{T} \mathbb{1}_{\mathfrak{t}_t^v = r} \cdot \ldots \cdot \mathbb{1}_{\mathfrak{t}_{t+\delta}^v = r} \cdot \mathbb{1}_{\mathfrak{t}_{t-1}^v \neq r} \,, \tag{4.13}$$

where the parameter δ is used to avoid counting *fast-crossings* of the room, like a visitor that simply enters the room to find out that he already visited it. We note, however, that δ should be kept in the order of the few units ($\delta = 3$ in our experiments) since usually returning visitors never return to a room for more than a couple of minutes.

The Galleria Borghese provides an excellent example for returning visitors analysis as it has no suggested path for visitors and the density of artworks is so remarkably high that visitors easily miss a fraction of the pieces during the first passage of a room.

We observe that each guest visits a room, on average, 1.3 times (1.5 times, for r_8, *Caravaggio*), while entrance rooms have 2.7 passages. On the other hand, 25% of the visitors skip at least one room (especially r_7, *Satiro su Delfino*). The time of permanence during the first passage by a room is generally the longest, in

comparison to the next ones (this, however, does not hold for entry rooms). The time of first return (time interval between the moment a visitor leaves a room and the moment they return) appears consistent throughout the museum rooms and is between 25 and 30 minutes.

Finally, we highlight that the occurrence of fast returns (less than 5 minutes), which in our case are about 10% of all returns, could indicate that visitors frequently get lost or change the direction of their visit (clockwise vs. counterclockwise, cf. museum map in Figure 4.2).

Despite being very different in conception and style, the Peggy Guggenheim Collection also provides a good example for returning visitors. In fact, like in Galleria Borghese, no suggested path is enforced by signage, and, conversely to the Galleria Borghese, only a few pieces of art are displayed in each room. However, due to the nature of modern art museums, visitors are likely inspired to return to admiring a few pieces of the collection, like Pollock's and Magritte's.

We observe that each guest visits a room 1.1 times (1.4 times for r_8) on average. The Sculpture garden (room r_{11}) reports 3.2 passages on average due to its central position and the longest visit is typically the second (as guests typically visit it after the main exhibition area). Furthermore, 18% of visitors skip at least one room of the main exhibition area, with $\sim$ 8% not visiting entirely the Schulhof collection r_{10}: this can be attributed to the fact that guests do not find it, actually thinking that the exhibition is made of the sole permanent exhibition (r_1 to r_9).

People per room

The number of people per room, PPR, is probably the most relevant indicator as well as that of largest interest for museum curators, as it connects with safety (hyper-congestion), comfort, and attractiveness for the audience (under-used rooms could indicate scarce interest). It follows that being able to evaluate it from Lagrangian data further unleashes the ability of a model capable of reproducing trajectories (see later, Section 4.7).

In order to evaluate PPR we first need to know how visitors enter the museum. Eulerian measurements for the entire museum come in handy since they provide an empirical distribution of the entrance time that can either be fitted or directly used to pad the trajectories.

To amend the fact that beacons are given to a sample of visitors, and only to one member of each social group, we consistently replicate each trajectory q times, where the integer q is uniformly distributed between 1 and 6.

Once trajectories have been coherently padded and magnified, $\mathrm{PPR}(r,t)$ can be directly evaluated by enumerating the trajectories that are in room r in time bin t:

$$\mathrm{PPR}(r,t) = \sum_{v=1}^{N} \mathbb{1}_{t_t^v = r} \ . \tag{4.14}$$

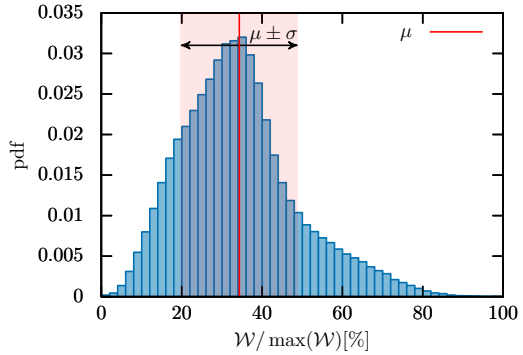


Fig. 4.22: Distribution of the mutual distances between trajectories from the Galleria Borghese dataset. The x-axis is normalized w.r.t. the longest measured distance. The mean pairwise distance μ is reported in red while the shaded area denotes the range $\mu \pm \sigma$ (σ being the standard deviation of the distribution).

The PPR time series of one room and of the whole museum is later used as a reference and compared with our simulations in Figure 4.32.

About the size of the dataset

Before using the datasets and the indicators introduced above, we need to check if the number of considered trajectories is enough to get useful information. To this end, we have compared ToP and PPR extracted from the whole dataset (848 trajectories in the Galleria Borghese case study and 481 trajectories in the Peggy Guggenheim Collection case study) with ToP and PPR extracted from several random subsets of the datasets with variable size. We observed that the difference between partial and complete datasets stabilises starting from 300 sampled trajectories, meaning that we would not have obtained significantly different results if we had tracked more than 300 trajectories.

4.6.2 Variability in trajectory dataset

The distance $\mathcal{W}$ introduced in Section 4.5 is a further very important tool when it comes to analysing a trajectory dataset. In particular, a strong indication of the variability amongst different trajectories can be obtained by evaluating the mutual pairwise distance between all trajectories and studying the corresponding distribution. As an example, in Figure 4.22 we report such distribution obtained from the Galleria Borghese dataset.

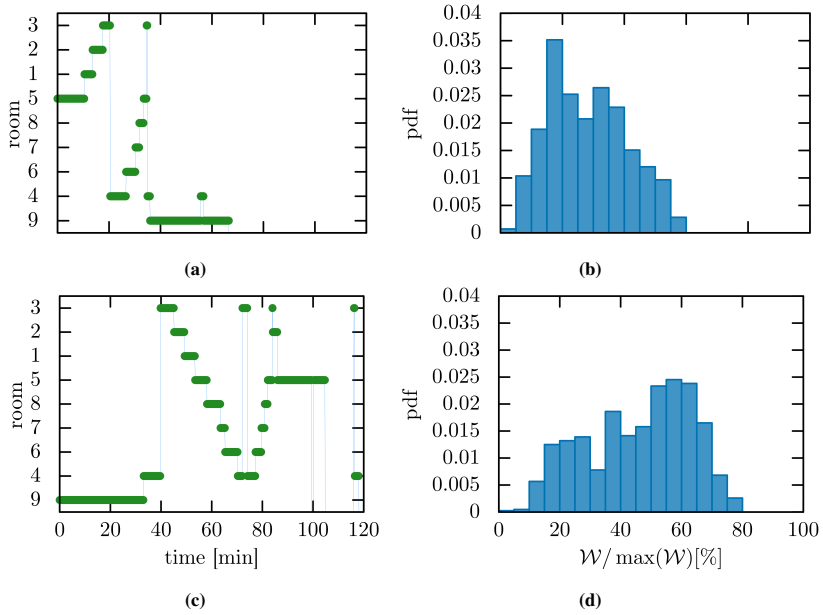


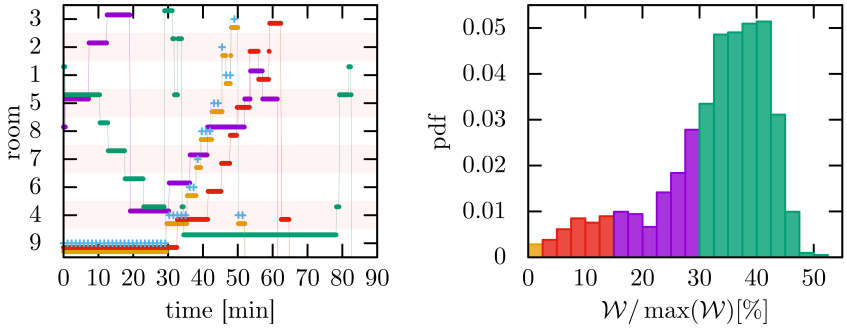
Fig. 4.23: **(a)** Most common trajectory from the Galleria Borghese dataset and, **(b)**, distribution of the distances between such trajectory and all the others. The visitor performs a circular visit following the room numbering on the main floor, then they reach the Pinacoteque upstairs. **(c)** & **(d)** Analogous plot for the least common trajectory in our dataset. The visitor enters the museum via the Pinacoteque and then they visit the main floor twice, once clockwise and once counterclockwise. x-axis in **(b)** & **(d)** is normalized *w.r.t.* the longest measured distance amongst all the trajectories.

The trajectory over the tails of this distribution also gives important information: the most present trajectories over the left (resp. right) tail correspond in fact to the most (resp. least) *common* trajectories of the dataset. Here "common" is to be intended as the *closer to the other* following the idea that the higher the number of trajectories *close* to a given one, the more common the trajectory is.

An example of this procedure applied to the Galleria Borghese dataset is given in Figure 4.23, where the single trajectory having the highest number of other trajectories within distance $\mu - \sigma$ and the one having the least number of other trajectories within distance $\mu + \sigma$ are reported.

Finally, we mention the capability of finding automatically members of social groups (it could happen that elements of the same social group went to the ticket office separately, and thus were assigned more than one beacon). Indeed, two or more trajectories very close to one another likely belong to visitors in a company.

Figure 4.24 reports a sample of trajectories from the Galleria Borghese dataset at different distances from a given reference trajectory.



(a) **(b)**

Fig. 4.24: (a) Sample of measured trajectories from the Galleria Borghese dataset. **(b)** Distribution of the distances between the trajectory marked by "blue plus signs" (+) in **(a)** and all the others. Distances are reported in percentage w.r.t. the longest distance measured. Trajectories closer than 0.025% (yellow bin) likely belong to the same group of visitors; trajectories closer than 0.15% (red bins) are slightly time-shifted; in trajectories closer than 0.30% (purple bins) relations are still identifiable; trajectories farther away than 0.30% (green ones) are completely unrelated. Trajectories in **(a)** are randomly sampled from corresponding colour percentile sets in **(b)**.

4.6.3 Clustering algorithms

As we recalled in Section 4.1.1, clustering algorithms can be used for inferring, from the whole trajectories dataset, the typical paths or, equivalently, the typical individual behaviours inside the museum.

Here we employ algorithms which do not require defining *a priori* the number k of clusters, nor to assign predefined reference trajectories around which clusters are agglomerated (as typically happens with, *e.g.*, k-means approach, see 2.4.1). Moreover, we do not use the typical taxonomy (ant, butterfly, fish, grasshopper, cf. Section 4.1.1) to guide the clustering, aiming at other, possibly hybrid, behaviours. To this end, we employ an *agglomerative hierarchical clustering analysis* (A-HCA) approach (see Section 2.4.2) to build the trajectory dendrogram.

To measure the distance between two clusters, we leverage on (4.8). We consider, in particular, the three common methods introduced in Section 2.4.2: complete linkage (C-LINK), single linkage (S-LINK), and Unweighted Pair Group Method with Mean Centroid (UPGMC).

Determining a representative trajectory $\bar{\mathfrak{t}}_C$ in a trajectory cluster C is useful in general, and mandatory to employ UPGMC. To do so, we compute a mode amongst all the trajectories: for each time bin t, our representative trajectory reports the most visited room amongst the elements of C:

$$(\bar{\mathfrak{t}}_C)_t := \operatorname*{argmax}_{r\in\{1,\dots,R\}} \left\{ \sum_{\mathfrak{r}\in C} \mathbb{1}_{\mathfrak{r}_t=r} \right\} , \qquad 0 < t \le T . \tag{4.15}$$

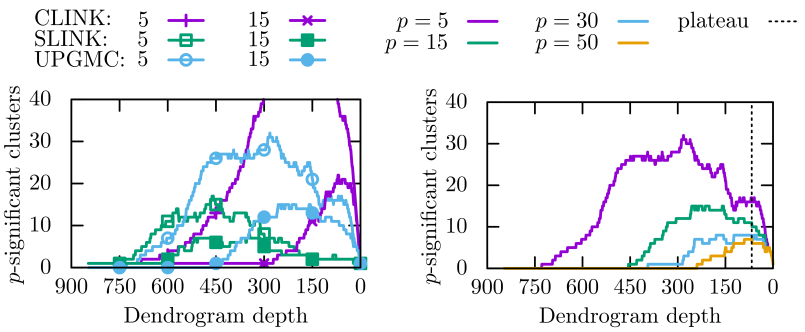


Fig. 4.25: **(a)** Number of 5- (filled markers) and 15- (empty markers) significant clusters in the Galleria Borghese dataset as a function of the dendrogram depth for C-LINK, S-LINK and UPGMC methods. **(b)** Number of $p = 5, 15, 30, 50$ significant clusters obtained via UPGMC method. The dendrogram is cut in correspondence to the plateau at depth 67.

Note that the centroids found with a specific cut may also be employed to cluster a different set of trajectories. This also means that, if new trajectories are gathered, the same centroids may be used in order to get a clustering. This may reveal that habits have been broken or new paths have been discovered.

For what concerns the choice of a valuable clustering amongst the family generated by HCA, we adopt the ξ-significant cluster strategy described in Section 2.4.2. In fact, we here recall that having a high variation in the number of significant clusters in the proximity of the root often implies that clusters are unstable, *i.e.* they merge randomly in the process, preventing valuable interpretations. Having instead a very small number of significant clusters, say one or two, often means that each cluster contains very non-homogeneous elements, thus resulting in less valuable categorisations in practice.

Clustering results in Galleria Borghese

Figure 4.25**(a)** reports the number of 5- and 15-significant clusters as a function of the dendrogram depth for the Galleria Borghese dataset. C-LINK yields many small unstable clusters joining together, with no meaningful interpretation, towards the end of the process. S-LINK offers, on the other hand, a poor set of typical clusters to which all the trajectories converge quickly throughout the clustering process. Conversely, UPGMC leads to a relatively small amount of consistent stable clusters.

In particular, the UPGMC dendrogram shows a plateau around layer $\bar{\ell} \sim 67$, for many values of p, see Figure 4.25**(b)**. We adopt such a cutting layer since it ensures the maximum amount of highly significant clusters ($p = 30, 50$ have the last absolute maximum there) without trading off too much information in smaller clusters.

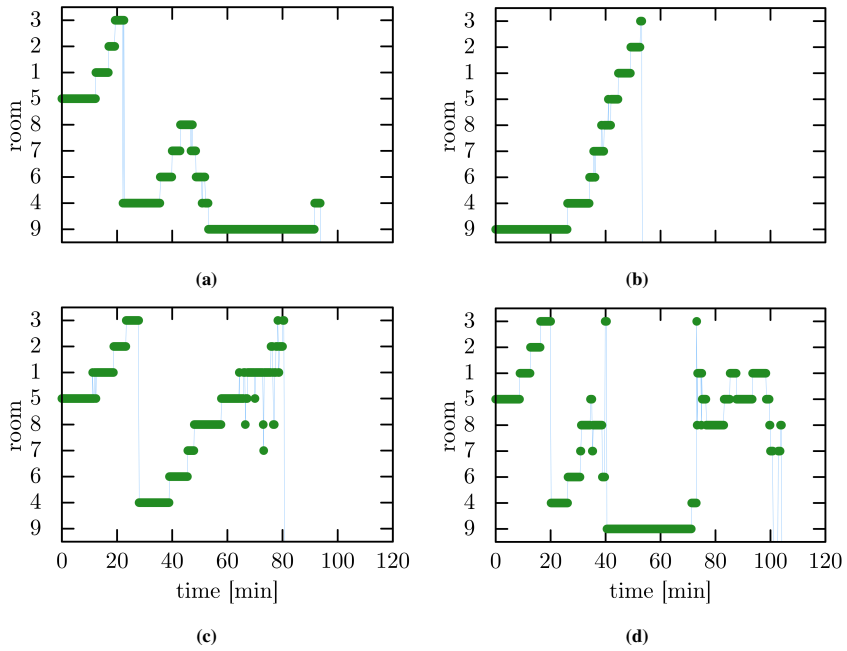


Fig. 4.26: Four representative trajectories (centroids) of the Galleria Borghese clusters joining respectively **(a)** 16%, **(b)** 9%, **(c)** 4%, **(d)** 1% of the trajectory dataset. Representative trajectories may show spikes (see, *e.g.*, **(c)**, $\sim 75^{\text{TH}}$ minute). According to (4.15), this phenomenon arises whenever rooms have approximately the same number of visitors within the same interval of time.

We consider here the representative trajectories of each cluster obtained after a dendrogram cut at layer $\bar{\ell}$. Although none of the representative trajectories strictly coincides with any amongst the trajectories observed, they all appear real (i.e. conform with a potential visit). This emphasises that clusters indeed aggregate similar trajectories.

Figure 4.26 shows four representative trajectories related to four clusters of different sizes. The two most common patterns are related to visitors who follow the natural numbering of the rooms, starting or ending the visit in the Pinacoteque, which is visited once. This identifies the most typical visit pattern for the curators. Nevertheless, clustering investigation brings to light other, less expected, patterns: the one which does not include the visit at the Pinacoteque (possibly visitors who did not find the staircase) and patterns where the Pinacoteque comes amidst the visit. Note that both patterns have been observed by the museum managers and are discouraged.

Observation (Filtering by clustering) Clustering can be also used to detect unfeasible/unreal trajectories coming from system malfunctioning; in fact, those trajectories tend to gather in a single cluster. This powerful feature helps to design filters to clean up the data during the preprocessing phase.

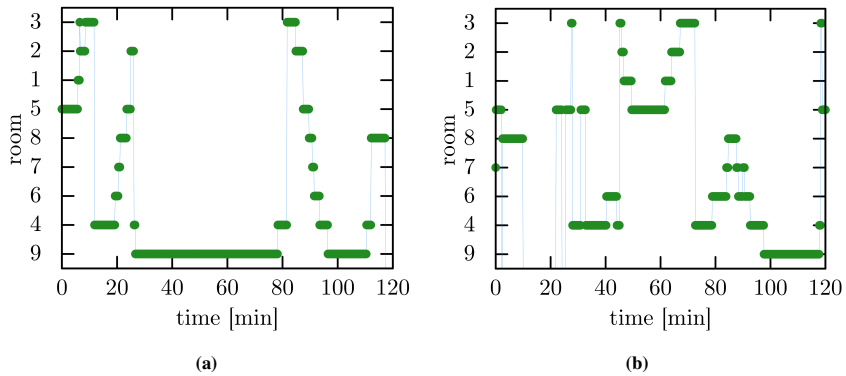


Fig. 4.27: Two anomalies of the Galleria Borghese dataset. **(a)** A rare pattern where the Pinacoteque and main floor are both visited twice. **(b)** A strange pattern with many changes of direction (clockwise/counterclockwise).

Observation (Anomaly detection) Trajectories which remain isolated in the last layers (close to the root of the tree) are, by definition, far from all the other centroids and therefore very atypical. We claim that these trajectories are *anomalies* detected during the process. If they do not come from system malfunctioning, they belong to people who behave abnormally or suspiciously and deem additional checks.

Figure 4.27 shows some of the anomalies detected in our study.

4.6.4 Clustering on coarser trajectories

As the Galleria Borghese case study demonstrates, clustering on full trajectories yields valuable insights into visiting anomalies and meaningful centroids for clusterings through pseudo-trajectories that, at first glance, are indistinguishable from real ones.

However, to extract more precise and useful pieces of information, it might be advantageous to work with coarser representations of the trajectories. This can be achieved in three main ways, each capable of capturing different behaviours: (i) considering a coarser representation of the museum, (ii) excluding the concept of ToP from the analysis and focusing solely on the sequence of rooms $r_{i_1}, r_{i_2}, \ldots$ (cf. [95, 132]), or (iii) combining both approaches.

Considering a coarser representation of the museum is a natural progression, especially with the introduction of colour-clustering described in Section 4.3.2. Trajectories can be reconstructed without applying all the localisers of the cascaded selector (cf. Section 4.5.3), resulting in a trajectory that consists of a succession of colour clusters rather than virtual rooms.

The main advantage of this approach is the reduced number of clusters due to the limited number of available *rooms*, making it easier to visualise and interpret the results.

Dropping the concept of ToP from the analysis is a widely used technique in literature. It significantly reduces the size of trajectories and allows for the use of alternative metrics compared to the complex definition of $\mathcal{W}$. It is the case, *e.g.*, the Levenshtein distance (edit distance, see [121]) that measures the minimum number of single-character edits (insertions, deletions, substitutions) required to transform one string into another.

This approach primarily focuses on the visitor's pattern of visits, highlighting the typical path followed regardless of the time spent in each room. Due to its nature, clustering based on such coarsened trajectories particularly emphasises the RET metric.

By combining the aforementioned methodologies, we obtain highly compact coarsened trajectories that still carry valuable information. These trajectories provide a high-level pattern of museum visits that can be processed using even simpler metrics, such as the Kronecker delta ($\mathfrak{D}^{\kappa}(r_i, r_j) = \mathbb{1}_{r_i \neq r_j}$).

Clusterings resulting from the analysis of such a dataset typically yield easily interpretable results that can be directly understood by museum curators to assess whether visitors are behaving as expected, at least at a high level.

Clustering results in the Peggy Guggenheim Collection

Being more intricate in its structure, the Peggy Guggenheim Collection serves as a compelling case study for elucidating the coarsening of trajectories. Specifically, the eight-shaped layout of the building vividly demonstrates why flux crossing can result in significant issues, leading to overcrowding in Calder Hall (r_9). Guests who begin their visit on the left wing frequently intersect with visitors who have opted for the right wing first, thereby converging at the centre and typically causing disorder.

This observation suggests that we should halt trajectory reconstruction before applying the room localisers $\mathfrak{R}_{\mathrm{L}}$ and $\mathfrak{R}_{\mathrm{R}}$ (see Figure 4.20(**b**)), thereby generating trajectories sampled from $r_{\mathrm{L}}, r_{\mathrm{R}}, r_9, r_{10}, r_{11}, r_{12}, r_{13}, r_{14}, r_{15}$. Figure 4.28 presents a sample visit with its coarsened reconstruction.

Furthermore, to focus primarily on the inherent patterns within the permanent exhibition, we can omit the ToP and further reduce the room set to $r_{\mathrm{L}}, r_{\mathrm{R}}, r_{\mathrm{S}}$, resulting in an extremely compact string representation of visits (here, we denote $r_{\mathrm{S}} = r_{10}$ for consistency, to maintain its correspondence with the Schulhof collection).

A typical trajectory can then be represented by a string such as $\mathtt{t} = (\mathtt{R}, \mathtt{L}, \mathtt{R}, \mathtt{S})$. Figure 4.29 displays a pie chart illustrating the most common trajectory patterns, with $\mathtt{t}^{\star_1} = (\mathtt{L}, \mathtt{R}, \mathtt{S})$ being the most frequent, followed by $\mathtt{t}^{\star_2} = (\mathtt{R}, \mathtt{S}, \mathtt{L})$, $\mathtt{t}^{\star_3} = (\mathtt{R}, \mathtt{L}, \mathtt{S})$, and $\mathtt{t}^{\star_4} = (\mathtt{S}, \mathtt{R}, \mathtt{L})$.

Moreover, guests who initiate their visit from r_9 are equally divided between the left and right wings of the museum, resulting in crossing flows from right to left

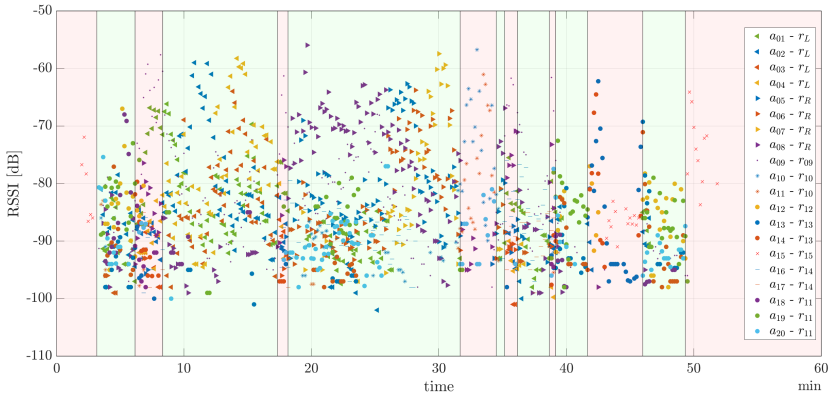


Fig. 4.28: Trajectory reconstruction of a sample beacon from the Peggy Guggenheim Collection dataset. The reconstruction is coarsened both in space (aggregating $r_{\text{L}} \equiv \{r_1, r_2, r_3, r_4\}$ and $r_{\text{R}} \equiv \{r_5, r_6, r_7, r_8\}$) and time (dropping ToP indicator), hence resulting in:

$$\text{t} = (r_{15}, r_{11}, r_9, r_{\text{L}}, r_9, r_{\text{R}}, r_{10}, r_{\text{R}}, r_9, r_{14}, r_9, r_{11}, r_{13}, r_{11}, r_{15}) \ .$$

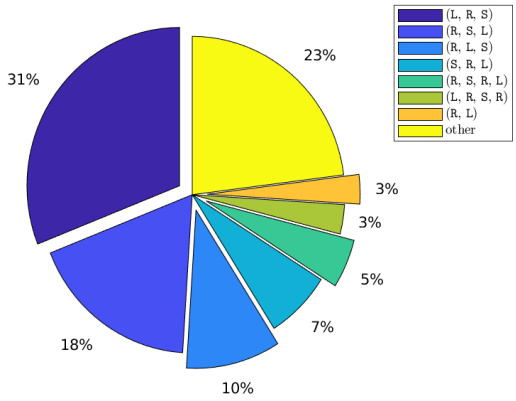


Fig. 4.29: Pie chart reporting the most common coarsen trajectories in the Peggy Guggenheim Collection dataset, restricted to the main exhibition area. ToP has been dropped and macro-areas are as follows: $\text{L} \equiv \{r_1, r_2, r_3, r_4\}$, $\text{R} \equiv \{r_5, r_6, r_7, r_8\}$, and $\text{S} \equiv \{r_{10}\}$.

that can be mitigated by suggesting an improved visiting pattern (*e.g.* by means of signage, see later Section 4.9 and Figure 4.40).

Lastly, it is noteworthy that the data reveals an unexpected behaviour: 10% of visitors start their visit from r_{10}, contrary to the expectations of the museum curator. This behaviour highlights a critical issue wherein visitors who are unfamiliar with the museum entrance system may mistakenly perceive the Schulhof collection as the entry point due to observing other visitors exiting from there.

4.7 Model and calibration

In this section, we develop a Lagrangian digital twin of the museum, which is an algorithm capable of generating new trajectories that are statistically indistinguishable from the measured ones.

To represent the complex visitor behaviour, we employ a stochastic approach based on Markov chains (cf. Section 1.7). Our simulator is designed to generate visiting paths with relevant observable features, such as guests skipping one or more rooms and/or returning multiple times to the same room.

The model is based on two important assumptions:

Visitors are independent of each other The decision to leave or remain in a room does not depend on the number of people in that room. This assumption is certainly reasonable up to mild congestion levels. On the other hand, hyper-congestion has surely an impact on visitors' choices, however, our current data collection seems still insufficient to quantify such a challenging aspect. We suspect that congestion can either increase or decrease the ToP, depending on the perceived importance and fame of the room content.

Social groups behave as one individual Social groups visit the museum remaining together, *i.e.* following the same trajectory and thus spending the same time in each room. This assumption, which is an important limitation, is consistent with the fact that beacons were given almost always to a single member of each social group. Therefore, we are not capable of disentangling interactions and differences within social groups.

In a regular Markov chain, the transition probability from one state (room) to the next depends only on the current state. However, in our context, it is intuitive to assume that visitors' choices depend on the rooms they have previously visited. Furthermore, since there is no predefined visit path, a naive Markov chain would create a *bounce phenomenon* amongst rooms (*e.g.*, $r_1 \to r_2 \to r_3 \to r_2 \to r_1 \to r_2$), while the majority of paths are more regular (*e.g.*, $r_1 \to r_2 \to r_3 \to r_4$ or $r_4 \to r_3 \to r_2 \to r_1$).

To simulate complex visitor behaviour, we hence introduce a concept of *memory* in the Markov chain to represent visitors' knowledge of the visited rooms. We also assume that visitors remember the time spent in each room. We use a non-

homogeneous transition matrix that is time-dependent through a weight function S. Henceforth, we refer to this model as a *time-varying Markov model*.

4.7.1 Time-varying Markov model (TVMM)

Assuming the museum comes with R non-sink rooms[7], in order to construct the time-varying Markov model we start considering an $R \times R$ matrix $\mathcal{K}$ representing room transitions. Following the most frequent definition of probability, we evaluate $\mathcal{K}_{i,j}$ as the number of transitions from room r_i to room r_j within all the measured trajectories, where $r_i = r_j$ holds if the visitor remains in the same room, *i.e.*:

$$\mathcal{K}_{i,j} = \sum_{v=1}^{N} k_v(r_i, r_j) \qquad \text{where} \qquad k_v(r_i, r_j) = \sum_{t=1}^{T-1} \mathbb{1}_{\mathfrak{t}_t^v = r_i} \cdot \mathbb{1}_{\mathfrak{t}_{t+1}^v = r_j} \,. \tag{4.16}$$

In other words, we have that $k_v(r_i, r_j)$ denotes the number of $r_i \to r_j$ transitions along the v-th trajectory of the dataset. Hence, the sum over rows of $\mathcal{K}$ (or by columns, with an offset of 1 unit on the first/last room) represents the total time, in time bins, spent by all tracked visitors in each room, *i.e.* it holds

$$\sum_{v=1}^{N} \mathrm{ToP}(v, r_i) = \sum_{k=1}^{R} \mathcal{K}_{i,k} \qquad \text{since} \qquad \mathrm{ToP}(v, r_i) = \sum_{k=1}^{R} k_v(r_i, r_k) \,. \tag{4.17}$$

To obtain transition probabilities, we perform a by-row sum-normalisation of $\mathcal{K}$, resulting in a new matrix $\bar{\mathcal{K}}$ defined as:

$$\bar{\mathcal{K}}_{i,j} = \frac{\mathcal{K}_{i,j}}{\sum_{k=1}^{N} \mathcal{K}_{i,k}} \,. \tag{4.18}$$

Hence, the element $\bar{\mathcal{K}}_{i,j}$ obtains the interpretation of the probability of moving from room r_i to room r_j. Do note that, as far as all the rooms have been visited at least once by a single visitor, the normalisation is well-posed since the graph associated with the museum is strongly connected and, consequently, no row $\mathcal{K}_{\cdot,j}$ is identically equal to zero.

To avoid the *room bounce phenomenon* of visitors moving back and forth between rooms, we introduce a time-dependent matrix $\mathcal{M}(t)$ based on $\bar{\mathcal{K}}$. More precisely, we consider:

$$\mathcal{M}_{i,j}(t) = \bar{\mathcal{K}}_{i,j} \cdot S_{r_j}(t), \qquad i, j \in \{1, \ldots, R\} \,, \tag{4.19}$$

where $S_r(t)$ is the survival function of the statistical distribution χ_r associated with the empirical distribution $\mathrm{ToP}(\cdot, r)$. In other words, $S_r(t)$ quantifies the probability that a guest spends a time interval longer than t in room r. In particular, $S_r(t)$ is a

[7] We recall that exit rooms are shaped as sink inside the graph representation. Here we omit them from matrix $\mathcal{K}$ since they will be modelled differently, see later this section.

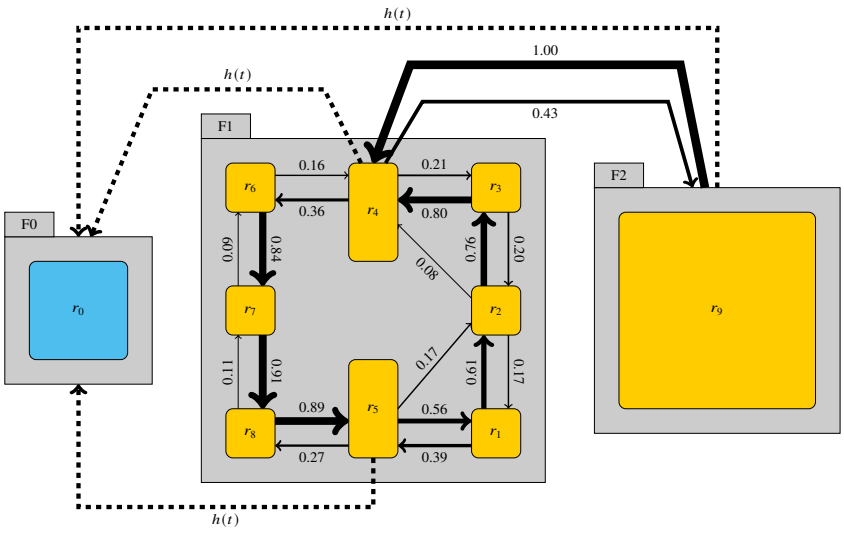


Fig. 4.30: The Galleria Borghese time-varying Markov model (cf. Figure 4.13(**a**)). Transition probabilities $\bar{\mathcal{M}}(0) = \bar{\mathcal{K}}$ between rooms are reported both numerically and pictorially. The probability to remain in the same room is not included for the sake of readability. We can see that the counter-clockwise path is preferred and fast transitions from rooms r_5, r_2 to rooms r_2, r_4, respectively, are admitted. The probability of leaving the museum is sampled as the hazard function $h^{\text{WEI}}(t)$.

monotone decreasing function such that $S_r(0) = 1$, $S_r(t) > 0$ and $S_r(t) \to 0$ for $t \to t_r^{\text{MAX}}$, where t_r^{MAX} is the largest measured $\text{ToP}(\cdot, r)$, *i.e.* $t_r^{\text{MAX}} = \max\{\text{ToP}(v, r) \mid v \in \{0, \dots, N\}\}$. Do note that it follows $\mathcal{M}(0) = \bar{\mathcal{K}}$.

We recall the (λ_r, k_r)-Weibull survival function (cf. Section 4.6.1) being

$$S_r^{\text{WEI}}(t) \equiv S_r^{\text{WEI}}(t; \lambda_r, k_r) = e^{-(t/\lambda_r)^{k_r}} \ . \tag{4.20}$$

During the simulation, the function $S_r(t)$ is updated based on the time spent in each room, concurrently modifying the value of $\mathcal{M}(t)$. Sum-normalisation by rows is further required in order to obtain a valid transition matrix $\bar{\mathcal{M}}(t)$. Do note that, since $S_r(t) > 0$ for $t > 0$, row-normalisation is well-posed.

Finally, we have to tackle the task of terminating the visit (*i.e.* how a trajectory exits from the museum). Assuming the existence of a room $r_\star$ marked as *exit* (a sink for the graph representation), it makes sense to avoid considering it within the matrix $\mathcal{K}$ since, usually, a visitor follows the concept of *once-exit-alway-exit*. Following this idea, we modelled the exit via the hazard function $h_\star(t)$ associated with the statistical distribution of the total time of visit $\chi_\star$. Namely, we have:

$$\mathbb{P}(\mathrm{t}_{t+1} = r_\star) = \begin{cases} h_\star(t) & \text{if } \mathrm{t}_t \sim r_\star \\ 0 & \text{otherwise} \end{cases} \ , \tag{4.21}$$

where $\sim$ denotes the adjacency over the museum graph.

We recall the $(\lambda_\star, k_\star)$-Weibull hazard function being

$$h_\star^{\text{WEI}} \equiv h_\star^{\text{WEI}}(t; \lambda_\star, k_\star) = \frac{k_\star}{{\lambda_\star}^{k_\star}} \cdot t^{k_\star - 1}. \tag{4.22}$$

Figure 4.30 provides a visual representation of the time-varying Markov model built for the Galleria Borghese case study. Such model, with the appropriate transition probabilities and exit probabilities extracted in Section 4.6.1 (cf. Figure 4.21), allows us to simulate visitor behaviour in a way that captures the observed characteristics of the museum visits.

4.7.2 Creation and validation of a virtual dataset of trajectories

Once the time-varying Markov model we introduced in the previous section is set, we can run it to produce virtual trajectories. Novel trajectories can, in principle, stick with any time granularity Δt. However, in our experiments, we set $\Delta t = 10$s to match our reconstructed trajectories, hence building a dataset simpler to compare.

In order to create a novel trajectory $\mathfrak{s}$, the starting room $r_\star$ is set at $t = 1$, *i.e.* $\mathfrak{s}_1 = r_\star$. If more than one entrance is available, then which one is the starting room is randomly chosen in accordance with the empirical distribution of entrances. Such distribution can either be derived by the Lagrangian dataset or can be estimated by other means, like an *ad hoc* measurement with the tool introduced in Section 4.4.2.

The model is then run by firstly initialising to zero a $R+1$-length vector $\mathbf{p}$ holding the trajectory ToP value for the R rooms ($\mathbf{p}_r$) and for the whole museum ($\mathbf{p}_\star$). At each time step t, the following instructions are executed:

1. If the exit conditions are met, then the visit (and the process) stops, that is, given the un-biased coin $c \leftarrow_{\$} [0, 1]$:

$$\mathfrak{s}_{t+1} = r_\star \qquad \text{if} \qquad \mathfrak{s}_t \sim r_\star \qquad \text{and} \qquad c < h_\star(\mathbf{p}_\star)\ . \tag{4.23}$$

2. Otherwise, the model advances the Markov chain by sampling the next room $\mathfrak{s}_{t+1} \leftarrow_{\$} \psi(r, t)$, where the distribution ψ is defined over the room set $\{r_1, \ldots, r_R\}$ and is given by

$$\psi(r, t) = \bar{\mathcal{M}}_{\mathfrak{s}_t, r}(\mathbf{p}_t) = \frac{\bar{\mathcal{K}}_{\mathfrak{s}_t, r} \cdot S_r(\mathbf{p}_t)}{Z}\ , \tag{4.24}$$

   with Z being the suitable normalisation constant.
3. Finally, the time of permanence is updated, meaning $\mathbf{p}_{\mathfrak{t}_{t+1}}$ and $\mathbf{p}_\star$ are both increased by one time bin.

The procedure can be repeated at will to build up a set of datasets of independent trajectories.

In addition to validating trajectories visually, we should then compare our new *in silico* dataset with the reconstructed one. It is in fact mandatory to quantify the

accuracy of the simulation through a set of observables. To this scope, we employ the following:

ToP We evaluate mean μ, standard deviation σ and variation coefficient $\mathrm{VC} = \sigma/\mu$ of the $\mathrm{ToP}(\cdot, r)$ distributions, for both real and simulated visits. We use the relative error of mean ($\delta\mu$) and variation coefficient ($\delta\,\mathrm{VC}$) as observables, which are:

$$\delta\mu = \frac{\mu_{sim}}{\mu_{real}} - 1\,, \qquad \delta\mathrm{VC} = \frac{\mathrm{VC}_{sim}}{\mathrm{VC}_{real}} - 1. \tag{4.25}$$

PPR We consider $K = 100$ statistically independent simulated days of visits, each including a total of generated trajectories coherent with a real visit day (similarly to the PPR in Section 4.6.1, simulated visits are replicated q times, where q is a uniform integer random variable between 1 and 6, to mimic social groups). We compute the PPR at each time bin as an ensemble average across such K realisations and we confront the results with the empirical PPR from both the Lagrangian and the Eulerian datasets.

Clusters We use the same HCA technique presented in Section 4.6.3 to aggregate simulated trajectories. This analysis aims to check if the most numerous cluster is sufficiently close to the measurements; this guarantees that the simulator creates a sufficient amount of plausible trajectories.

4.7.3 Simulation results in Galleria Borghese case study

The entrance system in the Galleria Borghese museum is pretty peculiar since visit turns are non-overlapping, due to the fact that the museum empties every two hours. For this reason, visitors tend to be on time, hence entering the museum like within a batch instead of a few at time. However, due to some delays (ticket control, late arrival, queue at wardrobe), the entrance process is completed in about 20 minutes. We simulate these dynamics by fixing $T = 2\mathrm{h}$ and extracting the delay t_0 at random from a set of measured delays and setting $\mathfrak{s}_t = r_0$ for $t < t_0$ (cf. Figures 4.13**(a)** and 4.30).

In addition, the museum has three possible mutual points of access from the entrance ($r_\star = r_0$). We sample the first room amongst *Ratto di Proserpina* ($e_1 = r_4$), *Portico* ($e_2 = r_5$), and Pinacoteque ($e_3 = r_9$) according to the measurements made, *i.e.*

$$\mathbb{P}[\mathfrak{s}_{t_0} = e_1] = 0.15\,, \qquad \mathbb{P}[\mathfrak{s}_{t_0} = e_2] = 0.60\,, \qquad \mathbb{P}[\mathfrak{s}_{t_0} = e_3] = 0.25\,. \tag{4.26}$$

Finally, since visitors are forced to exit after two hours, we slightly modified the exit condition so that if $t = T$ and a visitor is still in the museum, then the visit is considered finished, and the visitor is moved to the exit room. Coherently, if the *in silico* visitor decides to leave the museum at time $t < T$, then the trajectory is padded with the exit room to match the T-length.

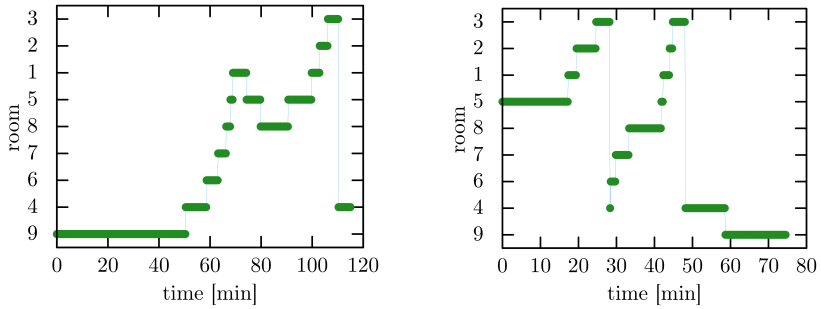


Fig. 4.31: Two simulated trajectories inside the Galleria Borghese. **(a)** A long trajectory which begins from the second floor (r_9) and then moves to the main floor following the room enumeration. **(b)** A short trajectory that begins from the main floor (r_5), traverses it according to the room enumeration, and finally reaches the second floor.

Room	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	Museum
$\delta\mu$	11%	10%	2%	8%	12%	−3%	−2%	7%	−8%	−3%
δVC	−28%	−10%	−15%	−15%	−20%	13%	12%	3%	31%	−11%

Table 4.5: Relative error between mean and variation coefficient of ToP distribution evaluated for simulated vs. real trajectories in the Galleria Borghese case study.

Figure 4.31 shows two representative simulated trajectories, which indeed share typical features with measurements: the Pinacoteque is visited once and the visit path follows the natural numbering of rooms. At times, people backtrack to rooms already visited, as in real life.

Table 4.5 compares the real and simulated ToP distributions, by considering the relative differences in ToP averages and variation coefficient. The mean values of the distributions are well approximated, despite the simulations tend to slightly overestimate the ToP in the main floor and to underestimate it in the Pinacoteque. The δVC indicator highlights instead some differences between model and data: real visitors are more unpredictable than simulated ones, which yields negative δVC values. On the contrary, the dynamics in the Pinacoteque appear predictable and even more consistent than in simulations. This most likely relates to the fact that the Pinacoteque is the area with the weakest antenna coverage: amending not-detected data diminishes the variance of the measured ToP distribution.

In Figure 4.32, we compare measurements and simulations considering the PpR as a function of time. Simulations are reported in terms of ensemble statistics amongst 100 realisations of 400 visits; in particular, we consider ensemble PpR average and ensemble PpR standard deviation.

In Figure 4.33 we finally report the representative trajectories of the two most numerous clusters obtained by gathering real and simulated trajectories.

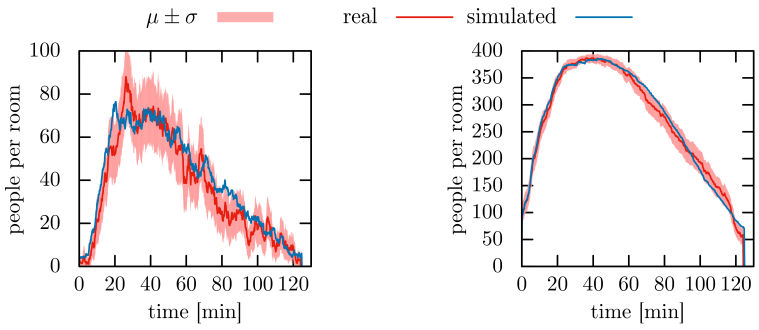


Fig. 4.32: Comparison of the average PpR of real visits (red line) and the ensemble-average PpR of simulated visits (blue line) in: **(a)** *Ratto di Proserpina* (r_4); **(b)** the whole museum of Galleria Borghese. The shaded area corresponds to the interval $[\mu - \sigma, \mu + \sigma]$. We note that the blue line is almost entirely contained in the shaded area, as expected.

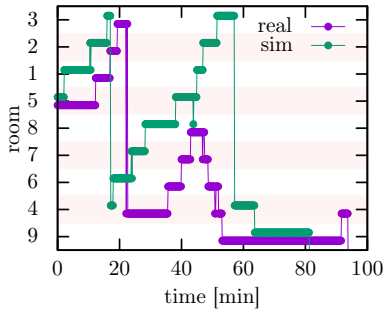


Fig. 4.33: The centroids of the two most numerous clusters obtained by applying hierarchical clustering over real and simulated datasets from the Galleria Borghese case study. The real case joins 16% of the dataset, whereas the simulated one joins 18%. We note that the centroids share a number features, *e.g.* the ToP in each room, the total time of visit, the entry room (*Portico*, r_5), and the final room (Pinacoteque, r_9). The main difference is the behaviour after completing the visit on the main floor. Real visitors come back counterclockwise, while simulated visitors keep walking clockwise. This could be explained by the fact that many visitors ask for information in room r_5 and are sent backwards to the staircases. The model does not include the interactions with the museum staff, hence cannot catch this feature.

4.8 Museum control and optimisation in Galleria Borghese

We are now ready to employ the digital twin introduced in the previous section as a tool to improve the museum experience. More precisely, we simulate different scenarios and observe visitors' behaviour in virtual environments, aiming at supporting curators' decisions. In this regard, it is useful to remark that changing the ticketing strategy or the duties of security staff can require weeks of training in real life.

For our case study, we identify the following control variables (C1–C4) and objectives (O1–O4).

C1 Considering that Galleria Borghese has three entrances (*Ratto di Proserpina*, *Portico*, and Pinacoteque), museum managers can assign a certain percentage of visitors to each entrance. Operationally, such control can be implemented by introducing a tag (*e.g.* name or colour) in the ticket which specifies the entrance.
C2 The scheduled entry times in the museum can be tuned.
C3 The number of visitors allowed in each turn can be modified.
C4 The fixed duration of a visit turn can be either (**C4a**) modified or (**C4b**) totally removed.

Objectives:

O1 Keeping the PPR below a certain room-dependent threshold. Historically, our study began precisely to control the number of visitors in the Pinacoteque, which has a very low admittance limit for safety reasons.
O2 Keeping the PPR, in any room at any time, approximately constant. This would reduce strong variations of relative humidity which can damage the artworks [94, Chapter 2].
O3 Decreasing the queue at the entrance.
O4 Increasing the number of visitors per day.

We recall that before and during our experimental campaign, the control variables were set as follows

C1 15%, 60%, and 25% of the visitors entering from *Ratto di Proserpina*, *Portico*, and Pinacoteque, respectively.
C2 Visitors entered at 09:00, 11:00, 13:00, 15:00, and 17:00, with the museum being completely empty.
C3 360 visitors were allowed to book for the visit in advance plus 30 *last-minute* visitors.
C4 Fixed 2h slots of visit.

Note that **O1** is also strictly related to the social distance issue, a hot topic during the emergency situation caused by the Covid-19 pandemic.

Among the many possibilities, we focused on two improvements: **C1** aiming at **O1**, and **C2**, **C3** & **C4b** aiming at **O1** & **O2**.

4.8.1 Entrance strategy optimisation

Keeping the existing conditions regarding the number of visitors and the time horizon, we explore the effects of a different visitor partition amongst the three entrances (**C1**). We aim at a PPR as low as possible in all rooms (**O1**), especially in the Pinacoteque, which is the room with the most stringent safety constraints.

We fix an *overcrowding threshold* for each room, representing a PPR limit the curators do not want to exceed. Then, we pursue a brute force attack on the optimisation

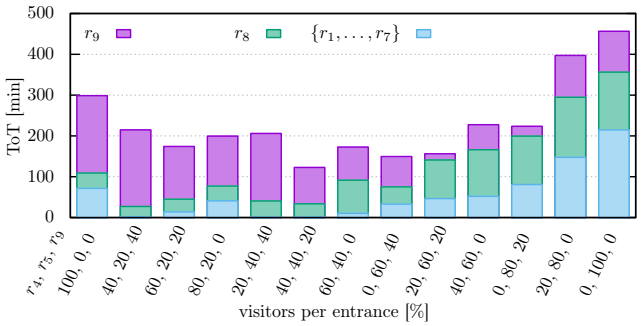


Fig. 4.34: Total time duration in which the overcrowding threshold is exceeded (ToT) in room r_9 (Pinacoteque), in room r_8 (*Caravaggio*), and in all other rooms of the Galleria Borghese (sum of each ToT is considered), for 13 triplets (E_1, E_2, E_3). We observe that the overall ToT exceeds at least 120 min over a day of visits regardless of the entrance system.

problem, trying all the possible triplets $(E_1, E_2, E_3) \in [0, 100]^3$, $\sum_{e=1}^{3} E_e = 100$, which indicates the percentages of visitors starting the visit from each entrance e_1 (*Ratto di Proserpina*), e_2 (*Portico*), and e_3 (Pinacoteque).

Figure 4.34 shows the results of the optimisation process evaluating the *total time the PPR exceeds the overcrowding threshold* (ToT, time over threshold), for room r_9 (Pinacoteque), room r_8 (*Caravaggio*), and for all the remaining rooms. The best triplet for the Pinacoteque is $(20, 60, 20)$, while the best triplet for *Caravaggio* is $(40, 20, 40)$: in fact, these configurations minimise the ToT in those rooms, respectively. More in general, it is easy to see that optimal choices for one room do not necessarily mean optimality for others. The solution currently employed by the museum, which is $(15, 60, 25)$, is almost optimal to reduce overcrowding in Pinacoteque and in the whole museum in general, but it sacrifices the pleasantness of the visit in some rooms on the main floor.

4.8.2 Removing the finite time horizon of the visits

The full elimination of the current finite time horizon allowed for the visits is a challenging improvement for the museum experience. The idea is to keep the reservation mandatory while setting an entry interval fixed every 30, 60, or 120 minutes (**C2**), but, unlike the current setting, *remove the requirement to leave after 2h* (**C4b**). The immediate advantage is that the museum staff does not have to empty the museum at the end of the visit turn, thus saving about 5-7 minutes during which the museum remains completely empty (**O2**). Moreover, this would also be a great advantage for the (few) visitors who want to stay for a very long time inside the museum.

Unfortunately, as it happens for every mathematical model, simulation results are reliable only in the conditions in which the simulator was developed and calibrated.

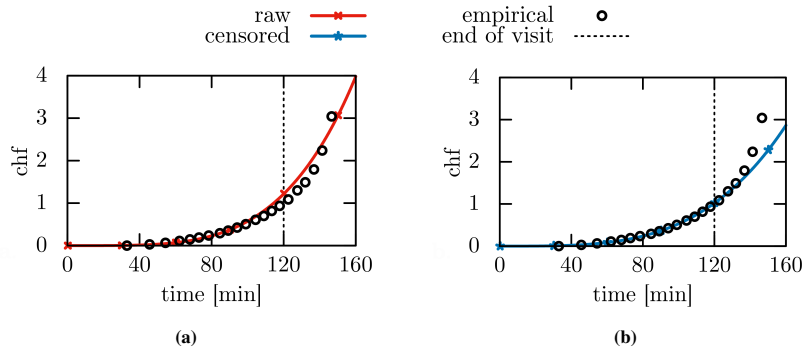


Fig. 4.35: Cumulative hazard function associated with the Weibull distribution of the whole museum of Galleria Borghese. Empirical values are evaluated with the Kaplan–Meier method. **(a)** Without censoring (cf. Figure 4.21**(d)**) and **(b)** after censoring the last 5 visit minutes (new parameters are $k_\star = 3.5$ and $\lambda_\star = 596$). This method allows us to get a better fit of the real distribution between 0 and 2h, *i.e.* the visit interval. The uncensored fit, instead, is negatively influenced by the forced exit.

In our measurements, less than 1/4 of visitors are still inside the museum when the time limit is reached (and are forced to exit); for these, a (negative) influence of the time limit certainly occurs. Nevertheless, such influence possibly exists also for the other 3/4, that might have scheduled their visit according to the existing time constraints.

We tackled the problem by *censoring* the Weibull distribution associated with the visit time of the whole museum (cf. Section 4.6.1). This statistical procedure allows us to deal with a dataset in which the event of interest is not observed during the study. We obtain the new distribution as a maximum likelihood estimate censoring the last 5 visit minutes, see Figure 4.35. We use the estimated parameters to modify the hazard function which controls the conclusion of the visit.

We simulated an entire day i.e. 9 a.m. – 7 p.m., corresponding to the total time span of the 5 visit turns currently implemented. This is necessary as after removing the time limit, visit turns overlap and the museum never empties. Figure 4.36 shows the result of the optimisation process. The best strategy is to let 100 visitors (**C3**) enter from the main floor (**C1**) every 30 minutes (**C2**). These choices eliminate completely the peaks in the PPR indicator (congestion moments, **O1**) and the PPR remains stable with small fluctuations during the whole visit day (**O2**). Having the system approximately at this thermodynamic-like equilibrium greatly facilitates the management since it allows us to evaluate – using the measured transition matrix – the average number of people in each room from the number of visitors allowed (*i.e.* sold tickets).

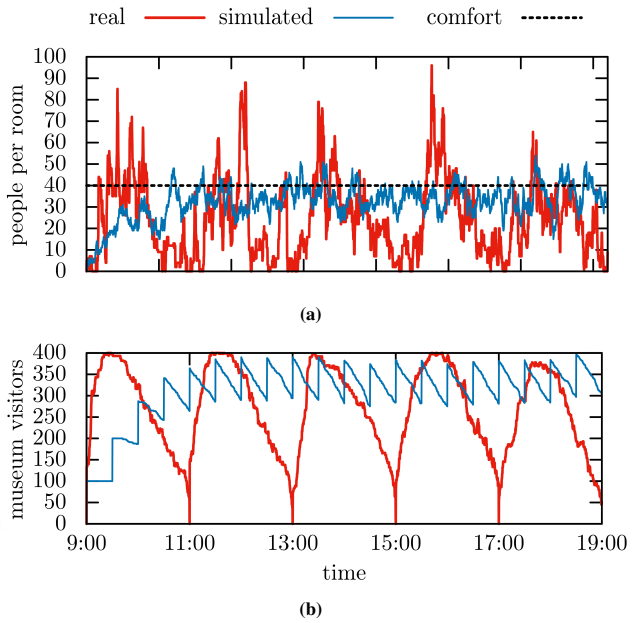


Fig. 4.36: PPR optimisation in the Galleria Borghese case study. The PPR is plotted as a function of time in the settings before our experimentation and considering the best entrance strategy. The comparison includes **(a)** room r_8 (*Caravaggio*) and **(b)** the whole museum.

4.8.3 Actual implementation of the proposed improvements

After the completion of our research, the museum curator decided to change the entrance strategy in light of the proposed optimisation by scheduling entrances on an hourly scheme, hence allowing 200 visitors per hour compared to 360+30 visitors per two-hour slot. In order to avoid overcrowding, the curator decided to let 50% of the visitors entering from E_1, then 20% from E_3 and, finally, the remaining part from E_1 again (this also complained with the restriction imposed during the sanitary emergency caused by Covid-19 pandemic).

We stress that the most important impact consists in the removal of the inter-time slot procedure that forced visitors to exit. However, the curator decided to keep the visit time limit of two hours, hence causing time slots to overlap. To be able to distinguish between different time slots, removable coloured stamps are currently given to visitors (to be stuck on their clothes) at the beginning of each visit slot. In order to provide the visitor with the indication that their time slot is over, the curator also introduced a sound message to remind the end of each time slot. Finally, it is important to notice that the novel entrance system allows visitors to enter also one hour before the museum closing time; hence, the curator introduced tickets with lowered prices for this last time slot.

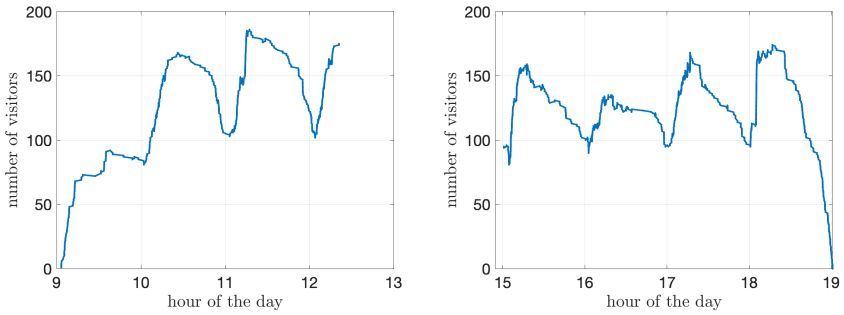


Fig. 4.37: Museum occupancy after the rescheduling of the entrance system where 200 visitors enter each hour. Data depicted are from two distinct measurement days, **(a)** a morning from 9 to 12:30 and **(b)** an afternoon from 15 to 19 highlights the behaviours at museum opening and closing time.

We decided to study the behaviours after these changes, by performing a novel gathering of Eulerian data. Unfortunately, such dataset was gathered during the sanitary emergency, hence with a limited number of tickets sold, *i.e.* 120 *w.r.t.* the current 200 per time slot. Results in terms of museum occupancy are reported in Figure 4.37 and should be compared with Figure 4.36**(b)** (with a suitable rescaling of the y-axis). Figure 4.37 **(a)**, **(b)** respectively highlights two main important features: (i) the museum never empties and (ii) the total number of visitors is (soft) bounded by 170.

Finally, in order to provide insights on how many visitors do follow the end-visit indication, *i.e.* the two-hours limit, we gathered the Eulerian measure of visitors inside the museum divided by turn colours (cf. Figure 4.9**(c)**). Figure 4.38 depicts a histogram of the visitor exit time per colour, hence showing that most of the visitors end their visit within 20 minutes after the real turn end.

4.9 Museum control and optimisation in the Peggy Guggenheim Collection

The initial setting in the Peggy Guggenheim Collection was very different from the one in Galleria Borghese. In fact, curators already employed an optimal ticketing system with no fixed time slots (actually a time slot per 10 minutes intervall, but without hard constraints on the tickets sold per time slot).

However, Eulerian and Lagrangian measurements highlighted a number of interesting criticalities worth of attention:

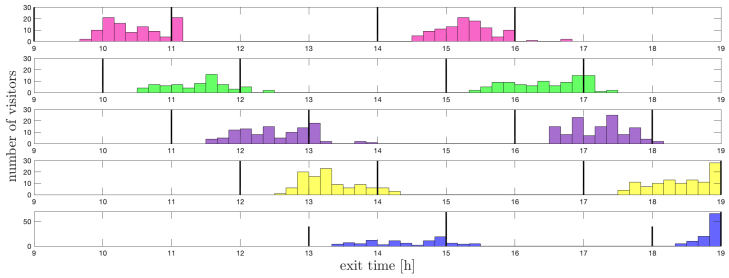


Fig. 4.38: Visitors' exit time with respect to their entrance time after rescheduling the entrance system. Most of the visitor (as expected) ends their visit before the "soft" time limit of 2h. A little number of visitors continue visiting the museum after two hours and 20 minutes, actually overlapping with two other time slots. Corresponding cumulative presence inside the museum is reported in Figure 4.37.

1. The Calder Hall (r_9), despite being an entrance room, also holds important artworks. However, visitors typically ignore it the first time they enter the museum and they rather spend more time there on their second passage by the room.
2. Visitors entering the main building from Calder Hall (r_9) splits equally between the two wings of the museum, hence without highlighting a preferred path of visit.
3. A direct consequence of the previous point is that Calder Hall (r_9) experiences a heavy flux cross between visitors entering the building from the garden (r_{11}), from the terrace (r_{14}) and from the two wings of the museum. Hence the room is hardly enjoyable, in particular during the most crowded hours of the day.
4. The number of visitors who enters the main building through the Schulhof collection (r_{10}) is $\sim 10\%$.
5. Conversely, $\sim 8\%$ of the visitors do not visit the Schulhof collection (r_{10}) at all.
6. The ticket office (r_{15}) experiences a flow cross between people entering the museum and people exiting the museum.
7. Only $\sim 80\%$ of the visitors visit (partially or entirely) the temporary exhibition.
8. Upon entrance, visitors typically seek restrooms that are mainly located at the museum exit and in the bar area (r_{12}).

All of the aforementioned remarks are likely ascribable to a lack of signage inside the museum. A visitor entering the garden along the obliged path from the ticket office first, and entering the Calder Hall then, is in fact leaved "on its own" to find a suitable visiting path. Our proposed optimisation was then to install a suggested path, *e.g.* by means of suitable signage, in order to guide visitors in a preferential direction.

The suggested path, also depicted in Figure 4.39, solve most of the above-mentioned issues. In fact, entering from the south part of the garden makes it natural to visit first the temporary exhibition and later the permanent exhibition, also creating an expectation *crescendo* during the visit. Furthermore, visiting the left wing

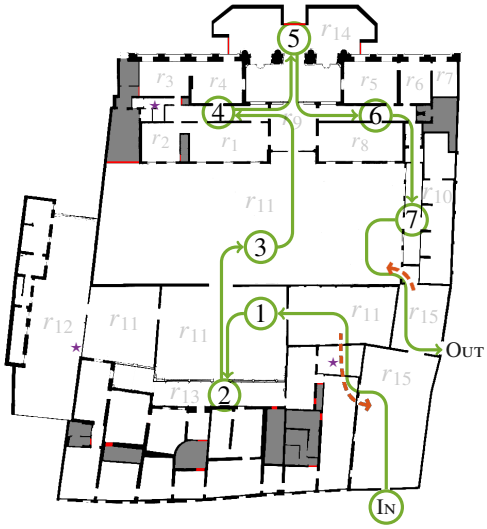


Fig. 4.39: Suggested visit path in the Peggy Guggenheim Collection that avoids fluxes crossing and potentially overcome the criticalities arisen during the experimentation. In particular, visitors enter the museum from the ticket office directly in the south part of the garden. This gives them direct access to restrooms (which are here highlighted with a purple star) and potentially to the bar. Then visitors move to the temporary exhibition and/or the veranda. Visitors continue the visit through the main part of the garden (and eventually the bar) and then enter the main exhibition area (*i.e.* permanent exhibition) from Calder Hall. The permanent exhibition is visited from left to right, with an intermediate stop on the terrace. Closes the visit the Schulhof collection and then the visitors are free to further explore the garden, go to the bar, or move to the exit.

of the main exhibition first allows visitors to be on the correct side of the museum to experience the Schulhof collection. Clearly, most of the flux crossings are hence avoided.

In particular, we partially implemented the proposed solution by adding signage to Calder Hall by means of a simple printed arrow visible from the garden entrance and pointing to the left wing (see Figure 4.40). We gathered a novel Eulerian dataset to measure the *first choice ratio* between left and right wing, *i.e.* we counted how many visitors chose the left wing against the right wing on their first visit to Calder Hall. The result of the measurement, surprisingly, showed that, also with the sole addition of a non-refined printed arrow, the choice ratio changed from $\sim$ 48% to $\sim$ 73%. This further suggests that, with more elaborated signage, better results are expected.

In light of the preliminary results obtained, the museum curators decided to accept the proposed suggestions and scheduled (at the time of writing) some renovations to make it possible to invert the one-way entrance/exit from/to the ticket office. Furthermore, a custom signage to highlight the suggested path from Figure 4.39 is also under construction.

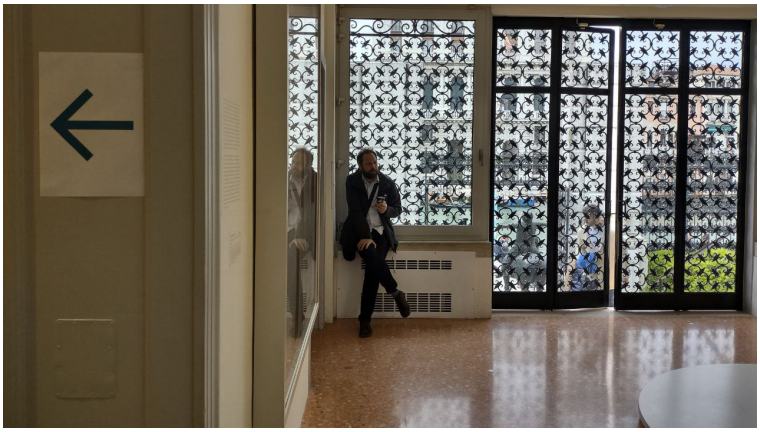


Fig. 4.40: Preliminary implementation of signage in Calder Hall (r_9, Peggy Guggenheim Collection) by means of a printed arrow (on the left of the picture). In the picture background, an operator is busy, manually counting the visitors' first choice between left and right wing with the help of the developed smartphone application (cf. Section 4.4.2, Figure 4.9).

Finally, thanks to the time of permanence measurements carried on single artworks, we were able to identify a particularly ill-fated piece of art: "the Empire of Light" one of the most known paintings by René Magritte. In particular, Magritte's was moved from r_8 (one of the largest) to r_6 (one of the smallest) during a temporary rearrangement and the time of permanence in front of it dropped significantly. While in the original position visitor spent 53s (on average, with standard deviation of 50s), once positioned in room r_6, the corresponding permanence time lowered to 38s (on average, with standard deviation of 37s). In particular, an unexpected behaviour worth of attention is the time spent by visitors with an audio guide which dropped by 27s (from 102s to 75s, on average); this result is particularly interesting since, clearly, registered descriptions on the audio guide last always the same time being room-agnostic, hence one would expect that visitors would spend the same amount of time in front of the piece of art while listening to it. Conversely, the data highlighted that locating the artwork in a position of minor relevance (*i.e.*, not a principal room) encourages visitors to stop the audio guide reproduction and move on to different artworks.

4.10 Conclusions and future work

This study aimed at measuring, analysing, modelling, and optimising visitors' behaviour in museums and similar environments. The practical goal was to provide suggestions to museum curators for efficiently managing visitors' flows.

The proposed Eulerian measurement approach is simple and easy to implement, simply requiring manpower that can easily be gathered by the museum *e.g.* by means

of interns. The results of the measurements are easily interpretable and compose an affordable way to verify the effectiveness of any change applied during the experimentation.

The implemented Lagrangian measurement system is sustainable for the museum, being economically viable and well-accepted by visitors. A free application to be installed on the smartphone could serve as a beacon as well, provided visitors find it useful (as an audio guide, for example). Employing Raspberry Pis as fixed Bluetooth antennas appeared quite convenient and allowed the necessary development flexibility.

A major issue surely comes from the noisiness of the Bluetooth signal, which must be overcome by suitable data post-processing. In this sense, the total-coloured graph interpretation of the museum (which is novel and interesting *per se*) allows the injection of useful expert-knowledge pieces of information as well as physical constraints in the representation of the museum. Such representation is found to be useful in both metric definition and trajectory reconstruction, allowing the design of a cascaded selector based on simple localisers that yields precise trajectory reconstruction.

From the trajectory analysis, we have identified some issues in the museum design and visit experience that can be considered by curators: for example, rooms of the same size have drastically different times of permanence, as it happens *e.g.* for *Caravaggio* and *Satiro su Delfino* in the Galleria Borghese case study. This suggests that the museum can benefit from a rearrangement of the artworks, although this is not always possible due to historical or architectural constraints.

The museum simulator allowed us to propose the implementation of a new ticketing and entrance system, further validate by the post-change Eulerian measurements.

In the next future, we plan to further improve the model presented here. In particular, we aim at including the internal dynamics of *social groups* (families, friends, guided tours), and at considering the impact of congestion on individual behaviour. This is to lift the current statistical independence of simulated trajectories, thus increasing the level of complexity.

The impact of visitors on the local microclimate is also an outstanding issue to which we aim (cf. [139]). A model is currently under development in collaboration with the ISPC-CNR (Istituto di Scienze del Patrimonio Culturale), which provides us with temperature, humidity, CO2, ammonia, and other pollutant data in the Peggy Guggenheim Collection, and will be the object of a future paper [284].

Finally, worth of attention is also the possibility to enrich the model by allowing possible displacement changes of some key artwork that might change the fruition of the exhibition space.

References

[87] R. K. Ahuja, K. Mehlhorn, J. Orlin, and R. E. Tarjan. "Faster algorithms for the shortest path problem". In: *Journal of the ACM (JACM)* 37.2 (1990), pp. 213–223. DOI: 10.1145/77600.77615 (cit. on p. 116).

[88] C. Balzotti, M. Briani, A. Corbetta, E. Cristiani, M. Minozzi, R. Natalini, S. Suriano, and F. Toschi. "Forecasting visitors behaviour in crowded museums". In: *Proceedings of the 9th International Conference on Pedestrian and Evacuation Dynamics (PED2018), Lund, Sweden, August 22-24, 2018*. 2018. DOI: 10.17815/CD.2020.82 (cit. on pp. 105, 108).

[89] C. Beder and M. Klepal. "Fingerprinting based localisation revisited. A rigorous approach for comparing RSSI measurements coping with missed access points and differing antenna attenuations". In: *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN 2012), November 13-15, 2012*. 2012. DOI: 10.1109/IPIN.2012.6418940 (cit. on pp. 106, 120).

[90] N. Bellomo and C. Dogbé. "On the modeling of traffic and crowds: a survey of models, speculations, and perspectives". In: *SIAM Rev.* 53.3 (2011), pp. 409–463. DOI: 10.1137/090746677 (cit. on p. 107).

[91] G. Biau and E. Scornet. "A random forest guided tour". In: *TEST* 25.2 (Apr. 2016), pp. 197–227. DOI: 10.1007/s11749-016-0481-7 (cit. on p. 135).

[92] A. Bollo and L. Dal Pozzolo. "Analysis of visitor behaviour inside the museum: An empirical study". Unpublished. 2005. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=5ceca23cba50c06bca4c300b02cac32595732 0f3 (cit. on p. 105).

[93] L. Bourdeau and J.-C. Chebat. "An empirical study of the effects of the design of the display galleries of an art gallery on the movement of visitors". In: *Museum Management and Curatorship* 19.1 (2001), pp. 63–73. DOI: 10.1080/09647770100601901 (cit. on pp. 104, 107).

[94] D. Camuffo. *Microclimate for cultural heritage*. Elsevier Science, 1998 (cit. on p. 157).

[95] G. Casolla, S. Cuomo, V. Schiano di Cola, and F. Piccialli. "Exploring unsupervised learning techniques for the Internet of Things". In: *IEEE Transactions on Industrial Informatics* 16.4 (2020), pp. 2621–2628. DOI: 10.1109/TII.2019.2941142 (cit. on pp. 106, 107, 148).

[96] A. Chianese and F. Piccialli. "Designing a Smart Museum: When Cultural Heritage Joins IoT". In: *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*. 2014, pp. 300–306. DOI: 10.1109/NGMAST.2014.21 (cit. on p. 104).

[97] H.-s. Choi and S.-h. Kim. "A content service deployment plan for metaverse museum exhibitions – Centering on the combination of beacons and HMDs, keywords = managing-crowded-museums". In: *International Journal of Information Management* 37 (2017), pp. 1519–1527. DOI: 10.1016/j.ijinfomgt.2016.04.017 (cit. on p. 106).

[98] A. Corbetta, J. A. Meeusen, C.-m. Lee, R. Benzi, and F. Toschi. "Physics-based modeling and data representation of pairwise interactions among pedestrians". In: *Physical Review E* 98 (2018), pp. 062310/1–16. DOI: 10.1103/PhysRevE.98.062310 (cit. on p. 106).

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. 2nd. ISBN: 978-0-262-03293-3. The MIT Press, 2001. ISBN: 0262032937 (cit. on pp. 3, 11, 16, 19, 21, 117).

[99] E. Cristiani, B. Piccoli, and A. Tosin. *Multiscale modeling of pedestrian dynamics*. Modeling, Simulation & Applications, keywords = managing-crowded-museums. Springer, 2014 (cit. on p. 107).

[100] S. Cuomo, P. De Michele, M. Pragliola, and G. Severino. "Mimic visiting styles by using a statistical approach in a cultural event case study". In: *Procedia Computer Science* 98 (2016). International Workshop on Data Mining on IoT Systems (DaMIS16), pp. 449–454. DOI: 10.1016/j.procs.2016.09.071 (cit. on p. 105).

[101] G. Deak, K. Curran, and J. Condell. "A survey of active and passive indoor localisation systems". In: *Computer Communications* 35.16 (2012), pp. 1939–1954. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2012.06.004 (cit. on p. 106).

[102] M. Delafontaine, M. Versichele, T. Neutens, and N. Van de Weghe. "Analysing spatiotemporal sequences in Bluetooth tracking data". In: *Applied Geography* 34 (2012), pp. 659–668. DOI: 10.1016/j.apgeog.2012.04.003 (cit. on pp. 106, 107).

[103] E. Dim and T. Kuflik. "Automatic detection of social behavior of museum visitor pairs". In: *ACM Transactions on Interactive Intelligent Systems* 4.4 (2014), 17:1–30. DOI: 10.1145/2662869 (cit. on p. 107).

[104] H. Dong, M. Zhou, Q. Wang, X. Yang, and F.-Y. Wang. "State-of-the-art pedestrian and evacuation dynamics". In: *IEEE Transactions on Intelligent Transportation Systems* 21.5 (2020), pp. 1849–1866. DOI: 10.1109/TITS.2019.2915014 (cit. on p. 107).

[105] D. C. Duives, W. Daamen, and S. P. Hoogendoorn. "State-of-the-art crowd motion simulation models". In: *Transportation Res. C* 37 (2013), pp. 193–209. DOI: 10.1016/j.trc.2013.02.005 (cit. on p. 107).

[106] R. Eftimie. "Multi-dimensional transport equations". In: *Hyperbolic and Kinetic Models for Self-organised Biological Aggregations, Lecture Notes in Mathematics (LNMBIOS)*. Vol. 2232. Springer, Cham, 2018, pp. 153–193. ISBN: 978-3-030-02586-1. DOI: 10.1007/978-3-030-02586-1_6 (cit. on p. 107).

[107] J. H. Falk. *Identity and the museum visitor experience*. Routledge, Taylor & Francis Group, London and New York, 2016 (cit. on p. 104).

[108] Y. Feng, D. Duives, W. Daamen, and S. Hoogendoorn. "Data collection methods for studying pedestrian behaviour: A systematic review". In: *Building and Environment* 187 (2021), p. 107329. ISSN: 0360-1323. DOI: 10.1016/j.buildenv.2020.107329 (cit. on p. 104).

[109] S. Georgievska, P. Rutten, J. Amoraal, E. Ranguelova, R. Bakhshi, B. L. de Vries, M. Lees, and S. Klous. "Detecting high indoor crowd density with Wi-Fi localization: A statistical mechanics approach". In: *Journal of Big Data* 6 (2019), 31:1–23. DOI: 10.1186/s40537-019-0194-3 (cit. on p. 106).

[40] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org (cit. on pp. 39, 135).

[110] Y. Gu, A. Lo, and I. Niemegeers. "A Survey of indoor positioning systems for wireless personal networks". In: *IEEE Communications Surveys & Tutorials* 11.1 (2009), pp. 13–32. DOI: 10.1109/SURV.2009.090103 (cit. on p. 106).

[111] K. Guler. "A simulation application for visitor circulation in exhibition environments". PhD thesis. Bilkent University, Turkey, 2009. URL: https://core.ac.uk/download/pdf/52925259.pdf (cit. on pp. 107, 109).

[112] K. Guler. *Simulating visitor behavior*. Cambridge Scholar Publishing, 2016 (cit. on pp. 107, 109, 123).

[113] M. Haghani. "Optimising crowd evacuations: Mathematical, architectural and behavioural approaches". In: *Safety Science* 128 (2020), pp. 104745/1–23. DOI: 10.1016/j.ssci.2020.104745 (cit. on p. 107).

[114] H. Hong, G. D. De Silva, and M. C. Chan. "CrowdProbe: Non-invasive crowd monitoring with WiFi probe". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2018, 115:1–23. DOI: 10.1145/3264925 (cit. on pp. 106, 107).

[115] E. Iadanza. "Designing an Indoor Real-Time Location System for Healthcare Facilities". In: *Mediterranean Forum–Data Science Conference: First International Conference, MeFDATA 2020, Sarajevo, Bosnia and Herzegovina, October 24, 2020, Revised Selected Papers*. Vol. 1343. Springer Nature. 2021, p. 110. DOI: 10.1007/978-3-030-72805-2_8 (cit. on p. 106).

[116] V. Kirchberg and M. Tröndle. "The museum experience: Mapping the experience of fine art". In: *Curator: The Museum Journal* 58.2 (2015), pp. 169–193. DOI: 10.1111/cura.12106 (cit. on pp. 104, 106).

[117] H.-J. Klein. "Tracking visitor circulation in museum settings". In: *Environment and Behavior (Sage journals)* 25.6 (1993), pp. 782–800. DOI: 10.1177/0013916593256007 (cit. on p. 105).

[118] T. Kuflik, Z. Boger, and M. Zancanaro. "Analysis and prediction of museum visitors' behavioral pattern types". In: *Ubiquitous Display Environments*. Ed. by A. Kr uger and T. Kuflik. Cognitive Technologies. Springer-Verlag Berlin Heidelberg, 2012, pp. 161–176. DOI: 10.1007/978-3-642-27663-7_10 (cit. on pp. 105, 107).

[119] J. Lanir, T. Kuflik, E. Dim, A. J. Wecker, and O. Stock. "The Influence of a location-aware mobile guide on museum visitors' behavior". In: *Interacting with Computers* 25.6 (2013), pp. 443–460. DOI: 10.1093/iwc/iwt002 (cit. on p. 107).

[120] J. Lanir, T. Kuflik, J. Sheidin, N. Yavin, K. Leiderman, and M. Segal. "Visualizing museum visitors' behavior: Where do they go and what do they

do there?" In: *Personal and Ubiquitous Computing* 21 (2017), pp. 313–326. DOI: 10.1007/s00779-016-0994-9 (cit. on pp. 104–107).

[121] V. I. Levenshtein et al. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Vol. 10. 8. Soviet Union. 1966, pp. 707–710. URL: https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf (cit. on p. 148).

[122] M. Liakou and O. Kosmas. "Modelling and simulation of pedestrian behaviour on museum exhibition spaces". In: *Strategic Innovative Marketing. IC-SIM 2017*. Ed. by D. P. Sakas and D. K. Nasiopoulos. Springer Proceedings in Business and Economics. Springer, Cham, 2019. DOI: 10.1007/978-3-030-16099-9_29 (cit. on p. 107).

[123] R. Lovreglio, D. C., S. X., and B. L. "Investigating pedestrian navigation in indoor open space environments using big data". In: *Applied Mathematical Modelling* 62 (2018), pp. 499–509. DOI: 10.1016/j.apm.2018.06.014 (cit. on p. 106).

[124] A. Mackey, P. Spachos, L. Song, and K. N. Plataniotis. "Improving BLE Beacon Proximity Estimation Accuracy Through Bayesian Filtering". In: *IEEE Internet of Things Journal* 7.4 (2020), pp. 3160–3169. DOI: 10.1109/JIOT.2020.2965583 (cit. on p. 106).

[125] C. Martella, A. Miraglia, J. Frost, M. Cattani, and M. van Steen. "Visualizing, clustering, and predicting the behavior of museum visitors". In: *Pervasive and Mobile Computing* 38 (2017). Special Issue IEEE International Conference on Pervasive Computing and Communications (PerCom) 2016, pp. 430–443. DOI: 10.1016/j.pmcj.2016.08.011 (cit. on pp. 105–107).

[126] F. Martinez-Gil, M. Lozano, I. García-Fernández, and F. Fernández. "Modeling, evaluation, and scale on artificial pedestrians: A literature review". In: *ACM Comput. Surv.* 50.5 (2017), pp. 72/1–35. DOI: 10.1145/3117808 (cit. on p. 107).

[127] M. Mokatren, V. Bogina, A. Wecker, and T. Kuflik. "A Museum visitors classification based on behavioral and demographic features". In: *Proceedings of Personalized access to Cultural Heritage (PATCH 2019) Workshop, UMAP'19 Adjunct, June 9-12, 2019, Larnaca, Cyprus*. 2019, pp. 383–386. DOI: 10.1145/3314183.3323864 (cit. on pp. 105, 107).

[128] N. Newman. "Apple iBeacon technology briefing". In: *Journal of Direct, Data and Digital Marketing Practice* 15 (2014), pp. 222–225. DOI: 10.1057/dddmp.2014.7 (cit. on p. 119).

[129] D. Oosterlinck, D. F. Benoit, P. Baecke, and N. Van de Weghe. "Bluetooth tracking of humans in an indoor environment: An application to shopping mall visits". In: *Applied geography* 78 (2017), pp. 55–65. DOI: 10.1016/j.apgeog.2016.11.005 (cit. on pp. 106, 107).

[130] F. Piccialli, P. Benedusi, L. Carratore, and G. Colecchia. "An IoT data analytics approach for cultural heritage". In: *Personal and Ubiquitous Computing* 24 (2020), pp. 429–436. DOI: 10.1007/s00779-019-01323-z (cit. on p. 104).

[131] F. Piccialli, S. Cuomo, F. Giampaolo, G. Casolla, and V. Schiano di Cola. "Path prediction in IoT systems through Markov Chain algorithm". In: *Future Generation Computer Systems* 109 (2020), pp. 210–217. DOI: 10.1016/j.future.2020.03.053 (cit. on pp. 106, 107).

[132] F. Piccialli, S. Cuomo, V. Schiano di Cola, and G. Casolla. "A machine learning approach for IoT cultural data". In: *Journal of Ambient Intelligence and Humanized Computing* (2019). DOI: 10.1007/s12652-019-01452-6 (cit. on pp. 106, 107, 148).

[133] F. Piccialli, Y. Yoshimura, P. Benedusi, C. Ratti, and S. Cuomo. "Lessons learned from longitudinal modeling of mobile-equipped visitors in a complex museum". In: *Neural Computing and Applications* (2019). DOI: 10.1007/s00521-019-04099-8 (cit. on pp. 106, 107).

[134] R. Pierdicca, M. Marques-Pita, M. Paolanti, and E. S. Malinverni. "IoT and engagement in the ubiquitous museum". In: *Sensors* 19 (2019), 1387:1–21. DOI: 10.3390/s19061387 (cit. on p. 106).

[135] A. Pluchino, C. Garofalo, G. Inturri, A. Rapisarda, and M. Ignaccolo. "Agent-based simulation of pedestrian behaviour in closed spaces: A museum case study". In: *Journal of Artificial Societies and Social Simulation* 17.1 (2014), p. 16. DOI: 10.18564/jasss.2336 (cit. on p. 108).

[136] C. Pouw, J. Willems, F. van Schadewijk, J. Thurau, F. Toschi, and A. Corbetta. "Benchmarking High-Fidelity Pedestrian Tracking Systems for Research, Real-Time Monitoring and Crowd Control". In: *Collective Dynamics* 6 (Jan. 2022), pp. 1–22. DOI: 10.17815/CD.2021.134 (cit. on p. 104).

[137] E. S. Robinson. *The behavior of the museum visitor*. New Series, Number 5. Washington, D. C.: Publications of the American Association of Museums, 1928. URL: https://files.eric.ed.gov/fulltext/ED044919.pdf (cit. on p. 104).

[138] S. Sadowski and P. Spachos. "RSSI-Based Indoor Localization With the Internet of Things". In: *IEEE Access* 6 (2018), pp. 30149–30161. DOI: 10.1109/ACCESS.2018.2843325 (cit. on p. 106).

[139] L. de Santoli, F. Mancini, S. Rossetti, and B. Nastasi. "Energy and system renovation plan for Galleria Borghese, Rome". In: *Energy and Buildings* 129 (2016), pp. 549–562. DOI: 10.1016/j.enbuild.2016.08.030 (cit. on p. 166).

[140] S. Seer, N. Brändle, and C. Ratti. "Kinects and human kinetics: A new approach for studying pedestrian behavior". In: *Transportation Research Part C: Emerging Technologies* 48 (2014), pp. 212–228. DOI: 10.1016/j.trc.2014.08.012 (cit. on p. 106).

[141] M. Tröndle, S. Greenwood, K. Bitterli, and K. van den Berg. "The effects of curatorial arrangements". In: *Museum Management and Curatorship* 29.2 (2014), pp. 140–173. DOI: 10.1080/09647775.2014.888820 (cit. on pp. 104, 105, 107, 121).

[142] M. Tröndle, S. Greenwood, V. Kirchberg, and W. Tschacher. "An integrative and comprehensive methodology for studying aesthetic experience in the field: Merging movement tracking, physiology, and psychological data".

In: *Environment and Behavior* 46.1 (2014), pp. 102–135. DOI: 10.1177/0013916512453839 (cit. on pp. 104, 106).

[143] K. Tzortzi. "Movement in museums: Mediating between museum intent and visitor experience". In: *Museum Management and Curatorship* 29.4 (2014), pp. 327–348. DOI: 10.1080/09647775.2014.939844 (cit. on p. 104).

[144] P. S. Varma and V. Anand. "Random Forest Learning Based Indoor Localization as an IoT Service for Smart Buildings". In: *Wireless Personal Communications* 117.4 (2021), pp. 3209–3227. DOI: 10.1007/s11277-020-07977-w (cit. on p. 106).

[145] E. Véron and M. Levasseur. *Ethnographie de l'exposition: L'espace, le corps et le sens*. Bibliothéque publique d'information, Centre Georges Pompidou, 1989 (cit. on p. 107).

[146] M. Versichele, T. Neutens, M. Delafontaine, and N. Van de Weghe. "The use of Bluetooth for analysing spatiotemporal dynamics of human movement at mass events: A case study of the Ghent Festivities". In: *Applied Geography* 32 (2012), pp. 208–220. DOI: 10.1016/j.apgeog.2011.05.011 (cit. on p. 106).

[147] V. Viswanathan, M. Lees, and P. M. A. Sloot. "The influence of memory on indoor environment exploration: A numerical study". In: *Behav. Res.* 48 (2016), pp. 621–639. DOI: 10.3758/s13428-015-0604-1 (cit. on p. 107).

[148] X. Wang, O. Bischoff, R. Laur, and S. Paul. "Localization in wireless ad-hoc sensor networks using multilateration with RSSI for logistic applications". In: *Procedia Chemistry* 1.1 (2009), pp. 461–464. DOI: 10.1109/GLOCOM.2009.5425237 (cit. on p. 106).

[149] W. Xue, W. Qiu, X. Hua, and K. Yu. "Improved Wi-Fi RSSI measurement for indoor localization". In: *IEEE Sensors Journal* 17.7 (2017), pp. 2224–2230. DOI: 10.1109/JSEN.2017.2660522 (cit. on p. 106).

[150] S. S. Yalowitz and K. Bronnenkant. "Timing and tracking: Unlocking visitor behavior". In: *Visitor Studies* 12.1 (2009), pp. 47–64. DOI: 10.1080/10645570902769134 (cit. on p. 104).

[151] B. Yang, L. Guo, R. Guo, M. Zhao, and T. Zhao. "A Novel Trilateration Algorithm for RSSI-Based Indoor Localization". In: *IEEE Sensors Journal* 20.14 (2020), pp. 8164–8172. DOI: 10.1109/JSEN.2020.2980966 (cit. on p. 106).

[152] Y. Yoshimura, F. Girardin, J. P. Carrascal, R. C., and B. J. "New tools for studying visitor behaviours in museums: A case study at the Louvre". In: *Information and Communication Technologies in Tourism 2012*. Ed. by F. M., R. F., and C. L. Proceedings of the International Conference in Helsingborg, Sweden, January 25–27, 2012. Springer, Vienna, 2012. DOI: 10.1007/978-3-7091-1142-0_34 (cit. on p. 106).

[153] Y. Yoshimura, R. Sinatra, A. Krebs, and C. Ratti. "Analysis of visitors' mobility patterns through random walk in the Louvre Museum". In: *Journal of Ambient Intelligence and Humanized Computing* (2019). DOI: 10.1007/s12652-019-01428-6 (cit. on p. 107).

[154] Y. Yoshimura, S. Sobolevsky, C. Ratti, F. Girardin, J. P. Carrascal, J. Blat, and R. Sinatra. "An analysis of visitors' behavior in the Louvre Museum: A study using Bluetooth data". In: *Environment and Planning B: Planning and Design* 41 (2014), pp. 1113–1131. DOI: 10.1068/b130047p (cit. on pp. 104, 106, 108, 121).

[155] M. Zancanaro, T. Kuflik, Z. Boger, D. Goren-Bar, and D. Goldwasser. "Analyzing museum visitors' behavior patterns". In: *User Modeling 2007, LNAI 4511*. Ed. by C. Conati, K. McCoy, and G. Paliouras. 11th International Conference, UM 2007, Corfu, Greece, June 25-29, 2007. Springer-Verlag Berlin Heidelberg, 2007, pp. 238–246. DOI: 10.1007/978-3-540-73078-1_27 (cit. on pp. 106, 107).

[156] C. Zhou, J. Yuan, H. Liu, and J. Qiu. "Bluetooth indoor positioning based on RSSI and Kalman filter". In: *Wireless Personal Communications* 96.3 (2017), pp. 4115–4130. DOI: 10.1007/s11277-017-4371-4 (cit. on p. 106).

Chapter 5
Hybrid approach for traffic state estimation and forecast

This chapter introduces a novel hybrid approach in traffic modelling that exploits machine learning to enrich data fed into the numerical model. The work presented here is currently under submission [282] and it was realised in collaboration with Maya Briani and Emiliano Cristiani, researchers at the "Istituto per le Applicazioni del Calcolo" *of the* "Consiglio Nazionale delle Ricerche" *(IAC-CNR).*

Abstract In this chapter, we aim to develop new methods that integrate machine learning techniques with macroscopic differential models for vehicular traffic estimation and forecast. It is well known that data-driven and model-driven approaches have (sometimes complementary) advantages and drawbacks. We consider here a dataset comprising vehicle flux and velocity data collected by fixed sensors on a highway. The data is classified by lane and vehicle class. By means of a machine learning model based on an LSTM recursive neural network, we extrapolate two important pieces of information. Firstly, we detect congestion under the sensor, and secondly, we forecast the total number of vehicles passing under the sensor within the next future (30 minutes). These extracted pieces of information are then used to enhance the accuracy of an LWR-based first-order multi-class model that describes the dynamics of traffic flow between sensors. The first piece of information is used to invert the (concave) fundamental diagram, thus recovering the density of vehicles from the flux data. Such density data is directly injected into the model, resulting in improved approximations of the dynamics between sensors, particularly in scenarios involving accidents in unmonitored road sections. The second piece of information serves as boundary conditions for the equations underlying the traffic model, to better reconstruct the total amount of vehicles on the road at any future time. To illustrate the effectiveness of our approach, we present and discuss examples from real-world scenarios. Real data are provided by the Italian motorway company Autovie Venete S.p.A.

Keywords: traffic · vehicles · fundamental diagram · LWR model · machine learning · LSTM

2020 MSC: 76A30 · 68T07

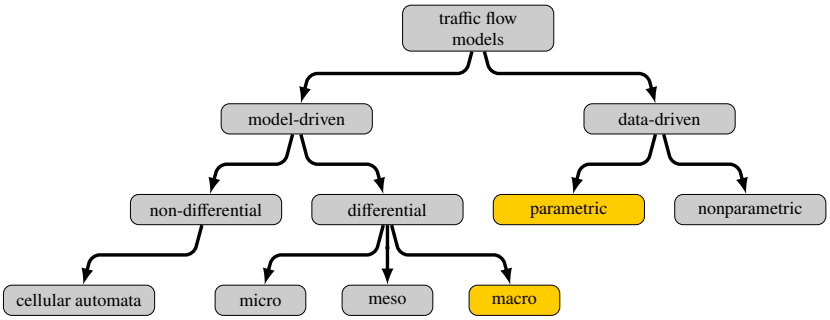


Fig. 5.1: Structure of the research lines in vehicular traffic flow modelling. Our approach builds upon the highlighted leaves.

5.1 Introduction

Traffic state estimation (TSE) and traffic forecast have a long and solid tradition which dates back to the 1950s. A very broad division of the research lines on vehicular traffic flow modelling is summarised in Figure 5.1. Model-driven and data-driven approaches have their own advantages and drawbacks, which were well described in, *e.g.*, [199, 208]: Model-driven approaches allow to inject in the simulator the human knowledge of the system, at least if it can be reasonably translated into equations. Differences between agents can be (stochastically) taken into account as well, including drivers' psychological aspects. Differential macroscopic models, in particular, can unveil the power of methodologies based on partial differential equations (PDEs), for example giving the right tools to compute the *Wardrop equilibrium* of a traffic system on a road network [165, 170, 206]. On the other hand, this approach tends to be an oversimplification of traffic physics since the model is never able to catch *all* the features of cars and drivers. Most models are difficult to work with noisy and fluctuated data collected by traffic sensors and the calibration of the numerous parameters is quite challenging. In addition, the numerical scheme used for the discretisation of the equations (*e.g.*, Godunov, Lax-Friedrichs) introduces a further, often not negligible, approximation error. Finally, models require additional inputs which are not available in real scenarios, such as, *e.g.*, boundary conditions at any future time.

Data-driven approaches, instead, are more suitable to deal with the nonlinearities which characterise traffic flow and, for this reason, can be more accurate than model-driven approaches, but they are agnostic to the physics of traffic flow and could lead to infeasible estimation results. These methods are also not often interpretable and lack robustness. More importantly, the generalisability of the models is often weak and they have a high dependence on the training data samples. If the quality of training data is poor (missing/overestimated/underestimated data), their predictive accuracy will be severely weakened. Recently, many machine learning (ML) approaches were proposed: they often rely on relatively simple structures (compared to PDEs intricate

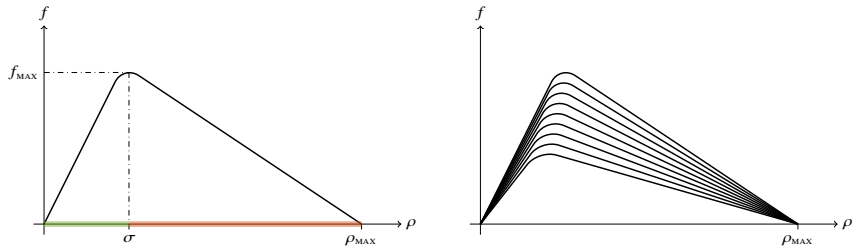


Fig. 5.2: (a) a typical single-valued fundamental diagram $f = f(\rho)$. The green part corresponds to the *free phase* while the orange part corresponds to the *congested phase*. Density σ corresponds to the maximal flux f_{MAX} (road capacity). Flux is null when the road is empty ($\rho = 0$) and when the road is fully congested ($\rho = \rho_{MAX}$) and vehicles are stopped. **(b)** a possible multi-valued fundamental diagram. For any given density ρ a set of possible values for f can be chosen (between the lowest and the highest curve). The multiple choice comes from the heterogeneity of drivers' behaviour.

systems) making them more lightweight from a computational point of view, hence being more suitable for real-time applications. However, dependency on large sets of historical data means that training can be computationally very expensive and can easily lead to data overfitting.

Recent research is growing in interest toward *hybrid approaches* which try to get the best from each of the two approaches; however, they are rarer in the literature and each paper uses only single aspects from the two methodologies to obtain very different results. This chapter tries to advance in this research field by proposing a computational method where ML techniques extrapolate from data the information needed by differential macroscopic models for traffic flow (see highlighted nodes in Figure 5.1).

5.1.1 Relevant literature

Fundamental diagram

First of all, we have to mention the *fundamental diagram*, which is one of the basic ingredients of all model-driven approaches, especially at the macroscopic scale. It defines the relationship between the flux f and the density ρ of vehicles [176, 183]. It is plain that the flux of vehicles is null in either the case of an empty road (null density) or in the case of a fully congested road (maximal density, stopped bumper-to-bumper vehicles). For intermediate density levels, real data show complex dynamics, especially in correspondence to the maximal capacity of the road. Indeed, drivers act differently in response to the same traffic conditions and, in addition, accelerations and decelerations are far from being instantaneous. As a consequence, traffic shows some instabilities [163, 192, 202]. In first-order traffic models, the fundamental

diagram can be defined by means of a single function while second-order traffic models allow the fundamental diagram to be multivalued, in the sense that a single value of the density can be associated with many values of the flux, exactly as it happens in reality, see Figure 5.2. Recently, an interesting ML-based method to estimate the fundamental diagram was proposed in [197].

Model-driven

Mathematical models for traffic flow first appeared in 1955-6 with the seminal papers [186, 194], which introduced the well-known LWR model. It is a first-order (velocity-based) model constituted by a hyperbolic PDE where the observed quantity is the vehicle density ρ. The model stems from the reasonable analogy between vehicular dynamics and fluid dynamics. Following the same line, the model was successively extended to the second order to include inertia (acceleration) effects in [193, 207] and [158, 213], giving birth to the PW and ARZ models, respectively. Independently, in the engineering literature, the CTM model [171] and the METANET model [189] were introduced. These two last models are equivalent to the discretised versions of the LWR and PW model, respectively [205]. The literature about mathematical models is huge, we refer the reader to the surveys [160, 179, 195, 201, 205] for differential models and their calibration, and [187] for nondifferential models (cellular automata).

Some effort was also devoted to generalising mathematical models to road networks. This is not a trivial task due to the junction conditions which must be added to assure the uniqueness of the solution of the resulting system of PDEs. We refer the reader to [161, 162], and the book [177] for basic concepts.

Another generalisation of our interest is that of multi-class models. In this case, more than one class of vehicles (*e.g.*, cars and trucks) share the same road. Each class has specific dynamics and the classes interact with each other in a nontrivial manner. We refer to [157, 174, 200], and to the recent books [176, 183] for an overview of the most used multi-class models.

Data-driven

Traffic data can be collected by means of several methods and technologies. Commonly one can have Eulerian data, provided by fixed sensors placed along the road (which count passing vehicles), and Lagrangian data, provided by probe vehicles equipped, *e.g.*, with a GPS system. We refer to [166, 167, 209] for an overview on traffic data. Typical objects under analysis are, hence, flux f and velocity v, as opposed to the more abstract concept of density ρ that characterises the model-driven approaches.

Early works in data-driven methods mainly rely on classical statistical analysis of historical data. More recently, the broad research carried out in ML furnished new tools for data-driven methods.

In this chapter, we are mostly interested in Artificial Neural Network (ANN, see Section 2.3) methods for traffic understanding, estimation, and prediction, see [173, 191, 199, 210, 214, 216]. We recall that ANNs are mainly divided into two families: feed-forward models (FNN), like the Single- or Multi-Layer Perceptron (S/MLP) introduced in Section 2.3.2 and recurrent models (RNN) like LSTM, introduced in Section 2.3.3. We refer the reader to the recent surveys [175, 184, 188, 198, 204] on how these models are used in the context of traffic and, more specifically, we focus on the use of LSTM for traffic data forecast; in fact, we recall LSTMs proved to be capable of capturing long-range temporal feature dependencies and reducing gradient explosion/vanishing. Among the recent literature, LSTM were often used (both vanilla or as a building block of more complex structures) to perform analysis and prediction on f and v: for what concerns velocity, it is the case, *e.g.*, of [178] that mounts a fusion deep learning approach to predict lane-level traffic speed at two minutes, [181] that considers the correlation between car speed and car type for a prediction model (LSTM + 4 layer MLP) of the highway speed at 5 minutes, and [208] that combines LSTM with a careful data preprocessing aided with wavelets analysis to perform speed prediction at 15 minutes. For what concerns flux, it is the case, *e.g.*, of [215] where a temporal-spatial correlation is integrated into a 2D LSTM network to predict traffic flow at 15 minutes, [203] which integrates weather data with an attention model to perform short-term prediction of the traffic volume, and [175] which describes a novel methodology of an LSTM-based attention model to predict the upcoming flux based on 120 minutes of data (aggregated 10 minutes by 10 minutes).

Hybrid methods To overcome the shortcomings of both model- and data-driven approaches while exploiting their potentialities, recent studies introduced coupled methods where physics and data play together. The way the coupling is performed in the literature is very different since a common line is yet to be established: the physical model can be (i) injected in the training process of the ANN, obtaining the so-called Physics-Informed Neural Networks (PINNs), (ii) used in parallel with the ANN, as it is done, *e.g.*, in [197], where the TSE, the model parameter identification, and estimation of the fundamental diagram are performed simultaneously, or (iii) used *after* the ANN, like in the present work and in [180], where data are used to provide consistent boundary conditions at junctions for macroscopic traffic flow models.

The majority of the recent works fall back into the PINNs category, where physics is usually plugged into the model by building a custom cost function, in particular, trying to exploit the powerfulness of deep learning models (PIDL); it is the case, *e.g.*, of [196] where authors focus on highway TSE with observed data from loop detectors and probe vehicles, by building a coupled model with a Physics-Uninformed Neural Network (PUNN) and a PINN with custom loss function based on physical discrepancy measures. Paper [182], analogously, builds a custom loss function relying on CTM and LWR with different fundamental diagrams (Greenshields', Daganzo's, and inverse-lambda) to tackle the problem of data sparsity and sensor noise. Another example of custom cost function based on multiple physical aspects is given in [159], where authors perform TSE from probe vehicles data in an urban environ-

ment by building a 6-component loss function that is used both to reconstruct the road state and a smoothed version of the probe trajectories. Also worth mentioning is the paper [190], where authors introduce PIDL car-following model architectures encoded with different popular physics-based models to predict the evolution of the velocity of each vehicle. Finally, it is also interesting the approach based on physics regularised Gaussian process (PRGP), like the one proposed in [212] (later extended in [211]) where a stochastic PRGP is developed and a Bayesian inference algorithm is used to estimate the mean and kernel of the PRGP itself.

5.1.2 Chapter contribution

In this chapter, we deal with traffic data coming from a series of fixed sensors placed along a highway. Sensors are able to count vehicles passing under them, estimate vehicles velocity, and classify vehicles in terms of their length (dividing them, *e.g.*, between light and heavy vehicles).

The main goal is to estimate the traffic conditions *all along the road* (*i.e.*, between sensors) at *current* and *future* time – in terms of macroscopic quantities like flux f, density ρ, and velocity v – by means of a first-order multi-class LWR-like macroscopic differential model which describes the joint dynamics of light and heavy traffic, already introduced in [163]. More in detail, we consider 2 steps:

Nowcast This is the traffic estimation at current time t_0 (now), at every point of the road. To do that, we split the road into several consecutive segments, each of which starts and ends with a sensor. Then, we run the model setting the initial time $t = t_0 - \Delta t_{\text{PAST}}$ and the final time $t = t_0$, where Δt_{PAST} is a parameter. The model runs in each segment independently, providing the evolution of the density. At time $t_0 - \Delta t_{\text{PAST}}$ we assume that the road is empty. Then, the road starts filling thanks to the sensor data which act as inflow and outflow boundary conditions. If Δt_{PAST} is sufficiently large, the road fills completely and a reliable density estimate is computed along each segment.
The problem that arises is how to employ sensor data to enforce boundary conditions: mathematical models typically require *density data* as Dirichlet boundary conditions, but in our case sensors do not provide this information. Alternatively, one can inject the *flux data* directly into the numerical scheme chosen for the discretisation of the modelling equations. Unfortunately, this solution is not always feasible because sensor data are not guaranteed to be compatible with the solution of the numerical model; moreover, the solution is not always the "correct" one, especially in the case of congestion events appearing between sensors, see Section 5.6 for details. This is the reason why we explore a third approach: we train an ANN based on an LSTM to detect congestion formation at sensors in real time. This is an interesting and complex problem *per se*, which gives, as a by-product, a tool for *inverting the concave fundamental diagram without ambiguity*, being able to distinguish the free phase from the congested phase, see Figure 5.2**(a)**. The tool is then used to transform

the flux datum into a density value and to be injected into the model, thus solving the incompatibility issues mentioned above. Do note that this solution implicitly adds new physical information to the model, leading, in most cases, to a more accurate solution.

Forecast This is the traffic estimation at any future time $t_0 < t < t_0 + \Delta t_{\text{FUT}}$, where Δt_{FUT} is the duration of the simulation (30 min, in our case). In this case, sensors data are not yet available, hence we forget the sensors and we consider the whole highway as a unique long segment. We employ the same mathematical model considered before, using the nowcast traffic estimation as initial conditions for the density.

Since the model needs the boundary conditions for any time $t_0 < t < t_0 + \Delta t_{\text{FUT}}$, *i.e.* it needs to have an estimation of the number of vehicles which will enter and leave the road until time $t_0 + \Delta t_{\text{FUT}}$, the problem arises how to enforce these boundary conditions, which sensors clearly cannot provide at time t. To do that, we set up a different LSTM-based ANN to predict sensor data in the time interval $[t_0, t_0 + \Delta t_{\text{FUT}}]$. More precisely, the output of the ANN will not be the minute-by-minute flux data, since they are too fluctuating to guarantee a reliable prediction; instead, we opt to predict the *total* number of vehicles in the time interval $[t_0, t_0 + \Delta t_{\text{FUT}}]$: a simpler yet useful piece of information since it can be interpreted as a constant boundary condition that, in most of the cases, offer a good accuracy regarding the total mass found along the whole road at $t = t_0 + \Delta t_{\text{FUT}}$.

5.1.3 Chapter organisation

The rest of the chapter is organised as follows. Section 5.2 introduces the traffic flow data we work on and discusses the benchmark dataset provided by Autovie Venete S.p.A; in particular, we highlight the criticalities related to these kinds of data that further motivate this work. Section 5.3 introduces in general terms the structure of the ANN which will enrich the dataset. Section 5.4 discusses the creation of the dataset (Section 5.4.1) and the (hyper)training procedures (Section 5.4.2) of the networks which provide real-time and short-term information about congestion events; a few examples are discussed both for real-time detection (Section 5.4.3) and short-term prediction (Section 5.4.4). Section 5.5 discusses the creation of the dataset (Section 5.5.1), the training procedure (Section 5.5.2), and the performances (Section 5.5.3) of the network which provides information about a mid-term (30 minutes) forecast of the expected traffic volume at sensors. Section 5.6 introduces the mathematical model used for traffic estimation and discusses the link between it and the enriched dataset obtained by using the ANNs previously described. Results related to the inversion of the fundamental diagram are described in Section 5.6.1, where a simplified model is also used to clarify advantages and shortcomings; estimation of the upcoming traffic volume is described in Section 5.6.2 instead. Finally, Section 5.7 concludes the work with some final remarks.

5.2 Discussing the data

Our benchmark dataset is provided by the Italian motorway company Autovie Venete S.p.A. and it was collected between September 2020 and March 2022. It contains traffic data from fixed sensors located along three highways in the North-East of Italy, namely A23, A28, and (part of the) A4. All of the highways have two lanes per direction except for the A4 which has three lanes in some parts. As usual, vehicles have different speed limits on the basis of their length and weight, moreover, heavy vehicles cannot use the fastest lane.

Data are collected by 45 groups of fixed sensors which provide, in total 301,200 records per day on average. Each group is characterised by a position x along the road and a direction of travel and consists of *1 sensor per lane* (therefore we have 2 or 3 sensors per group). Each sensor

- counts passing vehicles;
- classifies them according to the German TLS[1] 5+1 class standard;
- measures the speed of vehicles.

The claimed relative error on counting and velocity is ±3%.

In the following, we aggregate TLS classes 1 and 2 as *light vehicles* (motorcycles, cars, vans, and car trailers) and classes 3, 4, and 5 as *heavy vehicles* (lorries, lorry trailers, tractor vehicles and buses), creating two macro-classes. In Sections 5.4 and 5.5 we will further aggregate data by macro-class or by group.

The spatial granularity is highly variable since the distance between sensors ranges from 1 to 20 km. The temporal granularity is instead more regular since data are transmitted by each sensor every 1 minute, as *aggregate measurements*: this means that the database stores the total number of vehicles passed in that time interval and the average velocity per each class, see Figure 5.3.

The measurements are kept for 2 hours for real-time analysis then they are moved to a separate database. Consequently, historicised data are not available at real time.

Some remarks are in order:

1. Although flux data show a regularity on a daily basis, they are very fluctuating from minute to minute, see Figure 5.3.
2. Sensors provide flux and velocity data only. Unfortunately, the density of vehicles cannot be accurately recovered from the available data simply inverting the basic relation
$$f = \rho v. \tag{5.1}$$
This limitation is due to the fact that, in the real world, flux f, velocity v, and density ρ can only be measured in finite intervals of time and space, *i.e.* they cannot be calculated exactly at the same point x and time t. Relation (5.1) fails in particular whenever large density values come into play since large densities

[1] Technische Lieferbedingungen für Streckenstationen (TLS) is a standard specification for the collection of traffic data that, amongst others, provides the guidelines to classify vehicles in either 2, 5+1, or 8+1 classes, see [164].

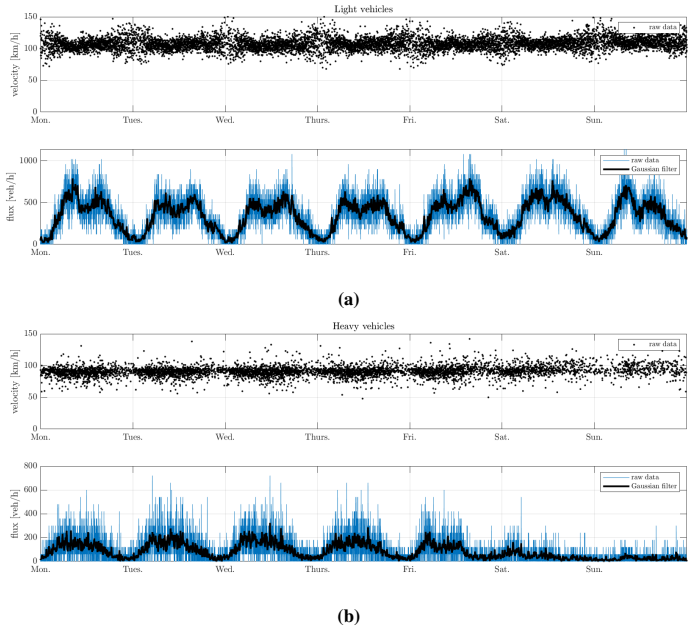


Fig. 5.3: Velocity and flux for **(a)** light and **(b)** heavy vehicles from a single representative sensor; entire week from Monday to Sunday. We observe that the raw flux data are very fluctuating from minute to minute, but, applying a Gaussian filter (black line), one can recognise a certain pattern repeated on a daily basis. At night, the flux data of all vehicles drop, while the velocity data of light vehicles become more scattered. As expected, during the weekend, the flux of heavy vehicles is quite low.

are usually associated with low fluxes and low velocities, which require long time intervals to be detected, cf. [180].

3. Most importantly, our data cannot distinguish between an empty road and a fully congested road. *In both cases, the measured flux is 0 and the velocity is undefined.*

These remarks are important to understand how it can be difficult to detect the formation of a congestion event in real time, which is, in turn, essential for a good estimation and forecast of the traffic flow, even far from the sensor which first detects the congestion. In order to better understand this point, we show in Figure 5.4 four congestion events which develop with different characteristics. In Figure 5.5, instead, we show two very similar traffic conditions characterised by a flux drop, which evolve in a totally different manner: one into the free regime and the other into the congested regime. This makes it clear that, despite being relatively easy to detect congestion events (distinguishing them from empty road conditions) by observing data *a posteriori* (*e.g.*, a whole day), it is very difficult to do the same *at the moment* of the congestion formation.

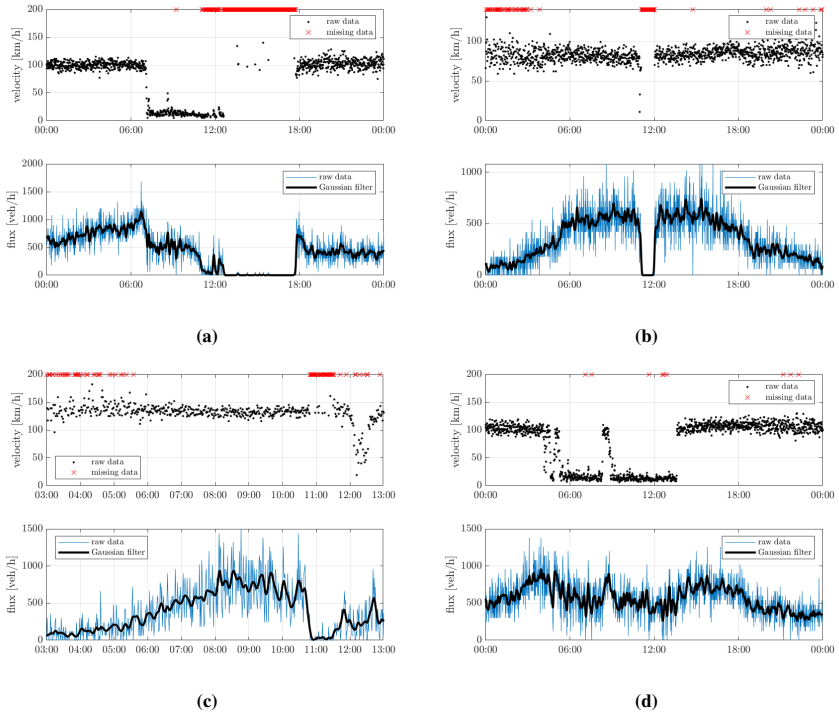


Fig. 5.4: Four congestion events detected by different sensors. Each event has peculiar features: **(a)** we observe a rapid velocity drop and flux drop, then flux vanishes while velocity is undefined (with some exceptions for some fast vehicles still passing); **(b)** flux and velocity drop abruptly; **(c)** flux drops first, then traffic restarts at low velocity; **(d)** velocity drops while flux is only partially lowered.

In the rest of the chapter, we will employ the dataset detailed above for describing our data-/model-driven methods. Although the numerical and computational procedures are tuned for this specific dataset, we think that the adopted methods are valid and can be used in more general situations, and that they can be especially relevant for highways with a small number of lanes and an important presence of heavy vehicles.

5.3 Supervised machine learning approach for the dataset

In this section, we set up a unique ML-based approach to solve both the two problems introduced in Section 5.1, namely the real-time detection of congestion events and the forecasting of the expected average flux at sensors. As we have already recalled, among the supervised learning techniques, RNN in general, and LSTM in particular,

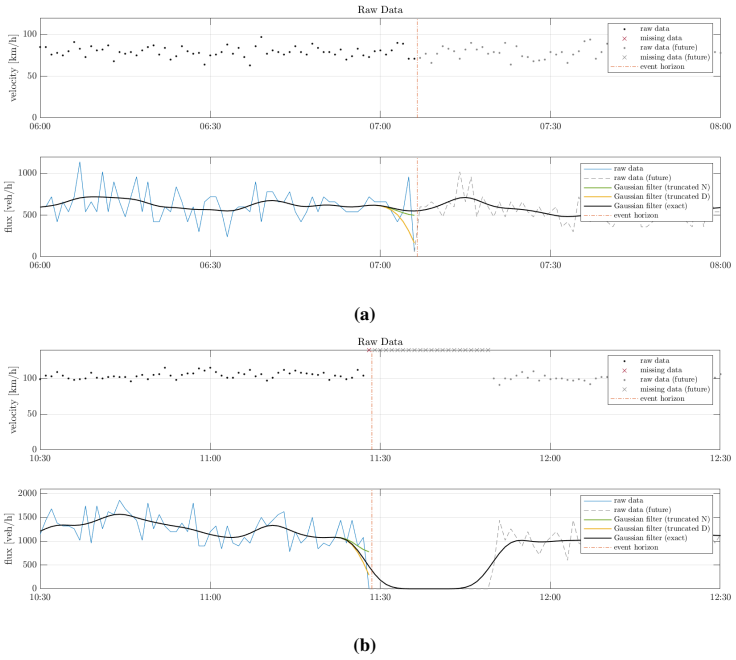


Fig. 5.5: Two similar traffic conditions that evolve differently. **(a)** Normal traffic conditions characterised by the usual high fluctuation of the flux. At 7:05 the flux drops abruptly then the traffic restarts normally; **(b)** At 11:29 a very similar situation appears but, this time, it evolves into a queue. Besides the Gaussian filter already shown in Figure 5.3, here we also show two other Gaussian filters obtained without using future data (beyond the event horizon). Truncation is obtained assuming either Dirichlet-like boundary conditions or Neumann-like boundary conditions. We see that neither Gaussian filters nor raw data are enough to distinguish between the two scenarios at the event horizon.

proved to be very effective in time series analysis. Hence, in the following, we define temporal sequences as vectors $\mathbf{X}$ of feature vectors $\mathbf{x}$, *i.e.* $\mathbf{X} = (\mathbf{x}_t)_{t=1,2,\dots}$, where the number of features N_{IN} is fixed (namely $|\mathbf{x}_t| = N_{\text{IN}}$). In our case, the features which can be extrapolated from the dataset are flux f and velocity v organised by class of vehicles and/or by lane.

We recall from Section 2.3.3 that LSTM output a N_{HID}-length vector $\mathbf{h}$ per each time step of the sequence, and that the internal memory of the network shares the same dimension with $\mathbf{h}$. Hence, in order not to bind the size N_{HID} of the internal memory to the size of the required output, we naturally need an intermediate component to manipulate $\mathbf{h}$ into the actual prediction. Our tool of choice is a vanilla SLP Feed-Forward Network that condenses the N_{HID} features in

Regression task a N_{PRED}-length output $\mathbf{o}$, where each entry represents an individual prediction.

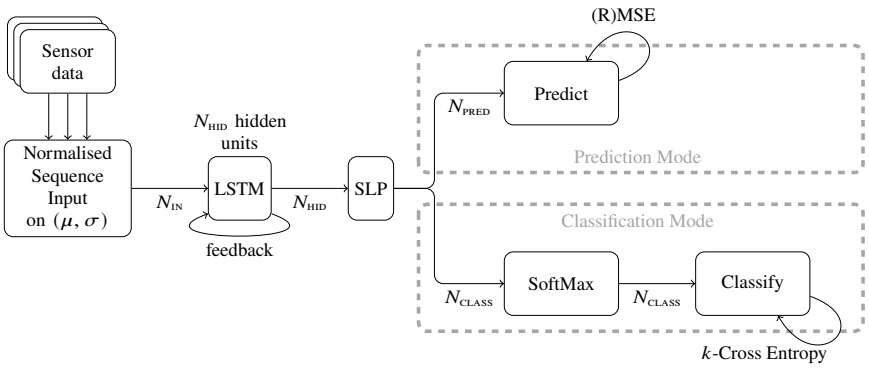


Fig. 5.6: Pipeline of the data enriching tool. The time series obtained from sensors are globally feature-wise normalised before feeding it in the LSTM, one-time step at a time. The output of the LSTM is then processed by an SLP layer. Finally, a SoftMax layer is eventually applied (if task is classification), and the result is provided.

Classification task a N_{CLASS}-length features-vector $\mathbf{z}$, where each entry represents a class of the problem, then fed into a softmax layer (see (2.6)) to transform them into a probability vector $\hat{\mathbf{o}}$ (of being of a specific class). The final output can be either the probability vector $\hat{\mathbf{o}}$ or the index of the highest-probability class o, *i.e.* $o = \text{argmax}(\hat{\mathbf{o}})$. Do note that $\hat{\mathbf{o}}$ can be used as a *confidence* indicator of the prediction as the closer is $\max(\hat{\mathbf{o}})$ to 1, the more certain is the prediction according to the ANN (see later *e.g.* Figure 5.9).

To complete the pipeline of our methodology (a summary of which, can be found in Figure 5.6), we further need to specify how to evaluate the system performance against our ground-truth data, *i.e.* we need to choose an error function to evaluate the distance of our guesses from ground truth (a loss function, in the jargon).

Regression task Being the prediction output a real number, since we are not interested in weighting differently over- and under-errors, we went for the vanilla (Root) Mean Squared Error (see Equation 2.8).

Classification task Since we always classify data between a positive and a negative class (boolean classification), we went for k-cross entropy (with $k = 2$, cf. Equations (2.10), (2.9)). In particular, since our dataset shows a great disequilibrium between positive and negative samples, we decide (instead of enriching it with synthetic data that are often complex to generate) to weight the loss entropy on the positive ratio $0 < p_r < 1$. We recall that the softmax applied to a binary classification task returns a 2-value vector $\hat{\mathbf{o}}_t = (1 - \hat{o}_t, \hat{o}_t)$ where $\hat{o}_t$ represents the probability that the positive class is chosen. Hence, if y_t is the ground-truth bit corresponding to input $\mathbf{x}_t$ – meaning that our target $\mathbf{y}_t$ is either $(1, 0)$ or $(0, 1)$ – then the weighted binary-cross entropy is defined as

$$-\frac{1}{|\mathbf{X}|}\sum_{t=1}^{|\mathbf{X}|}\Big(\big((1-p_r)\cdot y_t\cdot\ln(\hat{o}_t)\big)+\big(p_r\cdot(1-y_t)\cdot\ln(1-\hat{o}_t)\big)\Big)\,. \tag{5.2}$$

5.4 Detection and short-term forecast of congestion

In this Section, we build a labelled dataset of congestion events and we use the previously described ANN-based methodology to build two classifiers: a first classifier $\mathfrak{F}_c$ for performing real-time detection of congestion events, and a second classifier $\mathfrak{F}_p$ to perform short-term forecasting of the same congestion events. The two classifiers act as a sort of "congestion alarm" and "congestion pre-alarm" launchers, respectively.

For our purposes, it is crucial to note that 17 groups of sensors out of the 45 deployed are also equipped with advanced technologies[2] that, combining Doppler radar, ultrasound emitters, and passive infrared radiation detectors, are able to determine also stopped vehicles and intense congestion conditions, reporting them as boolean flags in the corresponding minute. In the following, we will refer to these sensors as $3T$ sensors and we will use them as one of the main sources of data for supervised learning purposes.

5.4.1 Building the dataset

The dataset is created using data gathered by the $3T$ sensors. Since the congestion event is detected by each sensor (*i.e.*, in each lane), we decided to aggregate the data of all classes of vehicles and working lane by lane. Therefore, the features are the total flux f and the average velocity v of all vehicles (all classes). Actually, do notice that recovering *a posteriori* a piece of information per class is often quite easy. In fact, if a congestion event is detected in the slow lanes and not in the fast lane, it is highly probable that the congestion is for heavy vehicles only, since they cannot use the fast lane, see Section 5.2.

We split the data per day, so to perform an incremental training phase using only randomised batches of days of data at a given time. Therefore, we get dataset samples constituted by a two-feature 1440-long (24 h × 60 min) sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{1440})$, with $\mathbf{x}_t = (f_t, v_t)$, $t = 1, \ldots, 1440$. The dataset should be labelled with a suitable 1440-long boolean array $\mathbf{y} = (y_1, \ldots, y_{1440})$ reporting whether the corresponding minute t is labelled as a congestion event or not. The array $\mathbf{y}$ is computed by means of a logical combination of three different computational procedures, each of which leads to a truth value in $\mathbb{F}_2$. More precisely, we have

$$\mathbf{y} = \mathbf{b}^{3T} + \mathbf{b}^{f} + \mathbf{b}^{v}, \tag{5.3}$$

[2] Provided by Asim Technologies Ltd, series TT295.

where we recall that + in $\mathbb{F}_2$ corresponds to the logical *or*, and the definition of the procedures follows.

$3T$ **data** $\mathbf{b}^{3T}$ is the flag for the congestion event transmitted directly by the $3T$ sensors. On average, the amount of daily congestion events per sensor is pretty low ($< 0.1\%$), hence, in order to make the dataset more balanced, we selected the samples reporting at least $\sim 1\%$ of positive labelling (*i.e.* $\sum_{t=1}^{1440}(b_t^{3T} \geq 15)$). We noticed that the $3T$ detection procedure is quite "conservative", meaning that it tends to report a congestion event only if there is a stable queue under the sensor. Moreover, the exact definition of congestion event, as well as the physical and computational procedure used to detect it, are not publicly available and they are not known by the authors. For these reasons, we decided to enrich the true-samples with the two following heuristics.

Flux heuristic $\mathbf{b}^f$ considers the flux array $\mathbf{f} := (f_1, \dots, f_{1440})$ only, along with a 10-fold Gaussian regularisation $\mathbf{f}^*$ obtained via convolution with a triangular kernel, cf. Figure 5.3; a congestion event is reported at time t if the sensor detects the following condition during the daytime (from 5 AM to 8 PM)

$$\mathbf{b}^f = 1 \qquad \Longleftrightarrow \qquad \begin{cases} f_{t-1} < 2, \\ f_t < 2, \\ f_t^* < 2, \\ \left(\frac{1}{60}\sum_{s=t-60}^{t-1} f_s\right) - f_t^* > 2 \end{cases} . \tag{5.4}$$

The idea behind this heuristic is that a sufficient condition for a congestion event is a low flux at the current time combined with a sudden drop of the flux (*i.e.*, a high average flux in the previous time period).

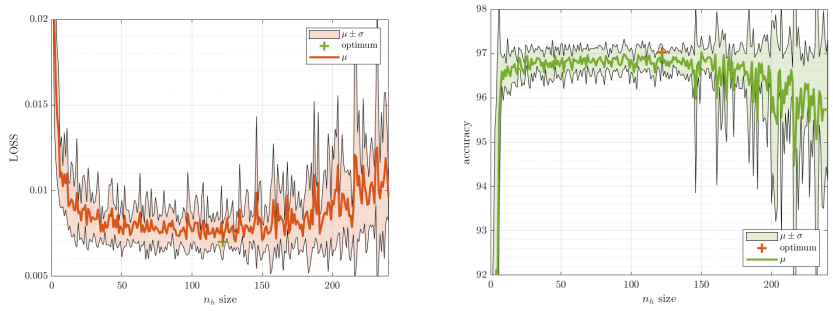
Velocity heuristics $\mathbf{b}^v$ considers instead the velocity array $\mathbf{v} := (v_1, \dots, v_{1440})$ only, along with a regularisation $\mathbf{v^*}$ (obtained as before). A congestion event is reported at time t if the sensor detects the following conditions during the daytime (from 5 AM to 8 PM):

$$\mathbf{b}^f = 1 \qquad \Longleftrightarrow \qquad \begin{cases} v_t < v_{t-1} \\ 0 < v_t < 65 \\ \max\{v_s^* - v_t \mid s = t-1, \dots, t-15\} \geq 40 \end{cases} . \tag{5.5}$$

The idea behind this heuristic is that a sufficient condition for a congestion event is a low velocity at the current time combined with a sudden drop of the velocity (*i.e.*, a high velocity in the previous time period).

Observation The two heuristics were developed during a collaboration with the data owner and have been human-validated through over a month of direct observations.

Observation We point out that the two heuristics both use data that are not available yet at time t (because of the regularisations), making them useless to perform real-time applications.



(a) **(b)**

Fig. 5.7: (a) k-Cross Entropy (loss) and **(b)** accuracy achieved by the training sessions for $\mathfrak{F}_c$ at the varying of N_{HID} over the test set. As it can be seen, the better parameter is $N_{\text{HID}} = 122$ (*i.e.* corresponding to 2 hours and two minutes). Since N_{HID} is comparable *w.r.t.* the length of the time series under analysis (two hours, cf. Section 5.2), this suggests that having longer time series might refine the results even more.

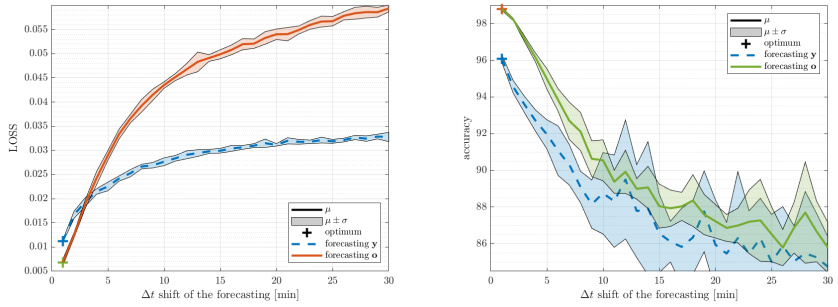
Analysing the average and the standard deviation of the flux data we found that most of the sensors dispatched on A4 behave very differently from the others ($\mu \sim 1 \times 10^{+4}$ vs. $2 \times 10^{+3}$ vehicles/day and $\sigma \sim 1 \times 10^{+4}$ vs. $5 \times 10^{+3}$). This is not surprising since A4 connects more populated areas compared to A23 and A28. For this reason, we split the sensors into two disjoint sets, namely *high flux* (HF) and *low flux* (LF), being LF all the sensors in A23, A28, and in the fastest lane of A4 (where it has 3 lanes). In the rest of this and the following section we consider HF sensors only, the procedure and analysis for LF sensors being similar.

We split the HF 119-day dataset $\{(\mathbf{X}^{(d)}, \mathbf{y}^{(d)})\}$, $d = 1, \ldots, 119$ (171,360 training minutes) in 96 training days and 23 validation days. The average sample positive rate after the enrichment is $p_r = 4.2\%$, hence we weighted the k-cross entropy function opportunely.

5.4.2 Training the model

We started by training the HF classifier $\mathfrak{F}_c$ first: we performed multiple training sessions to estimate the suitable size N_{HID} of the memory of the LSTM. Each training session was carried out by using the ADAM optimiser working on randomly shuffled batches of size 24 samples over 100 epochs arranged with 9 progressive refinements of the learning rate (10 epochs per refinement). Each training procedure requires 30 to 300 seconds while trained networks do process data in a few milliseconds, making them suitable for real-time applications.

Figure 5.7 reports the results in terms of loss and accuracy for the various training sessions, where we vary $N_{\text{HID}} \in [1, 240]$ *i.e.* at most double the length of the available time series. The choice of limiting the LSTM parameter N_{HID} to 240 hidden units



(a) **(b)**

Fig. 5.8: (a) k-Cross Entropy (loss) and **(b)** accuracy achieved by the training sessions for $\mathfrak{F}_p$ at the varying of the forecasting time window Δt (x-axis), both for predicting $\mathbf{y}^{(d)}$ (in dashed blue) and $\mathbf{o}^{(d)}$ (in orange/green). As expected, the smaller Δt, the better the prediction.

during the parameter tuning phase is further motivated by the fact that both loss and accuracy deteriorate around 150 hidden units and beyond.

We found $N_{\text{HID}} = 122$ being the most suitable parameters for the model (with $N_{\text{IN}} = 2, N_{\text{CLASS}} = 2$), achieving a mean accuracy of $\mu = 97.03$ ($\sigma = 0.17$) and a mean loss of $\mu = 7.01 \times 10^{-3}$ ($\sigma = 0.11 \times 10^{-3}$). We decided to further explore the values of N_{HID} around 122, performing multiple training sessions with different rate drops and $N_{\text{HID}} \in [110, 130]$. We obtain the best model for $N_{\text{HID}} = 120$, yielding an accuracy of 97.70% (99.75% if weighted *w.r.t.* p_r) and a corresponding loss value of 5.2×10^{-3} over the test set. 10-fold cross-validation achieved similar performances in both terms of loss ($\mu = 8.97 \times 10^{-3}, \sigma = 2.68 \times 10^{-3}$) and accuracy ($\mu = 96.27, \sigma = 0.56$).

Secondly, we tackled the problem of training the classifier $\mathfrak{F}_p$. We fixed the dimensions of the model (namely $N_{\text{IN}} = 2$, $N_{\text{HID}} = 120$, and $N_{\text{CLASS}} = 2$) and we focused on how many minutes we could anticipate the classification of $\mathfrak{F}_c$ without losing too much in accuracy. The trivial way would be producing a labelled dataset for the forecasting task by applying a Δt minutes shift to the vectors $\mathbf{y}^{(d)}$ in order to "bring backwards" the congestion events. However, we prefer forecasting the output $\mathbf{o}^{(d)}$ of $\mathfrak{F}_c$ itself (*i.e.* the argmax of the probability vector) rather than the target $\mathbf{y}^{(d)}$ of $\mathfrak{F}_c$. This change in objective is in fact convenient for two main reasons: (i) the output $\mathbf{o}^{(d)}$ is in someway a "regularised" version of the real target $\mathbf{y}^{(d)}$, hence it should be easier to forecast; (ii) philosophically speaking, we are expecting that a $\mathfrak{F}_p$ alarm to be followed by a $\mathfrak{F}_c$ one. In order to further correct the dataset, however, we also filtered the novel target vector from all the isolated congestion events, actually filtering the ones that probably were false-positives of $\mathfrak{F}_c$.

The results of the training with the two proposed datasets are reported in Figure 5.8, where loss and accuracy are compared at the varying of the Δt time shift. Analysing the accuracy, it can be seen that trying to forecast $\mathbf{o}^{(d)}$ is in general simpler; the loss, on the contrary, grows slower when the target is set to $\mathbf{y}^{(d)}$ actually prompting that training on $\mathbf{o}^{(d)}$ is less conservative in terms of false-positives. We

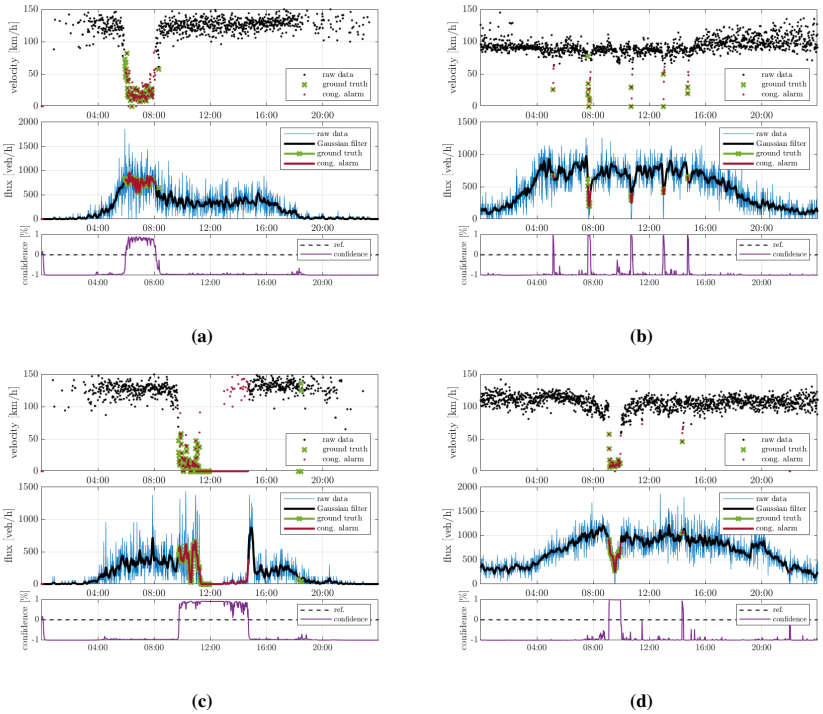


Fig. 5.9: Four examples of day-length classification of congestion events. Each plot reports the velocity (top), the flux (middle) and the confidence of the model for the classification (bottom). **(a)** and **(c)** are from LF sensors while **(b)** and **(d)** are from HF sensors.

found $\Delta t = 4$ min being the most suitable parameter (in particular with the second dataset, still having comparable loss) yielding interesting predictions while keeping the average accuracy above 95% (with the best peak of 96.50%, or 98.82% if weighted) with a corresponding average loss value of 2.4×10^{-2}, hence comparable to results obtained by $\mathfrak{F}_c$. Training times for the networks are consistent with those from $\mathfrak{F}_c$ and 10-fold cross-validation results in similar outcomes, in both terms of loss ($\mu = 2.95 \times 10^{-2}, \sigma = 3.53 \times 10^{-3}$) and accuracy ($\mu = 93.81, \sigma = 1.22$).

5.4.3 Results of the real-time classifier $\mathfrak{F}_c$

Figure 5.9 shows a few examples of day-length classification of congestion events for both LF and HF sensors. We report the velocity v, flux f, and the confidence of the classification (given as $\hat{o}$ rescaled in the interval $[-1, +1]$, *i.e.* $\hat{o} \cdot 2 - 1$). From Figure 5.9**(a)**, we can notice that when the drop is significant only in v but not in f, the model shows a little uncertainty to classify the event. However, as can be seen

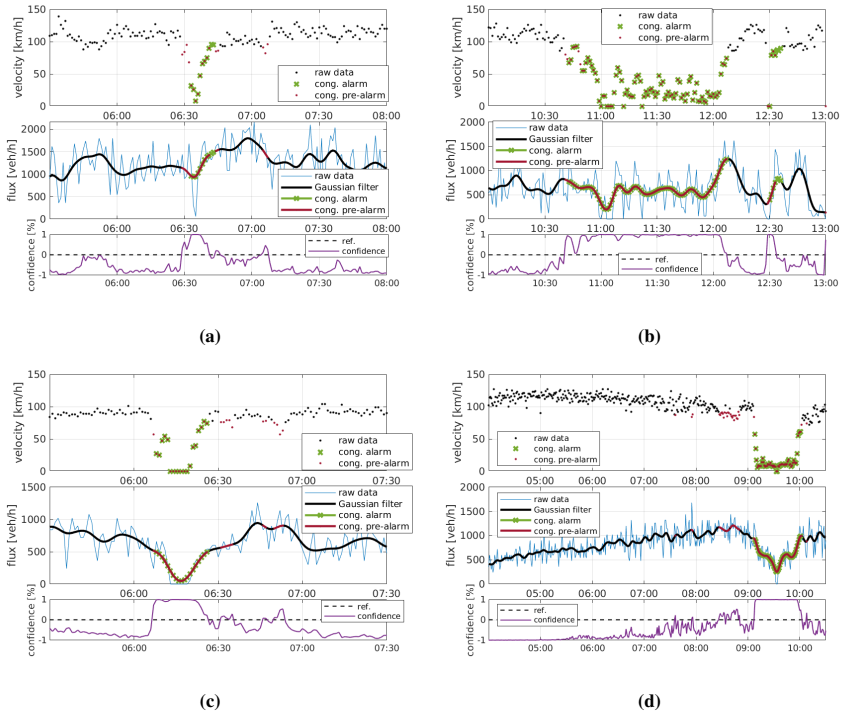


Fig. 5.10: Four examples of prediction of congestion events. **(a)** at 6:30 a congestion event is correctly pre-alarmed four minutes in advance. **(b)** at 10:45 a congestion event begins intermittently while a solid pre-alarm shows up. **(c)** a difficult-to-predict congestion appears at 6:06. **(d)** at 8:30 an uncertain situation shows up, characterised by below-average velocity and normal flux, then after 9:00 a clear congestion is formed (pre-alarmed 1 minute in advance).

in Figure 5.9**(b)**, if the drops involve also f, then the model is able to classify the congestion even if they are short in time. Figure 5.9**(c)** prompts that, in particular circumstances, the model is able to classify critical events also when the ground truth is uncertain, like when the flux is close to zero and velocity is high; do note that this is a typical situation at night, with no congestion. Finally, Figure 5.9**(d)** shows that the model also keeps the congestion alarm on when the critical situation is vanishing: this comes from the fact that the model does not know the future, conversely to the regularisation applied by the heuristics.

5.4.4 Results of the short-term classifier $\mathfrak{F}_p$

In order to give an idea about the actual possibility to predict congestion events, in Figure 5.10 we show four outcomes which are commonly observable. Figure 5.10**(a)**

shows a very favourable case in which we are able to predict the congestion 4 minutes before the actual formation. This is made possible since the velocity starts lowering a bit before dropping down, together with the flux. In Figure 5.10**(b)**, a long-standing congestion is preceded by a confusing scenario in which the congestion alarm is on and off. In this case, the congestion pre-alarm is constantly on. Figure 5.10**(c)** highlights a case where prediction is quite hard, even for a trained human: congestion begins abruptly with a velocity drop, and the ANN is able to predict it only 1 minute in advance. Finally, Figure 5.10**(d)** depicts an uncertain situation which can likely evolve into a queue (but it does not, at least for some minutes). The pre-alarm is on, and even if this is formally incorrect (the congestion pre-alarm is not followed by a congestion alarm) the uncertainty fully justifies the ANN behaviour.

5.5 Computation of expected traffic volume

In this section, we tackle the problem of performing mid-term prediction (30 min, coherent with the simulation time we adopt in the upcoming Section 5.6) of the expected traffic volume by building an ANN-based predictor $\mathfrak{P}$. Let us recall that, unlike in the previous section, here we aggregate sensors belonging to the same group (following the definition given in Section 5.2), while working on the two distinct, but coupled, classes of vehicles (light and heavy). This is important because the dynamics of the two classes are very different one from the other and, at the same time, they are strongly interconnected.

5.5.1 Building the datasets

Ground-truth data $\mathbf{y}$ can be easily produced for each (group of) sensor, only needing the real flux of light vehicles $\mathbf{f}^{\text{L}}$ and that of heavy vehicles $\mathbf{f}^{\text{H}}$ for the τ upcoming minutes ($\tau = 30$ min in our experiments). These compose the feature vector $\mathbf{x}$ as $(\mathbf{f}^{\text{L}}, \mathbf{f}^{\text{H}})$ and the corresponding ground truth $\mathbf{y}$ as:

$$y_t = \left(\frac{1}{\tau} \sum_{s=t+1}^{t+\tau} f_s^{\text{L}},\ \frac{1}{\tau} \sum_{s=t+1}^{t+\tau} f_s^{\text{H}} \right). \tag{5.6}$$

For what concerns the dataset creation, we selected the data collected during the entire 2021, corresponding to 348 day-wise readings (17 days are missing due to reported reading problems). Due to the high number of available data, we decided to tune a predictor per group of sensors. As for the previous training, we organised the features in day-length sequences, and we built a dataset $D^{(g)} = \{\mathbf{x}^{(d)}; \mathbf{y}^{(d)}\}$, $1 \leq d \leq 348$, for each group g. We extracted roughly one month of samples from the dataset to form the test set while we used the remaining part as the training set.

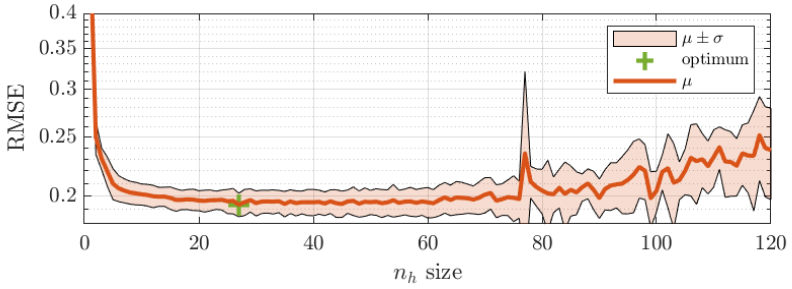


Fig. 5.11: Value of the RMSE achieved by the training session for $\mathfrak{P}$ at the varying of N_{HID}. As it can be seen, the best parameter is $N_{\text{HID}} = 27$.

5.5.2 Training the model

We trained a different model $\mathfrak{P}^{(g)}$ per group of sensors over its corresponding dataset $D^{(g)}$. In particular, we focus on the four groups representing the inflow boundaries of the highway network under consideration: Venice (g_1) and Trieste (g_2) for the A4, Conegliano (g_3) for the A28, and Udine (g_4) for the A23. g_1 is the only group deployed on a three-lane section.

By adopting the same nomenclature as from Figure 5.6, the choice of the input and output features requires $N_{\text{IN}} = 2$ and $N_{\text{PRED}} = 2$, while N_{HID} is a free parameter. Hence, we performed multiple training sessions to estimate the suitable size for the memory N_{HID} of the LSTM. Each training session was carried out with the ADAM optimiser, working with a variable learning rate, piece-wise decreasing over 5 progressive learning eras of 50 epochs each (for a total of 250 epochs). Each training session required 70 to 480 seconds, a larger time compared to $\mathfrak{F}_c$ and $\mathfrak{F}_p$ training due to the bigger datasets employed. However, such training times are feasible with large training campaigns over all the sensors. Single data are processed in a few milliseconds.

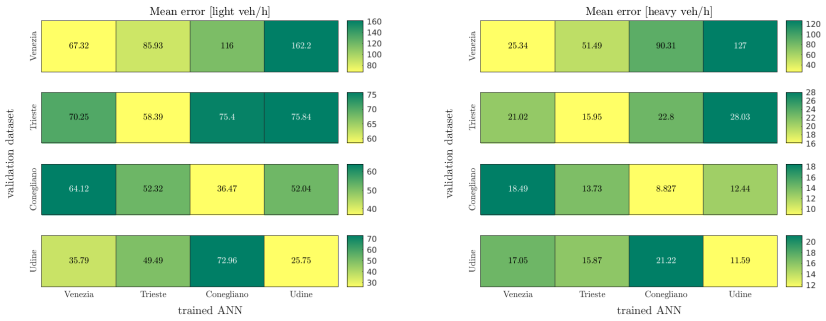
Figure 5.11 shows the value of the RMSE as a function of the free parameter $N_{\text{HID}} \in [1, 120]$, *i.e.* at most 4T. The choice of limiting the model parameter N_{HID} to 120 hidden cells during the parameter tuning phase is further motivated by the fact that RMSE quickly deteriorates around 70 and beyond (conversely to the trend obtained in Figure 5.7). We found $N_{\text{HID}} = 27$ being the best parameter to correctly capture the trend of the average flux, achieving a mean RMSE of $\mu = 0.1936$ with standard deviation $\sigma = 8.7 \times 10^{-3}$. Comparable RMSE performances are achieved with 10-fold cross-validation ($\mu = 0.24391, \sigma = 4.84 \times 10^{-2}$).

Table 5.1 reports the average errors committed by the best model (with $N_{\text{IN}} = 2, N_{\text{HID}} = 27, N_{\text{PRED}} = 2$) obtained per each group of sensors. Note that the RMSE reported in Figure 5.11 is evaluated on the normalised dataset, hence it is not comparable with results from Table 5.1.

The question arises if it is really needed to train a different LSTM per group of sensors or if one ANN can serve all. To address this question we report in Figure 5.12

	Light vehicles		Heavy vehicles	
	2-lane	3-lane	2-lane	3-lane
Mean average-error	42.74	76.60	11.74	25.06
Max average-error	70.80	95.06	19.48	33.44
Mean standard-deviation	41.72	62.96	12.04	23.94
Mean max-error	422.64	744.40	164.46	251.24
Mean average-flux	387.74	768.16	96.52	281.42
Mean max-flux	5320.64	5864.48	1825.30	2047.10

Table 5.1: Error statistics in [vehicles/h] achieved by multiple trainings of $\mathfrak{P}$ (with $N_{\text{HID}} = 27$) over the four datasets $D^{(g_i)}$ (average is performed on all minutes available in the dataset). Values are sorted by vehicle class and number of lanes. Average and maximum flux are also reported as benchmark values for relative error statistics.



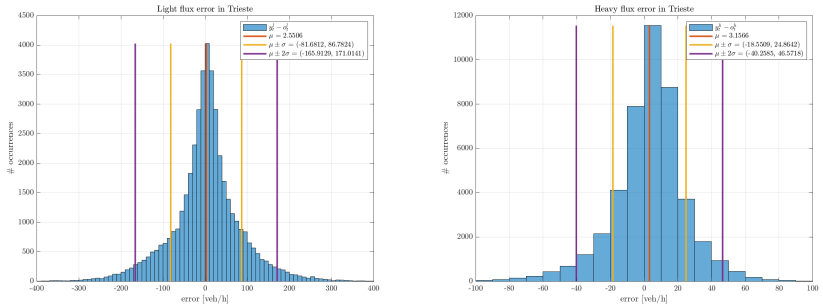
(a) **(b)**

Fig. 5.12: Mean error ($|y_t - o_t|$) in [vehicles/h] for **(a)** light vehicles and **(b)** heavy vehicles obtained by applying the network $\mathfrak{P}^{(g_i)}$, $i \in \{1, 2, 3, 4\}$ (x-axis) on the mutual test-sets $D^{(g_j)}$, $j \in \{1, 2, 3, 4\}$ (y-axis). As it can be seen, maximum-by-row is achieved on the main diagonal, meaning that applying a specific ANN on a different dataset is not convenient. It is worth noticing that maximum-by-row does not necessarily correspond to the maximum-by-column (see, *e.g.*, $i = 1$, $j = 4$); this phenomenon is due to the high regularity of some datasets.

the error made by $\mathfrak{P}^{(g_i)}$ tested against datasets $D^{(g_j)}$, with $i, j = 1, \ldots, 4$. We see that it is highly suggested to train as many ANN as groups of sensors are considered.

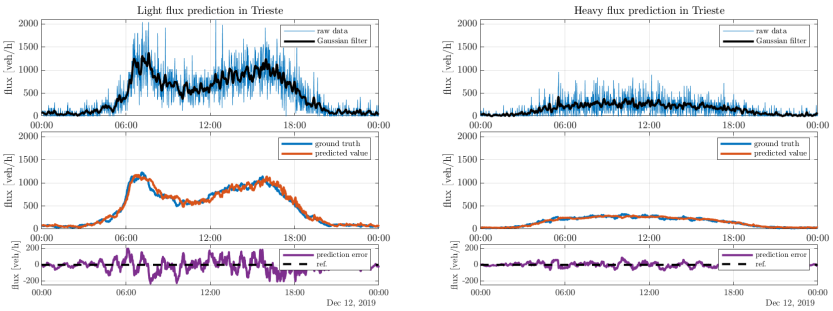
5.5.3 Performance evaluation

We consider the group of sensors g_2 as an example for the performance evaluation. Figure 5.13 shows the histograms of the errors (in [vehicles/h]) made running the prediction every minute available in the test set. Figure 5.14 shows instead the result of $\mathfrak{P}^{g_2}$ on an entire day. Every minute, the ground truth (*i.e.* the total number of vehicles passed in the next 30 minutes, in [vehicle/h]) is compared with its ANN-predicted value, and the offset is evaluated. Real flux $\mathbf{f}$ is reported along with its *a posteriori* regularisation for reference. It can be seen again that the error in prediction



(a) **(b)**

Fig. 5.13: Error histogram of the predictions of $\mathfrak{P}^{g_2}$ over its test set for **(a)** light and **(b)** heavy vehicles. Do note that error is almost symmetrical *w.r.t.* 0.



(a) **(b)**

Fig. 5.14: $\mathfrak{P}^{g_2}$-predicted vs. actually measured average flux with $\tau = 30$min. Averages are performed minute by minute. Results are reported in terms of both **(a)** light and **(b)** heavy vehicles.

is almost symmetrical and it usually corresponds to less than 200 vehicles/h for light vehicles and much less for heavy vehicles.

Finally, Figure 5.15 shows an example of a single-minute prediction where a pictorial representation of the ground truth is highlighted as the (discrete) integral of the upcoming flux.

5.6 Feeding traffic models with ML-enriched data

The tools introduced in the previous sections are useful *per se*, but they can also be used in combination with other, more classical, tools for traffic estimation and forecast. In this section, we try to get advantage of the pieces of information extrapolated by data in order to improve the accuracy of the macroscopic differential models.

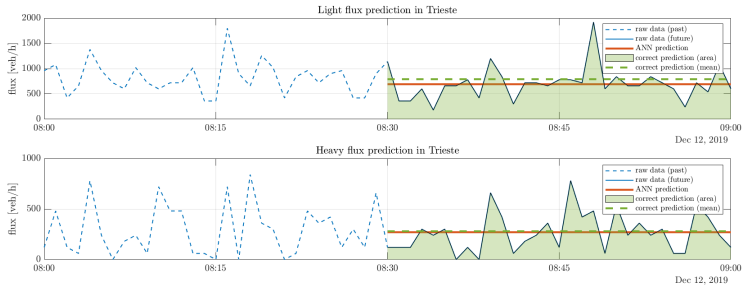


Fig. 5.15: Zoom of a single-minute prediction of the data presented in Figure 5.14. Prediction is performed at time t=8:30, where dashed data are available. The predicted value is given by the area beneath the orange line while the target of the prediction is represented by the value of the green shaded area (represented also as the green dashed line for a better comparison).

5.6.1 Nowcast

In this section, we explore the possible advantages of inverting the fundamental diagram when estimating the current status of traffic density by means of a macroscopic differential model. As mentioned in Section 5.1, we split the whole road $[x_{\text{MIN}}, x_{\text{MAX}}]$ into consecutive segments delimited by fixed sensors. We aim at estimating the traffic density at the current time t_0, therefore we start the simulation at a previous time $t_0 - \Delta t_{\text{PAST}}$, with $\Delta t_{\text{PAST}} > 0$, assuming an empty road at that time. Let us denote by N_S the number of road segments (sensors are $N_S + 1$). Each road segment $S^k := (s^k, s^{k+1})$, $k = 1, \dots, N_S$, is delimited by two fixed sensors, located at $x = s^k$ and $x = s^{k+1}$, respectively.

We begin with a simplified setting, then we move to the real one.

Simplified setting

We adopt the classical single-lane single-class LWR model on each road segment S_k

$$\begin{cases} \partial_t \rho^k(x,t) + \partial_x f(\rho^k(x,t)) = 0, & x \in S^k, \quad t \in (t_0 - \Delta t_{\text{PAST}}, t_0) \\ \rho^k(x, t_0 - \Delta t_{\text{PAST}}) = 0, & x \in S^k \\ \rho^k(s_k, t) = \rho^k_{\text{IN}}(t), & t \in [t_0 - \Delta t_{\text{PAST}}, t_0] \\ \rho^k(s_{k+1}, t) = \rho^k_{\text{OUT}}(t), & t \in [t_0 - \Delta t_{\text{PAST}}, t_0] \end{cases} \tag{5.7}$$

where $\rho^k \in [0, \rho_{\text{MAX}}]$ is the vehicle density for some maximal density $\rho^k_{\text{MAX}} > 0$, and $\rho \mapsto f(\rho)$ is the fundamental diagram. Let us assume, as usual, that $\rho \mapsto f(\rho)$ is concave and denote by σ the argmax of f, *i.e.* $f(\sigma) = \max_\rho f(\rho)$ (see Figure 5.2). Equation (5.7) is defined independently on each segment. Let us also recall that the relation (5.1) holds true.

Equation (5.7) is usually solved by numerical approximation. Let us introduce a grid in the domain $S^k \times [t_0 - \Delta t_{\text{PAST}}, t_0]$, with space step Δx and time step Δt. The time interval is divided into N_t intervals, while each segment S^k is divided into N_x^k cells of length Δx; we denote with $\rho_j^{k,n}$ the corresponding approximate average density in cell C_j^k, $j = 1, \dots, N_x^k$ at time n. Any conservative numerical scheme (see [185]) for (5.7) has the form

$$\rho_j^{k,n+1} = \rho_j^{k,n} - \frac{\Delta t}{\Delta x}\Big(F(\rho_j^{k,n}, \rho_{j+1}^{k,n}) - F(\rho_{j-1}^{k,n}, \rho_j^{k,n})\Big), \qquad j = 1, \dots, N_x^k, \quad (5.8)$$

where F is the *numerical flux* (*i.e.* an approximation of the flux f at the interface between two consecutive cells). For example, in the case of the Godunov scheme, we have

$$F(\rho_-, \rho_+) := \begin{cases} \min\{f(\rho_-), f(\rho_+)\} & \text{if } \rho_- \le \rho_+ \\ f(\rho_-) & \text{if } \rho_- > \rho_+ \text{ and } \rho_- < \sigma \\ f(\sigma) & \text{if } \rho_- > \rho_+ \text{ and } \rho_- \ge \sigma \ge \rho_+ \\ f(\rho_+) & \text{if } \rho_- > \rho_+ \text{ and } \rho_+ > \sigma \end{cases} \quad (5.9)$$

Let us consider, *e.g.*, the right boundary condition of a given segment S^k, which corresponds to the left boundary condition of the following segment S^{k+1} (in the following we drop the index k from ρ for readability). Hence, if $j = N_x^k$, one could follow two kinds of approaches

flux-based approach Directly inject in the scheme the flux datum f_{s^k} measured by the sensor across the interface $S^k|S^{k+1}$ in place of the numerical outgoing flux $F(\rho_{N_x}^n, \rho_{\text{OUT}}^n)$, without estimating the density ρ_{OUT};

density-based approach Evaluate the numerical flux $F(\rho_{N_x}^n, \rho_{\text{OUT}}^n)$, by estimating the density ρ_{OUT}.

The two ways are, in principle, both correct and the first one seems to be more practical since sensors provide flux data only (*i.e.* density is not available at all). On the other hand, flux data f_s provided by the sensor are not always compatible with the solution carried by the numerical scheme. In fact, any flux f_s outside the set of admissible values

$$\left\{F(\rho_{N_x}^n, \rho) : \rho \in [0, \rho_{\text{MAX}}]\right\} \quad (5.10)$$

is not compatible with (5.7) and leads to negative densities, since more mass comes out than it is available in the road, see Figure 5.16**(a)**. The question arises how to enforce the compatibility of the sensor data: a natural solution is to project the sensor data into the admissible set of flux data (5.10).

Regarding the density-based approach, the idea is to use the algorithm introduced in Section 5.4 to help invert the concave fundamental diagram, *i.e.* passing from the flux data to the density data by duly distinguishing between free and congested regimes. More precisely, given the sensor flux datum $f_s < f_{\text{MAX}}$, the choice between ρ and ρ', with $\rho' \neq \rho$, $f(\rho) = f(\rho') = f_s$, is taken depending on the presence or not of the congestion, see Figure 5.16**(b)**.

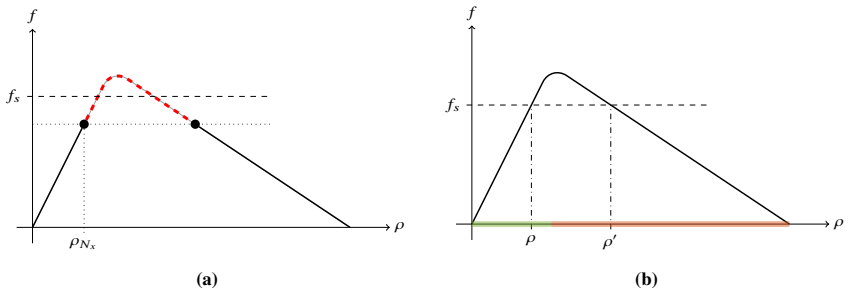


Fig. 5.16: Fundamental diagram $f = f(\rho)$. **(a)** Example of non-admissible sensor flux datum for the numerical flux (5.9): no flux $f_s > f(\rho_{N_x})$ is compatible with any boundary condition ρ_{OUT}. **(b)** Example of flux datum f_s with two possible admissible density ρ and ρ': the correct side (freeway vs. congested) of the fundamental diagram is chosen depending on the presence or not of a congestion.

In order to discuss the difference between the two approaches, we devise a simple numerical test. We consider an infinitely-long single-lane road. As it is usually done in the mathematical literature, we normalise both density and velocity in the interval $[0, 1]$ and we choose the fundamental diagram as $f(\rho) = \rho(1 - \rho)$. In this case, the maximal flux is 0.25 and it is achieved for $\rho = \sigma := 0.5$. Two sensors are located at $x = s^k := 0.45$ and $x = s^{k+1} := 0.8$.

At initial time $t = t_0 - \Delta t_{\text{PAST}}$ the road has constant density $\rho = 0.45$ for $x < 0.52$ while the rest of the road is empty ($\rho = 0$). Immediately after the initial time, an accident occurs at $x = b := 0.6$ and a bottleneck is formed between the two sensors.

Figure 5.17 shows the simulation where the model is aware of the bottleneck. This represents our reference since it evaluates what we assume it is really happening on the road and, hence, it is the result we would ideally like to reproduce with data at our disposal. We can resume the outcome as follows:

1. At the initial time, a rarefaction fan immediately appears at $x = 0.52$ and the right part of the road starts populating
2. When enough vehicles have reached the bottleneck at $x = b$, a queue appears and starts back-propagating
3. The queue reaches the sensor at s^k and continues back-propagating
4. From the bottleneck on, vehicles set off at maximal flux ($\rho = \sigma$) and proceed normally.

It is plain that, if the traffic is observed only at sensors, we cannot be able to perceive the accident and the formation of the bottleneck in its actual position. The effects of the accident will be visible only when the queue reaches the sensor at $x = s^k$.

Figure 5.18**(a)** shows the result obtained by the *density-based* approach. When the queue is perceived at $x = s^k$, the recorded flux $f_{s^k} = 0$ is correctly translated into the maximal density $\rho = 1$ inverting the fundamental diagram. Therefore, the queue continues back-propagating while, for $x > s^k$, the traffic restarts with maximal flux.

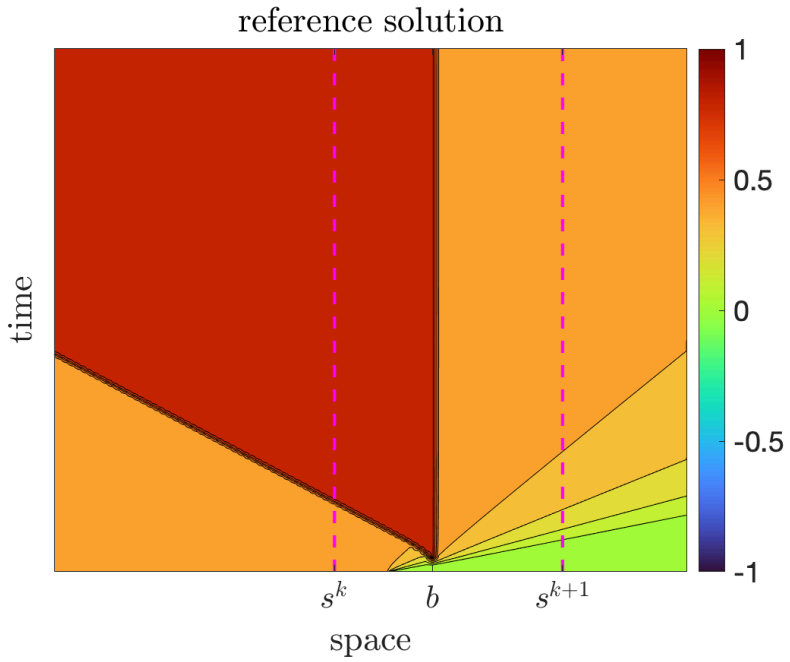


Fig. 5.17: Academic example of a traffic simulation where a bottleneck causes a traffic congestion between two sensors. Reference solution where the bottleneck is known by the model: the queue naturally propagates backwards from the locus of the bottleneck.

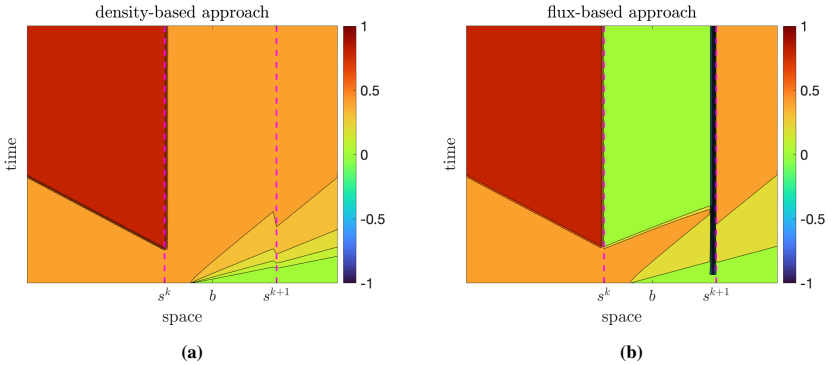


Fig. 5.18: Solution for the bottleneck simulation from Figure 5.17 with density- and flux-based approach where the model receives data from the sensors only. Both simulations locate the queue at the upstream sensor and propagate the queue backwards. **(a)** The solution obtained with the density-based approach correctly keeps the high-density value downstream of the sensor k. **(b)** The solution obtained with the flux-based approach empties the street between the two sensors, hence causing an incompatibility (black vertical line) at the interface $s^k|s^{k+1}$.

At $x = s^{k+1}$ whatever approach is used, the solution is the same and the traffic keeps going with the same dynamics.

Figure 5.18**(b)** show instead the result obtained by the *flux-based* approach. As before, when the queue reaches $x = s^k$ the sensor registers a null flux there, and a queue starts back-propagating. Since the flux through the sensor is null, no one moves from the sensor on, and the space between the sensors becomes empty in a short time. At this point, the flux data at the right sensor becomes incompatible with the traffic condition (the road is empty but the sensor perceives moving vehicles). Therefore, a negative density appears at $x = s^{k+1}$. For $x > s^{k+1}$ traffic dynamics restart correctly with the measured flux.

In conclusion, in this case, the density-based approach is preferable since it catches with better precision the real scenario. In particular, this result is achieved since the density-based approach actually puts in the simulation additional information about the system other than the naked flux data, *i.e.* the discrimination between free and congested scenarios.

Real setting

In this section, we consider real sensor data provided by Autovie Venete (cf. Section 5.2). The major difference with respect to the previous test is that now we deal with a three-lane highway and two classes of vehicles (light and heavy), with coupled dynamics. The LWR-like model is generalised to this case by a system of PDEs

$$\begin{cases} \partial_t \rho_{\text{L}}^k + \partial_x f_{\text{L}}(\rho_{\text{L}}^k, \rho_{\text{H}}^k) = 0, \quad x \in S_k, \quad t \in (t_0 - \Delta t_{\text{PAST}}, t_0) \\ \partial_t \rho_{\text{H}}^k + \partial_x f_{\text{H}}(\rho_{\text{L}}^k, \rho_{\text{H}}^k) = 0, \quad x \in S_k, \quad t \in (t_0 - \Delta t_{\text{PAST}}, t_0) \end{cases} \tag{5.11}$$

where ρ_{L}, f_{L} and ρ_{H}, f_{H} are the density and flux of light and heavy vehicles, respectively. Equation (5.11) is complemented with initial and boundary conditions as in (5.7). Moreover, the two classes of vehicles do not share the road in the same manner, being heavy vehicles not allowed in the fastest lane. The coupled dynamics with uneven space occupancy is taken from [163] and we refer the reader to it for both the mathematical and numerical details. Here we just recall that we consider a *phase transition* (cf. [168, 169, 172]) due to the presence of two states of the system, see Figure 5.19:

partial-coupling phase Heavy vehicles influence the dynamics of light ones but not vice versa. Light vehicles are then mainly in the fast lane and heavy vehicles are independent of them. In this case, the two equations in the system (5.11) are partially coupled, *i.e.* f_{H} only depends on ρ_{H},

$$\begin{cases} \partial_t \rho_{\text{L}} + \partial_x f_{\text{L}}(\rho_{\text{L}}, \rho_{\text{H}}) = 0 \\ \partial_t \rho_{\text{H}} + \partial_x f_{\text{H}}(\rho_{\text{H}}) = 0 \end{cases} . \tag{5.12}$$

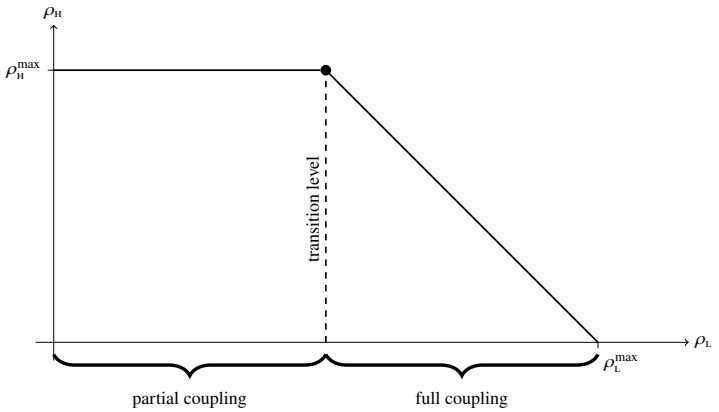


Fig. 5.19: The two phases of partial- and full-coupling model, see (5.12) and (5.11). The density of light vehicles determines if the dynamic is partially or fully coupled.

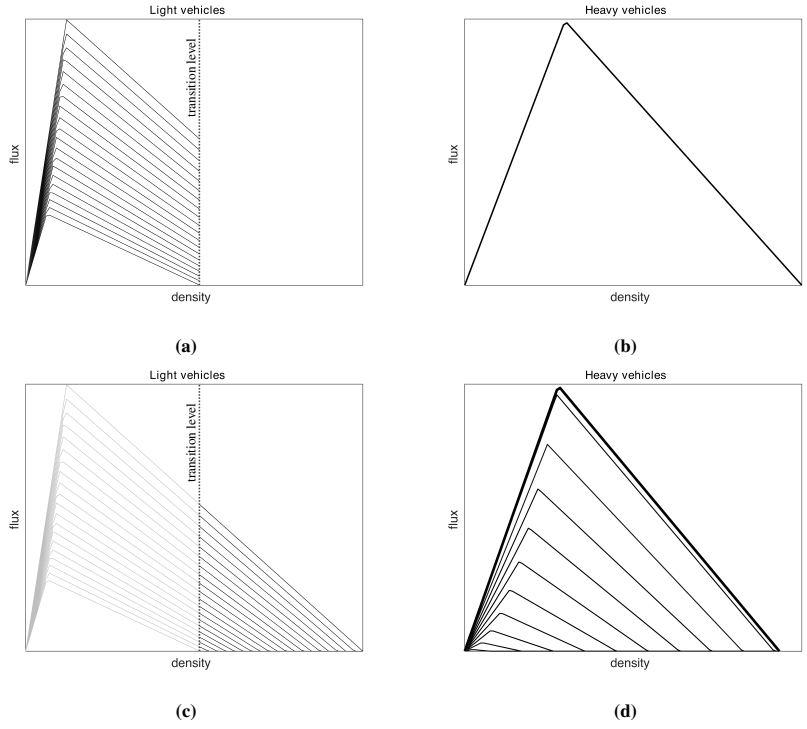


Fig. 5.20: Example of families of fundamental diagrams for light and heavy in the two phases of partial- and full-coupling. **(a)** and **(b)** represent partial-coupling phase from (5.12). **(c)** and **(d)** represent full-coupling phase from (5.11). **(a)** and **(c)** depicts the function $\rho_{\mathrm{L}} \to f_{\mathrm{L}}(\rho_{\mathrm{L}}, \rho_{\mathrm{H}})$ for several values of ρ_{H}. **(b)** depicts the single function $\rho_{\mathrm{H}} \to f_{\mathrm{H}}(\rho_{\mathrm{H}})$ and **(d)** depicts the function $\rho_{\mathrm{H}} \to f_{\mathrm{H}}(\rho_{\mathrm{L}}, \rho_{\mathrm{H}})$ for several values of ρ_{L}.

full-coupling phase Light vehicles are too much to fit the fast lane only and then invade the slow lane(s), influencing the dynamics of heavy vehicles. In this case, the two equations are fully coupled and fall in the general form of the system (5.11) .

Figure 5.20 shows the families of fundamental diagrams devised for taking into account the flux-density dependence for each class of vehicles given the density of the other class (see [163] for more details). Interestingly, there is a similarity with the multi-valued fundamental diagram sketched in Figure 5.2**(b)**. As we said, a multi-valued diagram takes into account the heterogeneity of the drivers' behaviour (inside each class of vehicles). Here, instead, the drivers of each class are homogeneous, but the presence of vehicles of the second class changes the behaviour of the drivers of the first class, thus mimicking a multi-response setting.

We conclude this section by describing a sample test on real data.

Let us consider a stretch of road of length 32 km, across two segments. A sensor is located at the interface at $x = 20.5$ km. A permanent bottleneck caused by the transition from 3 to 2 lanes is located at $x = 29.5$ km and it is modelled within the model as a change in the fundamental diagrams. In the real scenario, confirmed by direct observation of Autovie Venete personnel, the bottleneck causes congestion for heavy vehicles only, which propagates backwards and, in turn, slows down the light vehicles. As in the academic test, the model perceives the congestion only when it reaches the sensor.

Figure 5.21 shows the density and the velocity of light and heavy vehicles in the case of flux- and density-based approaches after the completion of a nowcast. If the flux-based approach is used, a piece of congestion for heavy vehicles propagates upstream of the sensor and it keeps slowing down the light vehicles (their speed is about 60 km/h), while downstream traffic is free. If instead the density-based approach is used, the dynamics are more complex because the corrections $\rho \to \rho'$ for each class correspond to a change in the fundamental diagram of the other class (cf. Figure 5.20). Downstream we observe a slowdown for both vehicle classes (velocity is about 20 km/h for heavy vehicles and 100 km/h for light vehicles) while upstream both vehicle classes are totally congested (velocity is zero). Again, the density-based approach better matches the real scenario, especially between the sensor and the bottleneck.

5.6.2 Forecast

In this section, we explore the possible advantage of estimating the incoming traffic volume for traffic forecast. The idea is to run again the simulator based on the model (5.11) from time t_0 to time $t_0 + \Delta t_{\text{FUT}}$. Conversely to the previous case, here we aim at forecasting the traffic distribution in the whole road $[x_{\text{MIN}}, x_{\text{MAX}}]$ (with no interruption at sensors), starting from the outcome of the nowcast procedure as initial condition for the densities $(\rho_{\text{L}}, \rho_{\text{H}})$. Obviously, sensor data can no be longer used here since

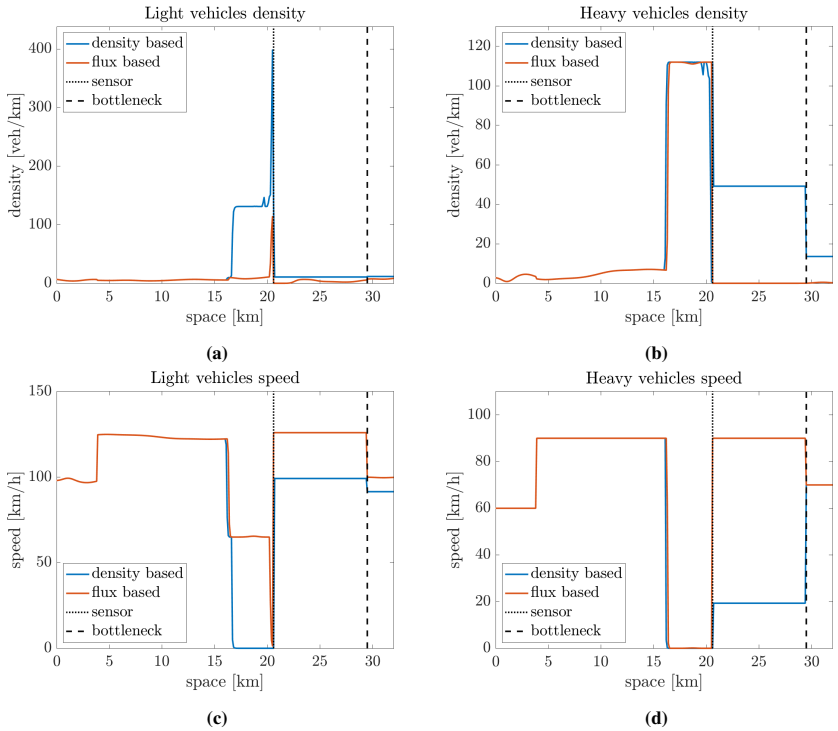


Fig. 5.21: Real example of traffic simulation where a bottleneck causes a traffic congestion in a (known) location between two sensors. **(a)** and **(c)** show the behaviour of light vehicles while **(b)** and **(d)** shows the one of heavy vehicles. As can be seen in **(a)** and **(b)**, the density obtained with the density-based approach is realistic since it fills the road downstream. **(c)** and **(d)** shows the corresponding velocities.

they are not yet available, and the problem arises which boundary conditions should be used. Ideally, one should estimate the correct future inflow and outflow at each time step Δt, but this is extremely difficult considering the high variability of the traffic dynamics. On the opposite side, the simplest solution is to assume $\rho_{\text{IN}} = 0$ so as to assume that nobody enters the road, and $\rho_{\text{OUT}} = 0$ so as to guarantee maximal outflow, but this leads to a gradual emptying of the road starting from the inflow boundary. A possible compromise is to set $\rho_{\text{OUT}} = 0$ for maximal outflow and keep a *constant* inflow, equal to the last available datum, or an average of the last minutes, or equal to a certain value predicted by a separate procedure. Here we consider the outcome of the ANN set up in Section 5.5, which estimates the traffic volume for 30 min in the future, directly injecting the flux datum in the numerical scheme.

In order to measure the error of a simulation, let us consider the density distribution $x \longrightarrow \rho^F_{\text{L,H}} \equiv (\rho^F_{\text{L}}, \rho^F_{\text{H}})$ of the traffic obtained by the forecaster for light and heavy vehicles. More formally, we are denoting with ρ^F_{L} the density $\rho^F_{\text{L}}(x, t_0 + \delta)$ at any future time $t_0 + \delta$ where $\delta \in [0, \Delta t_{\text{FUT}}]$ (resp. for heavy vehicles). The idea is hence

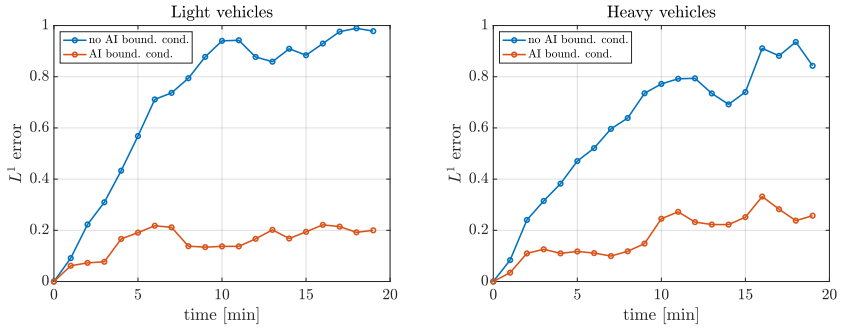


Fig. 5.22: Normalised L^1 error committed by the forecast as a function of time in the case of ANN-generated inflow boundary condition (orange line) and with constant null inflow (blue line). Results are reported for both **(a)** light and **(b)** heavy vehicles.

to compare these densities with the ones obtained by the nowcaster, if run as soon as data becomes available, *i.e.* $\forall \delta \in [0, \Delta t_{\text{FUT}}]$. If we analogously denote the density obtained by the nowcaster as $x \to \rho^F_{\text{L,H}}$, we can define the (relative) L^1-distance between the two traffic densities as follows

$$E^1_{\text{L,H}}(t) := \frac{\|\rho^F_{\text{L,H}} - \rho^N_{\text{L,H}}\|_{L^1}}{\|\rho^N_{\text{L,H}}\|_{L^1}} , \tag{5.13}$$

where

$$\|\rho\|_{L^1} := \int_{x_{\text{MIN}}}^{x_{\text{MAX}}} |\rho(x,t)|\, dx . \tag{5.14}$$

Figure 5.22 shows the error as a function of time for light and heavy vehicles separately, on a stretch of road of length 16 km with two lanes. We compare the error made by using the predicted value of the incoming flux with the one made by simply assuming a null inflow (the simplest choice). It is interesting to note that in the ANN-aided traffic prediction, the error initially increases and then stays bounded below about 30% for both light and heavy vehicles. This means that, although the traffic distribution can be shifted horizontally with respect to the real one, the total mass (*i.e.* the number of vehicles) does not differ excessively. On the contrary, using a null inflow the error rapidly increases up to 100% and then stabilises, as expected.

5.7 Conclusions

In this chapter, we have proposed a hybrid model-/data-driven method for reconstructing and predicting traffic distributions on extra-urban roads and highways. Similar to [180, 197], the idea is to use an ANN as a joining link between real data

and the mathematical model, avoiding using the latter in the training phase of the ANN. This approach allows exploiting the power of ML in extrapolating information from real-time and historical data and then passing to the model a piece of processed information that can be immediately incorporated. We have also observed that data-driven approaches based on data measured by fixed sensors can hardly extrapolate any kind of spatial information, *i.e.* information about the spatial distribution of traffic *between* sensors. This is the reason why we think that the mathematical model is essential in TSE since it catches the right causality of traffic dynamics in space and time.

References

[157] Z. (Sean) Qian, J. Li, X. Li, M. Zhang, and H. Wang. "Modeling heterogeneous traffic flow: A pragmatic approach". In: *Transportation Research Part B: Methodological* 99 (2017), pp. 183–204. DOI: 10.1016/j.trb.2017.01.011 (cit. on p. 176).

[158] A. Aw and M. Rascle. "Resurrection of "second order" models of traffic flow". In: *SIAM Journal on Applied Mathematics* 60.3 (2000), pp. 916–938. DOI: 10.1137/s0036139997332099 (cit. on p. 176).

[159] M. Barreau, M. Aguiar, J. Liu, and K. H. Johansson. "Physics-informed learning for identification and state reconstruction of traffic density". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE. 2021, pp. 2653–2658. DOI: 10.1109/cdc45484.2021.9683295 (cit. on p. 177).

[160] N. Bellomo and C. Dogbe. "On the modeling of traffic and crowds: A survey of models, speculations, and perspectives". In: *SIAM Review* 53.3 (2011), pp. 409–463. DOI: 10.1137/090746677 (cit. on p. 176).

[161] G. Bretti, M. Briani, and E. Cristiani. "An easy-to-use algorithm for simulating traffic flow on networks: Numerical experiments". In: *Discrete & Continuous Dynamical Systems series S* 7 (2014), pp. 379–394. DOI: 10.3934/dcdss.2014.7.379 (cit. on p. 176).

[162] M. Briani and E. Cristiani. "An easy-to-use algorithm for simulating traffic flow on networks: Theoretical study". In: *Networks & Heterogeneous Media* 9 (2014), pp. 519–552. DOI: 10.3934/nhm.2014.9.519 (cit. on p. 176).

[163] M. Briani, E. Cristiani, and P. Ranut. "Macroscopic and multi-scale models for multi-class vehicular dynamics with uneven space occupancy: A case study". In: *Axioms* 10.2 (2021), p. 102. DOI: 10.3390/axioms10020102 (cit. on pp. 175, 178, 199, 201).

[164] Bundesanstalt für Straßenwesen Bergisch Gladbach (BAST). *Technische Lieferbedingungen für Streckenstationen*. Tech. rep. PDF, URL:bast.de/tls. Bundesministerium für Verkehr, Bau und Stadtentwicklung (BMDV), Aug. 2012 (cit. on p. 180).

[165] G. Carlier and F. Santambrogio. "A continuous theory of traffic congestion and Wardrop equilibria". In: *Journal of Mathematical Sciences* 181.6 (2012), pp. 792–804. DOI: 10.1007/s10958-012-0715-5 (cit. on p. 174).

[166] W. Chen, F. Guo, and F.-Y. Wang. "A survey of traffic data visualization". In: *IEEE Transactions on Intelligent Transportation Systems* 16.6 (2015), pp. 2970–2984. DOI: 10.1109/TITS.2015.2436897 (cit. on p. 176).

[167] A. H. Chow, Y. Li, and K. Gkiotsalitis. "Specifications of fundamental diagrams for dynamic traffic modeling". In: *Journal of Transportation Engineering* 141.9 (2015), p. 04015015. DOI: 10.1061/(ASCE)TE.1943-5436.0000781 (cit. on p. 176).

[168] R. M. Colombo. "Hyperbolic phase transitions in traffic flow". In: *SIAM Journal on Applied Mathematics* 63.2 (2002), pp. 708–721. DOI: 10.1137/S0036139901393184 (cit. on p. 199).

[169] R. M. Colombo, P. Goatin, and B. Piccoli. "Road networks with phase transitions". In: *Journal of Hyperbolic Differential Equations* 7.1 (2010), pp. 85–106. DOI: 10.1142/S0219891610002025 (cit. on p. 199).

[170] E. Cristiani and F. S. Priuli. "A destination-preserving model for simulating Wardrop equilibria in traffic flow on networks". In: *Networks & Heterogeneous Media* 10.4 (2015), p. 857. DOI: 10.3934/nhm.2015.10.857 (cit. on p. 174).

[171] C. F. Daganzo. "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory". In: *Transportation Research Part B: Methodological* 28.4 (1994), pp. 269–287. DOI: 10.1016/0191-2615(94)90002-7 (cit. on p. 176).

[172] M. L. Delle Monache, K. Chi, Y. Chen, P. Goatin, K. Han, J. Qiu, and B. Piccoli. "Three-phase fundamental diagram from three-dimensional traffic data". In: *Axioms* 10 (2021), p. 17. DOI: 10.3390/axioms10010017 (cit. on p. 199).

[173] W. Du, Q. Zhang, Y. Chen, and Z. Ye. "An urban short-term traffic flow prediction model based on wavelet neural network with improved whale optimization algorithm". In: *Sustainable Cities and Society* 69 (2021), p. 102858. DOI: 10.1016/j.scs.2021.102858 (cit. on p. 176).

[174] S. Fan and D. B. Work. "A heterogeneous multiclass traffic flow model with creeping". In: *SIAM Journal on Applied Mathematics* 75.2 (2015), pp. 813–835. DOI: 10.1137/140977977 (cit. on p. 176).

[175] W. Fang, W. Zhuo, J. Yan, Y. Song, D. Jiang, and T. Zhou. "Attention meets long short-term memory: A deep learning network for traffic flow forecasting". In: *Physica A: Statistical Mechanics and its Applications* 587 (2022), p. 126485. DOI: 10.1016/j.physa.2021.126485 (cit. on p. 177).

[176] A. Ferrara, S. Sacone, and S. Siri. *Freeway Traffic Modelling and Control*. Advances in Industrial Control. Springer Cham, 2018. ISBN: 978-3-319-75961-6. DOI: 10.1007/978-3-319-75961-6 (cit. on pp. 175, 176).

[177] M. Garavello and B. Piccoli. *Traffic Flow on Networks*. American Institute of Mathematical Sciences, 2006. ISBN: 1-60133-000-6 (cit. on p. 176).

[178] Y. Gu, W. Lu, L. Qin, M. Li, and Z. Shao. "Short-term prediction of lane-level traffic speeds: A fusion deep learning model". In: *Transportation Research Part C: Emerging Technologies* 106 (2019), pp. 1–16. DOI: 10.1016/j.trc.2019.07.003 (cit. on p. 177).

[179] D. Helbing. "Traffic and related self-driven many-particle systems". In: *Reviews of Modern Physics* 73.4 (2001), p. 1067. DOI: 10.1103/revmodphys.73.1067 (cit. on p. 176).

[180] M. Herty and N. Kolbe. "Data-driven models for traffic flow at junctions". In: *arXiv preprint* (2022). ARXIV:2212.08912 (cit. on pp. 177, 181, 203).

[181] Y.-L. Hsueh and Y.-R. Yang. "A short-term traffic speed prediction model based on LSTM networks". In: *International Journal of Intelligent Transportation Systems Research* 19.3 (2021), pp. 510–524. DOI: 10.1007/s13177-021-00260-7 (cit. on p. 177).

[182] A. J. Huang and S. Agarwal. "Physics-informed deep learning for traffic state estimation: Illustrations with LWR and CTM models". In: *IEEE Open Journal of Intelligent Transportation Systems* 3 (2022), pp. 503–518. DOI: 10.1109/ojits.2022.3182925 (cit. on p. 177).

[183] F. Kessels. *Traffic Flow Modelling*. EURO Advanced Tutorials on Operational Research. Springer Cham, 2019. ISBN: 978-3-319-78695-7. DOI: 10.1007/978-3-319-78695-7 (cit. on pp. 175, 176).

[184] K. Lee, M. Eo, E. Jung, Y. Yoon, and W. Rhee. "Short-term traffic prediction with deep neural networks: A survey". In: *IEEE Access* 9 (2021), pp. 54739–54756. DOI: 10.1109/access.2021.3071174 (cit. on p. 177).

[185] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Lectures in Mathematics. ETH Zurich. Birkhäuser Basel, 1992. DOI: 10.1007/978-3-0348-8629-1 (cit. on p. 196).

[186] M. J. Lighthill and G. B. Whitham. "On kinematic waves II. A theory of traffic flow on long crowded roads". In: *Proceedings of the Royal Society of London Series A* 229 (1955), pp. 317–345. DOI: 10.1098/rspa.1955.0089 (cit. on p. 176).

[187] S. Maerivoet and B. De Moor. "Cellular automata models of road traffic". In: *Physics Reports* 419.1 (2005), pp. 1–64. DOI: 10.1016/j.physrep.2005.08.005 (cit. on p. 176).

[188] E. L. Manibardo, I. Laña, and J. Del Ser. "Deep learning for road traffic forecasting: Does it make a difference?" In: *IEEE Transactions on Intelligent Transportation Systems* 23.7 (2021), pp. 6164–6188. DOI: 10.1109/tits.2021.3083957 (cit. on p. 177).

[189] A. Messner and M. Papageorgiou. "METANET: A macroscopic simulation program for motorway networks". In: *Traffic Engineering & Control* 31.8-9 (1990), pp. 466–470 (cit. on p. 176).

[190] Z. Mo, R. Shi, and X. Di. "A physics-informed deep learning paradigm for car-following models". In: *Transportation Research Part C: Emerging Technologies* 130 (2021), p. 103240. DOI: 10.1016/j.trc.2021.103240 (cit. on p. 177).

[191] S. Modi, J. Bhattacharya, and P. Basak. “Multistep traffic speed prediction: A deep learning based approach using latent space mapping considering spatio-temporal dependencies”. In: *Expert Systems with Applications* 189 (Mar. 2022), p. 116140. DOI: 10.1016/j.eswa.2021.116140 (cit. on p. 176).

[192] D. Ni, H. K. Hsieh, and T. Jiang. “Modeling phase diagrams as stochastic processes with application in vehicular traffic flow”. In: *Applied Mathematical Modelling* 53 (2018), pp. 106–117. DOI: 10.1016/j.apm.2017.08.029 (cit. on p. 175).

[193] H. J. Payne. “Model of freeway traffic and control”. In: *Mathematical Model of Public System* (1971), pp. 51–61 (cit. on p. 176).

[194] P. I. Richards. “Shock waves on the highway”. In: *Operations Research* 4.1 (1956), pp. 42–51. DOI: 10.1287/opre.4.1.42 (cit. on p. 176).

[195] T. Seo, A. M. Bayen, T. Kusakabe, and Y. Asakura. “Traffic state estimation on highway: A comprehensive survey”. In: *Annual Reviews in Control* 43 (2017), pp. 128–151. DOI: 10.1016/j.arcontrol.2017.03.005 (cit. on p. 176).

[196] R. Shi, Z. Mo, and X. Di. “Physics-informed deep learning for traffic state estimation: A hybrid paradigm informed by second-order traffic models”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 2021, pp. 540–547. DOI: 10.1609/aaai.v35i1.16132 (cit. on p. 177).

[197] R. Shi, Z. Mo, K. Huang, X. Di, and Q. Du. “A Physics-Informed Deep Learning Paradigm for Traffic State and Fundamental Diagram Estimation”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.8 (2022), pp. 11688–11698. DOI: 10.1109/TITS.2021.3106259 (cit. on pp. 176, 177, 203).

[198] D. A. Tedjopurnomo, Z. Bao, B. Zheng, F. Choudhury, and A. K. Qin. “A survey on modern deep neural network for traffic prediction: Trends, methods and challenges”. In: *IEEE Transactions on Knowledge and Data Engineering* (2020). DOI: 10.1109/tkde.2020.3001195 (cit. on p. 177).

[199] B. T. Thodi, Z. S. Khan, S. E. Jabari, and M. Menéndez. “Incorporating kinematic wave theory into a deep learning method for high-resolution traffic speed estimation”. In: *IEEE Transactions on Intelligent Transportation Systems* (2022). DOI: 10.1109/tits.2022.3157439 (cit. on pp. 174, 176).

[200] F. van Wageningen-Kessels. “Framework to assess multiclass continuum traffic flow models”. In: *Transportation Research Record* 2553.1 (2016), pp. 150–160. DOI: 10.3141/2553-16 (cit. on p. 176).

[201] F. van Wageningen-Kessels, H. Van Lint, K. Vuik, and S. Hoogendoorn. “Genealogy of traffic flow models”. In: *EURO Journal on Transportation and Logistics* 4.4 (2015), pp. 445–473. DOI: 10.1007/s13676-014-0045-5 (cit. on p. 176).

[202] H. Wang, D. Ni, Q.-Y. Chen, and J. Li. “Stochastic modeling of the equilibrium speed-density relationship”. In: *Journal of Advanced Transportation* 47 (2013), pp. 126–150. DOI: 10.1002/atr.172 (cit. on p. 175).

[203] K. Wang, C. Ma, Y. Qiao, X. Lu, W. Hao, and S. Dong. "A hybrid deep learning model with 1DCNN-LSTM-Attention networks for short-term traffic flow prediction". In: *Physica A: Statistical Mechanics and its Applications* 583 (2021), p. 126293. DOI: 10.1016/j.physa.2021.126293 (cit. on p. 177).

[204] S. Wang, J. Cao, and P. Yu. "Deep learning for spatio-temporal data mining: A survey". In: *IEEE Transactions on Knowledge and Data Engineering* 34.8 (2022), pp. 3681–3700. DOI: 10.1109/tkde.2020.3025580 (cit. on p. 177).

[205] Y. Wang, M. Zhao, X. Yu, Y. Hu, P. Zheng, W. Hua, L. Zhang, S. Hu, and J. Guo. "Real-time joint traffic state and model parameter estimation on freeways with fixed sensors and connected vehicles: State-of-the-art overview, methods, and case studies". In: *Transportation Research Part C: Emerging Technologies* 134 (2022), p. 103444. DOI: 10.1016/j.trc.2021.103444 (cit. on p. 176).

[206] J. G. Wardrop. "Some theoretical aspects of road traffic research". In: *Proceedings of the Institution of Civil Engineers* 1.3 (1952), pp. 325–362. DOI: 10.1680/ipeds.1952.11259 (cit. on p. 174).

[207] G. B. Whitham. *Linear and nonlinear waves*. John Wiley & Sons, 2011. ISBN: 9781118032954. DOI: 10.1002/9781118032954 (cit. on p. 176).

[208] W. Xiangxue, X. Lunhui, and C. Kaixun. "Data-driven short-term forecasting for urban road network traffic based on data processing and LSTM-RNN". In: *Arabian Journal for Science and Engineering* 44.4 (2019), pp. 3043–3060. DOI: 10.1007/s13369-018-3390-0 (cit. on pp. 174, 177).

[209] J. Xing, W. Wu, Q. Cheng, and R. Liu. "Traffic state estimation of urban road networks by multi-source data fusion: Review and new insights". In: *Physica A: Statistical Mechanics and its Applications* (2022), p. 127079. DOI: 10.1016/j.physa.2022.127079 (cit. on p. 176).

[210] H. Yan, L. Fu, Y. Qi, D.-J. Yu, and Q. Ye. "Robust ensemble method for short-term traffic flow prediction". In: *Future Generation Computer Systems* 133 (2022), pp. 395–410. DOI: 10.1016/j.future.2022.03.034 (cit. on p. 176).

[211] Y. Yuan, Z. Zhang, and X. T. Yang. "Macroscopic traffic flow modeling with physics regularized Gaussian process: Generalized formulations, preprint". In: *arXiv preprint* (2022). ARXIV:2007.07762 (cit. on p. 178).

[212] Y. Yuan, Z. Zhang, X. T. Yang, and S. Zhe. "Macroscopic traffic flow modeling with physics regularized Gaussian process: A new insight into machine learning applications in transportation". In: *Transportation Research Part B: Methodological* 146 (2021), pp. 88–110. DOI: 10.1016/j.trb.2021.02.007 (cit. on p. 178).

[213] H. M. Zhang. "A non-equilibrium traffic model devoid of gas-like behavior". In: *Transportation Research Part B: Methodological* 36.3 (2002), pp. 275–290. DOI: 10.1016/s0191-2615(00)00050-3 (cit. on p. 176).

[214] Z. Zhang, Y. Yuan, and X. Yang. "A hybrid machine learning approach for freeway traffic speed estimation". In: *Transportation Research Record*

2674.10 (2020), pp. 68–78. DOI: 10.1177/0361198120935875 (cit. on p. 176).

[215] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu. "LSTM network: A deep learning approach for short-term traffic forecast". In: *IET Intelligent Transport Systems* 11.2 (2017), pp. 68–75. DOI: 10.1049/iet-its.2016.0208 (cit. on p. 177).

[216] G. Zheng, W. K. Chai, and V. Katos. "A dynamic spatial–temporal deep learning framework for traffic speed prediction on large-scale road networks". In: *Expert Systems with Applications* 195 (2022), p. 116585. DOI: 10.1016/j.eswa.2022.116585 (cit. on p. 176).

Part IV
Computational biology

Overview

Chapter 6
Explainable drug repurposing approach from biased random walks

This chapter describes a novel recommender system methodology for drug repurposing. Its content, realised in collaboration with Filippo Castiglione, Christine Nardini, and Paolo Tieri, researchers of the "Digital Biology Unit" *(DBU) of the* "Istituto per le Applicazioni del Calcolo" *of the* "Consiglio Nazionale delle Ricerche" *(IAC-CNR) and with Marco Pedicini, professor at the* "Università degli Studi Roma Tre"*, was published on IEEE/ACM Transactions on Computational Biology and Bioinformatics [270]. Here I revise the notation and partially extend benchmarks.*

Abstract Drug repurposing is a highly active research area, aiming at finding novel uses for drugs that have been previously developed for other therapeutic purposes. Despite the flourishing of methodologies, success is still partial, and different approaches offer peculiar advantages. In this composite landscape, we present a novel methodology focusing on an efficient mathematical procedure based on gene *similarity scores* and *biased random walks* which rely on robust drug-gene-disease association datasets. The recommendation mechanism is further unveiled by means of the *Markov chain* underlying the random walk process, hence providing *explainability* about how findings are suggested. Performances evaluation and the analysis of a case study on *rheumatoid arthritis* show that our approach is accurate in providing useful recommendations and is computationally efficient, compared to the state of the art of drug repurposing approaches.

Keywords: Drug repurposing · Explainable artificial intelligence · Markov chain · Network medicine · Biased random walk

6.1 Introduction

Drug repurposing (DR) is emerging as an essential and potentially valuable undertaking to rapidly exploit existing and tested drugs for new uses, such as emerging and neglected diseases, as well as an alternative and convenient choice as opposed to the *de novo* drugs development. However, in the exemplary case of the recent COVID-19 pandemic, despite the high number of DR attempts (a non-exhaustive PubMed search performed on January 2022 reported more than 900 results using the keywords "covid" and "drug repurposing"), the effectiveness of DR still appeared to be low: out of over 400 drugs tested, just a few, and precisely four of them, were definitively shown to be effective, *i.e.* graded "A" (*i.e.* established effectiveness; en-

dorsement by professional societies) [219, 230, 265]. The reported success rate of ∼1% is certainly not satisfactory [239] and calls for better use of the increasingly robust data and for the development of more efficient methods for DR algorithms and processes capable of making the most out of previous knowledge.

DR approaches fuse well also with the concept of Synergistic Drug Combinations, where the aim is to find two or more active pharmaceutical ingredients that co-op well to target multiple conditions (*e.g.* [226] and [268]).

In what follows, we explain some prominent examples of current DR processes representing the starting point of our study, and we highlight some salient related limitations.

6.1.1 Brief exploration of previous relevant works

The pioneering work of Keiser *et al.* [242] introduced the possibility to infer by computational means novel drugs for neglected diseases, and suggested general computational DR systems. Since then, various sorts of computational approaches exploiting different databases and different similarity criteria have been designed to tackle the problem. In what follows we briefly report, with no claim of completeness, some of the studies that are most relevant for the work presented here (see *e.g.* [227] for a more in-depth review).

Luo *et al.* [248] exploited the concept of similarity of drugs (chemical-based) and diseases (MeSH-based [245]) to build two networks, then they linked them together exploiting data from the databases DrugBank [266] and Online Mendelian Inheritance in Man (OMIM) [218], to generate 1933 drug-disease associations among 593 drugs and 313 diseases. Finally, a random walk approach was implemented simultaneously on the two similarity networks to rank drug-disease associations and propose predictions.

Nam *et al.* [250] proposed a method involving three steps: (i) drug network reconstruction using drug-target protein associations, (ii) network reinforcement, *i.e.* a machine learning approach providing *information augmentation* by using drug-drug interaction knowledge, and (iii) a recommendation step via the generation of a score computed by a graph-based semi-supervised learning procedure. The authors identified and recommended 11 novel drugs for the case study of vascular dementia through their approach.

Ozsoy *et al.* [253] carried out DR as a recommendation process in three steps: similarity evaluation, neighbour, and disease selection, actually implementing a collaborative filtering with the possibility to integrate multiple data sources and multiple features. The method produced recommendations based on the similarity and overlap between symptoms of the diseases and the effectiveness of the drugs, hence showing better performances compared to other methods available in the literature.

6.1.2 Limitations and issues with available data

DR approaches exploit previous knowledge, data cross-linking and associations via database interoperability to infer *de novo* predictions and testable hypotheses. Such approaches leverage many different large datasets developed during the last decades, when the global knowledge of drugs and diseases properties has considerably increased [262]. However, a fast growth like this often generates an inconsistency in nomenclatures, resulting in a persistent difficulty in fusing data coming from different sources [263].

Literature reports many attempts in setting up actual standards (*e.g.* the latest World Health Organization's ICD-11 [240]), however, different usages imply different requirements, not always addressed by all evolving standards. As such, for our purposes, we refer to the common practice in the scientific community operating on DR, which currently widely employs only a few *de facto* leading open-access reference sources and standards (*i.e.* the DrugBank knowledge base for drug-target associations [266] and the DisGeNet discovery platform containing one of the largest publicly available collections of genes and variants associated with human diseases [254]). We refer to more pragmatic reviews like [227] for an in-depth analysis of the available datasets.

6.1.3 Chapter contribution

Building on previous works and with the aim of enhancing the reliability and the capabilities of preceding attempts, we here propose a Markov chain-based similarity approach that exploits available data in the form of a knowledge digraph [234] such as experimental drug-gene interactions, disease-gene associations, and drug-disease pharmacological indications.

Our approach consists of five distinctive features: (i) a careful data selection and nomenclature mapping, (ii) a database bipartite digraph design, (iii) a SparseBLAS-based data structure [4], (iv) an Ergodic Markov Process representation of the DR system, and (v) an explainable output.

A careful selection of the terms grants maximum consistency and therefore minimal loss of information, in particular when datasets are joined together. This also considerably reduces the effort spent manually curating the datasets. For this reason, we chose two *de facto* standards: `cas-number` (Chemical Abstracts Service Reference Number) for drugs [223] as provided by DrugBank, and the `UMLS` (Unified Medical Language System) identifier for diseases [221]. We converted all the involved entities from the various datasets through different dictionaries provided by MalaCards [257], OMIM [218], DrugBank [266], and many others and we assigned each drug/disease a unique integer identifier (used for evaluation purposes). This choice enables us to employ two widely used, highly curated, and up-to-date datasets – DrugBank [266] and DisGeNET [254] – granting nomenclature standardisation and interoperability.

The savvy mathematical formulation of the datasets as bipartite digraphs allows the natural construction of useful entropy-inspired similarity measures that lays the foundations for our knowledge digraph (see *e.g.* [238] where a similar approach is used to create a collaborative filtering for miRNA-disease associations).

We embed our digraph structure in the BLAS environment, where structures are stored as sparse (Boolean) matrices. This solution yields fast and reliable performances based on matrix-matrix operation, like the ability to represent connections between entities as a sequence of multiplications.

We used the resulting knowledge digraph with normalised connections to generate a *Markov-process-based* DR system. The underlying mathematical structure enforces the usage of the notion of *ergodicity*, therefore providing a mathematical proof of the stability of the recommendations with respect to small changes in the input data, see Section 6.3.2.

Finally, recommendations provided on the basis of a biased random walk make them self-explainable. Explainability in Artificial Intelligence methods (see [260]) like recommender systems is particularly useful since it allows, *e.g.* to interpret the results and provide practical hints in testing, both features strongly demanded in recent trends.

We found that all the above-mentioned characteristics are crucial to providing effective, fast, usable, and reliable results for DR.

6.1.4 Chapter organisation

The remainder of this chapter is organised as follows. In Section 6.2 we describe the databases and datasets used, their cross-mappings, the related necessary integration, and their pre-processing. Section 6.3 introduces the approach used to exploit such data (Section 6.3.1), how to build the recommender system (Section 6.3.2), and which are its parameters (Section 6.3.3). In Section 6.4 we discuss parameters tuning (Section 6.4.1) and compare the methodology's performances with four existing similar methods (Section 6.4.2). We provide further results in Section 6.5, by analysing a specific DR case study, that of rheumatoid arthritis (RA), a chronic autoimmune disease with complex aetiology and no cure to date. Finally, Section 6.6 provides a summary of the contribution and possible further developments.

6.2 Materials

The present DR approach makes use of several datasets and data sources for the reconstruction of a knowledge digraph on which the recommender system is built.

As presented in Section 6.1.2, we extracted drugs and diseases information from the two *de facto* standards DrugBank (*DB*) [266] and DisGeNET (*DGN*) [254] respectively. This helped us in finding many (five) different datasets to obtain drug–

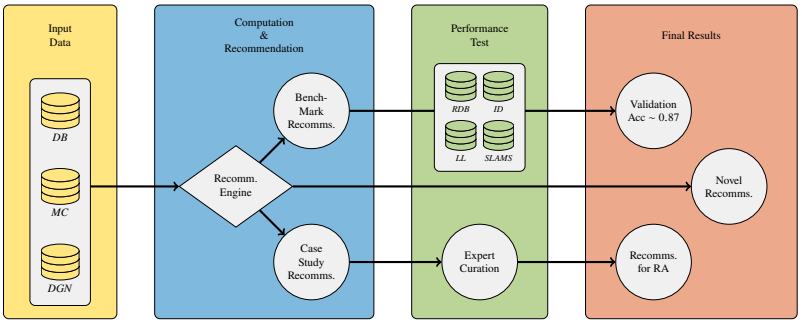


Fig. 6.1: The architecture of the proposed DR model. From left to right, the main datasets (see also Figure 6.2) are used to build the recommender engine (see also Figure 6.3). The recommender engine is validated against four benchmark datasets, hence obtaining an average accuracy of 0.87 (see also Table 6.3 and Figure 6.6) and by a manual expert curation in the specific case of Rheumatoid Arthritis, hence obtaining promising recommendations (see also Table 6.4 and Figure 6.7).

Name	Usage	#Diseases	#Drugs	#Genes	#Connections	Ref.
DB	DR construction	–	13563	4118	20279	[266]
DGN	DR construction	30293	–	26137	3261324	[254]
MC	DR construction	31642	12240	–	544857	[257]
RDB	DR validation	1229	1519	–	10563	[222]
ID	DR validation	3966	1314	–	111481	[224]
LL	DR validation	719	799	–	3250	[244]
SLAMS	DR validation	406	305	–	3871	[269]

Table 6.1: Structure of the seven datasets involved in our DR approach both as a source of knowledge and for benchmark purposes.

diseases association, namely: MalaCards (*MC*) [257], RepoDB (*RDB*) [222], iDrug (*ID*) [224], as well as data from Li and Lu's article (referred to as *LL*) [244], and data from the Similarity-based LArge-margin learning of Multiple Sources DR framework (*SLAMS*) [269]. Since the latter four (described in Section 6.2.2) were employed by other published DR methodologies, we found them suitable to be used as benchmark datasets (see Section 6.4.2). On the contrary, we used the first three (described in Section 6.2.1) to build and tune our methodology (see Section 6.3).

Table 6.1 summarises the statistics of the seven datasets reporting the number of vertices and edges of the underlying graph while in the following we briefly account for them. Their usage is summarised in Figure 6.1.

6.2.1 Main datasets and data sources

***DB* [266]** DrugBank is a pharmaceutical knowledge base that enables major advances across the data-driven medicine industry. It is provided as a drug-oriented XML, where each drug is labelled by a DrugBank unique identifier `DB` (`DBxxxxx`). We used it to extract drug-gene relations (target gene polypeptides), drug names (and `cas-number`), and their market status (approved, illicit, experimental, . . .) to provide further filtering on the final recommendations. We were able to extract 7262 `cas-number` drugs (out of 13563) connected to 4118 Genes through 19792 connections (from 1 to 305 connections per drug).

***DGN* [254]** DisGeNET is a discovery platform containing one of the largest publicly available collections of genes and variants associated with human diseases. It is provided as an SQLite DB built upon many domain, typological and associative tables. We used it to extract the relations between diseases and target genes, along with diseases' names and `UMLS` identifiers. We consider 30170 Diseases (out of 30293) connected to 21671 (out of 26137) genes through 1135045 connections (from 1 to 10161 connections per disease).

***MC* [257]** MalaCards is an integrated database of human maladies and their annotations, modelled on the architecture and richness of the popular GeneCards database of human genes. It provides the relations between drugs and diseases, already expressed as relations between `cas-number` for drugs and `UMLS` identifier for diseases. We also used it as a source of dictionaries for converting the test datasets. We filtered the entries, obtaining a total of 2088 `cas-number` (out of 12240) and 7009 `UMLS` identifiers (out of 31642) connected via 495060 links.

The diagram in Figure 6.2 shows the links between the records of the databases. We represented relations as highly sparse Boolean matrices and records as unit sparse vectors of the corresponding size.

As described, each dataset is mainly used to extract the relations between two different kinds of entities; hence it can be interpreted as a bipartite graph $G_\circ = (V_\circ, E_\circ)$ where nodes $V_\circ$ represents entities (drugs, diseases, or genes) and edges $E_\circ$ enforces connections among them. For ease of notation, we denote each graph with the acronym of the corresponding dataset, meaning we build the recommender system through three of them:

1. A Drug-Gene graph $G_{DB} = (V_{DB}, E_{DB})$ based on *DB* dataset
2. A Disease-Gene graph $G_{DGN} = (V_{DGN}, E_{DGN})$ based on *DGN* database
3. A Drug-Disease graph $G_{MC} = (V_{MC}, E_{MC})$ based on *MC* relations

The graphs, stored as sparse Boolean adjacency matrices, are built by multiplying the matrices in Figure 6.2.

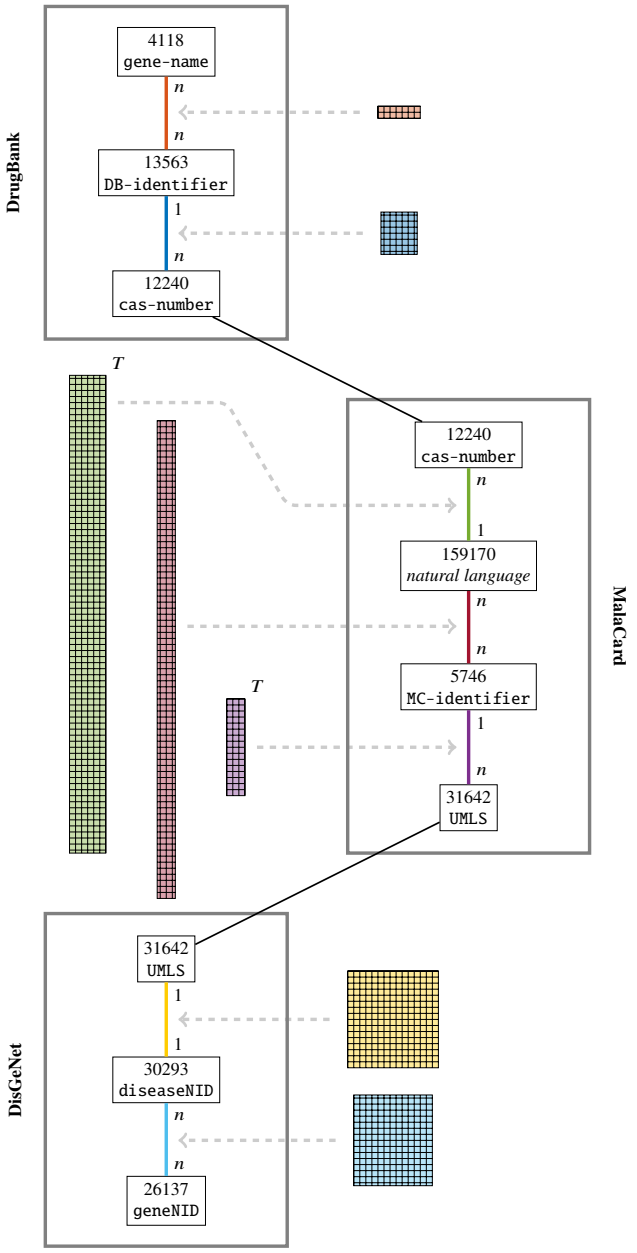


Fig. 6.2: The structure of the three datasets *DB*, *MC* and *DGN* (top to bottom). Data and connections are shown in the diagram by means of the sparse matrix representation adopted (coloured rectangles with proportional sizes), which grants an efficient and convenient solution to data storage and manipulation.

6.2.2 Databases for test and performance comparison

***RDB* [222]** The RepoDB recommender system provides a dataset extracted from DrugCentral [233] and ClinicalTrials [229]. Such dataset was originally designed as a benchmark database for drug repurposing systems testing. It counts 10563 connections between 1519 unique approved drugs (in *DB* format) and 1229 unique diseases (in `UMLS` format). We filtered these data on the drugs and diseases available in our system, therefore obtaining 2860 links between 930 drugs and 556 diseases, 49% of which are not in *MC*.

***ID* [224]** The iDrug recommender system provides a drug-diseases dataset coming from Comparative Toxicogenomics Database [232] and the gold dataset from PREDICT [237]. It is made of 111481 links between 1314 drugs (in *DB* format) and 3966 diseases. The conversion in our nomenclature produced a total of 46607 links between 813 drugs and 1335 diseases, 84% of which are not in *MC*.

***LL* [244]** The article by Li and Lu provides a sparse matrix representation of 3250 links between 799 drugs and 719 diseases, both in natural language. Data are extracted from the National Drug File - Reference Terminology. We were able to extract 1240 valid links between 673 drugs and 384 diseases, 22% of which are not in *MC*.

***SLAMS* [269]** The SLAMS recommender system provides 3871 interactions between 355 drugs and 406 diseases, both stored in natural language. Data are extracted from *DB* and from the National Drug File - Reference Terminology as for [244]. After our filtering, we obtained 1420 useful links between 242 drugs and 194 diseases, 77% of which are not in *MC*.

6.3 Methods

In the following, we describe the implementation of the recommender engine, starting from how data gathered from the sources (described in Section 6.2) are used and providing details about the working parameters.

6.3.1 Assembling available data

We employed G_{DB} and G_{DGN} to extract similarity scores σ between drugs and diseases respectively, hence obtaining two complete edge-weighted digraphs $G_{\text{drg}} = (V_{\text{drg}}, E_{\text{drg}}, \omega_{\text{drg}})$ and $G_{\text{dis}} = (V_{\text{dis}}, E_{\text{dis}}, \omega_{\text{drg}})$ with drugs/diseases as nodes and similarity scores σ as the weight of the edges, *i.e.* $\omega_E(u, v) = \sigma(u, v)$. Then, G_{MC} provides a natural way to connect V_{drg} and V_{dis} by weighting connections according to vertices degree (cf. simple random walk in Section 1.7.1), hence obtaining a single digraph G. A pictorial representation of these steps can be found in Figure 6.3 while a pseudo-code is provided in Algorithm 23.

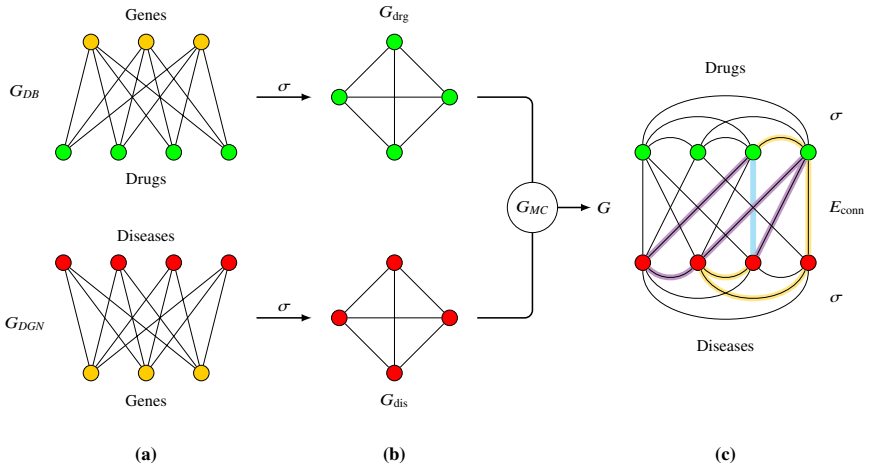


Fig. 6.3: (a) The datasets *DB* and *DGN* are processed according to the σ measure, yielding **(b)** G_{drg} and G_{dis} respectively. **(c)** These two digraphs are then connected by E_{conn} edges, obtained from *MC* with weights set to 1. The recommendation shown in cyan is obtained from $\mathcal{R}^{(\ell)}$ that relies on biased random walks like yellow and purple ones.

We define $p(t)$ as the presence ratio of the gene t in the graph $G_{DB} = (V_{DB}, E_{DB})$, namely

$$p(t) = \frac{|\{u \in V \mid t \sim_{DB} u\}|}{\|G_{DB}\|}\,, \tag{6.1}$$

where we are denoting connectivity on G_{DB} as $\sim_{DB}$ rather than $\sim_{G_{DB}}$ for ease of notation. Here, $p(t)$ is a probability distribution, *i.e.* $\sum_t p(t) = 1$, therefore, given a gene sets T, the Shannon Entropy

$$H(T) = -\sum_{t \in T} p(t) \cdot \log_2(p(t)) \tag{6.2}$$

is well-defined. We define $V_{\text{drg}} = V_{DB} \cap V_{MC}$ as the set of drugs and we compare two given drugs $u, v \in V_{\text{drg}}$ by means of their target genes sets T_u and T_v as

$$\sigma(u, v) = \frac{2H(T_u \cap T_v)}{H(T_u) + H(T_v)}\,, \tag{6.3}$$

hence weighting each gene according to its relevance within the dataset. In particular, $0 \leq \sigma(u, v) \leq 1$, where $\sigma(u, v) = 1$ if and only if $T_u = T_v$ and $\sigma(u, v) = 0$ if and only if $T_u \cap T_v = \emptyset$. We finally consider the drug similarity-weighted complete digraph $G_{\text{drg}} = (V_{\text{drg}}, E_{\text{drg}}, \omega_{\text{drg}})$ by defining the weight of the edge as

$$\omega_{\text{drg}}(u, v) \equiv \omega_{E_{\text{drg}}}(u, v) = \begin{cases} \frac{\sigma(u,v)}{\sum_{w \in V_{\text{drg}}} \sigma(u,w)} & \text{if } \sigma(u, v) > 0 \\ 0 & \text{otherwise} \end{cases}\,. \tag{6.4}$$

Algorithm 21: `EvalSigma`$(V, G_d) \mapsto \mathbf{P}$

Input: A dataset $G_d = (V_d, E_d)$ for σ-score evaluation over the vertex set V
Result: The edges weight given as transition matrix $\mathbf{P}$ for the complete graph over V
Constraint: $V \subset V_d$, $\langle V \rangle_{G_d} \cong I^{|V|}$

```
P ← 0^{|V|×|V|};
for (u ∈ V)
    Z ← 0;
    for (v ∈ V, v ≠ u)
        P_{u,v} ← σ(u, v);
        Z ← Z + P_{u,v};
    for (v ∈ V, v ≠ u)
        P_{u,v} ← P_{u,v}/Z;
    return P;
```

In particular, since $\sum_{v \in V_{\text{drg}}} \omega_{\text{drg}}(u, v) = 1 \ \forall u \in V_{\text{drg}}$, we have that the corresponding weighted adjacency matrix $\mathbf{P}_{\text{drg}}$ is right-stochastic. It follows that $\mathbf{P}_{\text{drg}}$ can be interpreted as a transition matrix[1] (cf. Section 1.7.1).

Following the same pattern on G_{DGN}, we define $G_{\text{dis}} = (V_{\text{dis}}, E_{\text{dis}}, \omega_{\text{dis}})$ as the disease similarity-weighted complete digraph with edge-weight ω_{dis}, hence making diseases and drugs comparable by means of the similarity score.

The steps leading to the similarity digraphs are summarised and generalised in Algorithm 21.

We later consider the two complete bipartite digraphs $G_{\text{drg,dis}}$ over $\mathcal{V} = V_{\text{dis}} \cup V_{\text{drg}}$, from V_{drg} to V_{dis} and vice versa. It is pretty straightforward to consider the edge weighting induced by the out-degree of each node in G_{MC}, that is

$$\omega_{\text{drg,dis}}(u, v) = \begin{cases} \frac{1}{\partial^+_{MC}(u)} & \text{if } u \sim_{MC} v \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

for $u \in V_{\text{drg}}$, $v \in V_{\text{dis}}$, and for $u \in V_{\text{dis}}$ and $v \in V_{\text{drg}}$. The corresponding adjacency matrix $\mathbf{P}_{\text{drg,dis}}$ is, again, right-stochastic (see Footnote 1). A pseudo-code representing the procedure is provided in Algorithm 22.

If we consider the complete digraph in the vertices $\mathcal{V}$, namely $(\mathcal{V}, E) = G_{\text{drg}} * G_{\text{dis}}$, it is easy to see that the three sets of edges weights ω_{dis}, ω_{drg}, $\omega_{\text{drg,dis}}$ assign positive values to disjoint set of edges. In fact, we have that the corresponding transition matrices opportunely extended on $\mathcal{V}$ are of the form

$$\mathbf{P}_{\text{drg}} = \left(\begin{array}{c|c} \omega_{\text{drg}} & 0 \\ \hline 0 & 0 \end{array}\right), \quad \mathbf{P}_{\text{dis}} = \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & \omega_{\text{dis}} \end{array}\right), \quad \mathbf{P}_{\text{drg,dis}} = \left(\begin{array}{c|c} 0 & \omega_{\text{drg,dis}} \\ \hline \omega_{\text{drg,dis}} & 0 \end{array}\right) \tag{6.6}$$

[1] Actually, some of the rows could be empty if a drug has no common target genes with the others, hence making necessary the "otherwise" clause; we get rid of this problem, also causing the matrix not to be right-stochastic, in the next steps.

Algorithm 22: `EvalConnections`$(V_1, V_2, G_d) \mapsto \mathbf{P}$

Input: A graph G_d to evaluate the connection weights between V_1 and V_2
Result: The edges weight $\mathbf{P}$ for the complete bipartite digraph over $V_1 \cup V_2$
Constraint: $V_d \subseteq V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$

```
1 V ← V_1 ∪ V_2;
2 P ← 0^{|V|×|V|};
3 for (u ∈ V)
4 |   Z ← ∂_d(u);
5 |   for (v ∈ N_{G_d}(u))                    // N_{G_d}(u) ⊆ V_2
6 |   |   P_{u,v} ← 1/z;
7 return P;
```

It is then straightforward to build a complete weighted digraph (with self loops) $G = (\mathcal{V}, \mathcal{E}, \omega_{\mathcal{E}})$ by combining such weights and by adding suitable weights to self loops: in order to preserve the transition matrix interpretation, we halve the contribution of each weight function and we set the weight $\omega_{\mathcal{E}}(v, v)$ for each vertex $v \in \mathcal{V}$ as $1 - \sum_{w \neq v} \omega_{\mathcal{E}}(v, w)$[2], namely

$$\omega_{\mathcal{E}}(u, v) = \begin{cases} 1/2 \cdot \omega_{\text{drg}}(u, v) & \text{if } u, v \in V_{\text{drg}} \\ 1/2 \cdot \omega_{\text{dis}}(u, v) & \text{if } u, v \in V_{\text{dis}} \\ 1/2 \cdot \omega_{\text{drg,dis}}(u, v) & \text{if } u \in V_{\text{drg}}, v \in V_{\text{dis}} \\ 1/2 \cdot \omega_{\text{drg,dis}}(u, v) & \text{if } u \in V_{\text{dis}}, v \in V_{\text{drg}} \\ 1 - \sum_{w \neq v} \omega_{\mathcal{E}}(u, w) & \text{if } u = v \end{cases} \tag{6.7}$$

The transition matrix of graph G obtained according to the weights from (6.7) represents the base of our DR. In the following, we will denote it as $\mathcal{R}$, since it holds all the information we know from the very first step of our DR. The complete procedure to create $\mathcal{R}$ is provided in Algorithm 23.

6.3.2 The core of the recommender system

In this section, we exploit the transition matrix property of $\mathcal{R}$ to model our system as a recommender for the most probable destination of fixed-length biased random walks.

We recall from Section 1.7.1 a ℓ-length $\mathcal{R}$-biased random walk over $\mathcal{V}$ being a sequence of $\ell + 1$ nodes

[2] It is a good practice to reduce each positive entry $\omega_E(u, v)$ by a factor $o(\min\{\omega_{\mathcal{E}}(e) \; \forall e \in \mathcal{E} \mid \omega_{\mathcal{E}}(e) > 0\})$ before evaluating $\omega_{\mathcal{E}}(u, u)$. This operation gets rid of any numerical issue caused by rounding operations on small floating-point values, hence ensuring no negative value is assigned to $\omega_{\mathcal{E}}(u, u)$.

Algorithm 23: `AssembleData`$(G_{DB}, G_{DGN}, G_{MC}) \mapsto \mathcal{R}$

Input: The dataset graphs G_{DB}, G_{DGN} and G_{MC}
Result: The edges weight $\mathcal{R}$ for the complete digraph $G = (\mathcal{V}, \mathcal{E}, \omega_{\mathcal{E}})$

$V_{\text{drg}} \leftarrow V_{DB} \cap V_{MC}$;
$V_{\text{dis}} \leftarrow V_{DGN} \cap V_{MC}$;
$\mathcal{V} \leftarrow V_{\text{drg}} \cup V_{\text{dis}}$;
$\mathbf{P}_{\text{drg}} \leftarrow$ `EvalSigma`(V_{drg}, G_{DB});
$\mathbf{P}_{\text{dis}} \leftarrow$ `EvalSigma`$(V_{\text{dis}}, G_{DGN})$;
$\mathbf{P}_{\text{drg,dis}} \leftarrow$ `EvalConnections`$(V_{\text{drg}}, V_{\text{dis}}, E_{MC})$;
$\mathbf{I}_{\text{drg}} \leftarrow \mathbf{1}^{|\mathcal{V}|\times|V_{\text{drg}}|}$;
$\mathbf{I}_{\text{dis}} \leftarrow ((\mathbf{1}^{|\mathcal{V}|\times|V_{\text{dis}}|})^{\intercal})^{\intercal}$; // *i.e.* filled diag. is from bottom right corner
$\mathcal{R} \leftarrow \mathbf{P}_{\text{drg,dis}} + \mathbf{I}_{\text{drg}} \times \mathbf{P}_{\text{drg}} \times (\mathbf{I}_{\text{drg}})^{\intercal} + \mathbf{I}_{\text{dis}} \times \mathbf{P}_{\text{dis}} \times (\mathbf{I}_{\text{dis}})^{\intercal}$;
$\epsilon \leftarrow o(\min\{w \in \mathcal{R} \mid w > 0\})$;
for $(u, v \in \mathcal{V}^2)$
  $\mathcal{R}_{u,v} \leftarrow \max\{{}^1\!/_2(\mathcal{R}_{u,v} - \epsilon), 0\}$;
for $(u \in \mathcal{V})$
  $\mathcal{R}_{u,u} \leftarrow 1 - \sum\{\mathcal{R}_{u,v} \mid v \in \mathcal{V}\}$;
return $\mathcal{R}$;

$$X = (\overbrace{x_0}^{\text{source}}, \dots, \overbrace{x_\ell}^{\text{dest}}), \quad x_i \in \mathcal{V} \tag{6.8}$$

sampled with the probability distribution given by $\mathcal{R}$ according to (1.33), *i.e.*

$$\mathbb{P}\{x_{i+1} = v, \mid x_i = u\} = \mathcal{R}_{u,v} \, . \tag{6.9}$$

We define the ℓ-th recommendation $\mathcal{R}^{(\ell)}$ as

$$\mathcal{R}^{(\ell)}_{u,v} = \mathbb{P}\{x_\ell = v, \mid x_0 = u\} \, , \tag{6.10}$$

that, according to (1.39), can be evaluated as

$$\mathcal{R}^{(\ell)} = \mathcal{R}^{\ell} = \overbrace{\mathcal{R} \times \cdots \times \mathcal{R}}^{\ell\text{-times}} \, . \tag{6.11}$$

Given a percentage of drugs $0 < \mathfrak{p} < 1$ we want to recommend, we define the drug recommender system $\mathfrak{R}^{\mathfrak{p}}_{\ell} \equiv \mathfrak{R}(\mathcal{R}, \ell, \mathfrak{p})$, as the map:

$$\begin{aligned} \mathfrak{R}^{\mathfrak{p}}_{\ell} : V_{\text{dis}} &\to V^{p}_{\text{drg}} \\ u &\mapsto \text{argmax}^{p}_{v \in V_{\text{drg}}}(\mathcal{R}^{(\ell)}_{u,v}) \, , \end{aligned} \tag{6.12}$$

where $p = \lfloor \mathfrak{p} \cdot |V_{\text{drg}}| \rfloor$ and $\text{argmax}^{p}_{\circ}(f(\circ))$ are the pre-images $\circ$ of the first p maximum values of $f(\circ)$.

The definition of the recommender system also makes the methodology self-explainable. In fact, given a recommendation r for a disease d, we can query the system to retrieve which paths contribute most to the novel connection itself, *i.e.* the heaviest paths $d = w_0\, w_1 \ldots w_\ell = r$.

We can also define an analogous *disease* recommender system $\mathfrak{R}^{\mathfrak{p}}_{\ell} : V_{\text{drg}} \to V^{p}_{\text{dis}}$ that, given a specific drug, unveils its relationship with other diseases. Due to the asymmetric nature of the argmax operator and of the matrix $\mathcal{R}$, the recommendations generated by the two methodologies may differ *i.e.* $\mathfrak{R}^{\mathfrak{p}}_{\ell}$ is not symmetric. In fact, given a disease $d \in V_{\text{dis}}$, we can have $r \in \mathfrak{R}^{\mathfrak{p}}_{\ell}(d)$ for some drug $r \in V_{\text{drg}}$ while $d \notin \mathfrak{R}^{\mathfrak{p}}_{\ell}(r)$. In other words, it could happen that the DR does not recommend a disease d to be treated via a specific drug r even if r was recommended for the treatment of that specific disease d.

6.3.3 The parameters

Assuming $\mathcal{R}$ as fixed, there are two parameters in $\mathfrak{R}^{\mathfrak{p}}_{\ell}$: the length of the walk $\ell > 1$ and the percentage of recommendations $0 < \mathfrak{p} < 1$.

In order to provide an upper bound for ℓ we tackle the task to determine the argmax as a Markov process problem, where the graph G forms the basis for an ℓ-step Markov chain $\mathcal{G}$ (see Section 1.7).

In our experiments, $\mathcal{G}$ is irreducible. However, in general, this assumption can be made without loss of generality. In fact, if $\mathcal{G}$ is not irreducible, then $G - \{e \in \mathcal{E} \mid \omega_{\mathcal{E}}(e) = 0\}$ is disconnected and hence we can study its components individually, hence splitting $\mathcal{G}$ into independent irreducible Markov chains.

According to (6.7), for any state $u \in \mathcal{V}$ we have $\mathcal{R}_{u,u} > 0$ and therefore $\mathcal{G}$ is aperiodic (and positive recurrent). We recall from Section 1.7.3 that an irreducible, aperiodic and positive recurrent Markov chain is said to be Ergodic and that Ergodic chains admit a unique stationary distribution $\boldsymbol{\pi}$. In other words, we have that

$$\mathcal{R}^{(\ell)} = \mathcal{R}^{\ell} \xrightarrow{\ell \to \infty} \mathbf{\Pi} \tag{6.13}$$

and, in general,

$$\forall \epsilon > 0,\ \exists \bar{\ell} < \infty \quad s.t. \quad \|\mathcal{R}^{(\ell)} - \mathbf{\Pi}\| < \epsilon \quad \forall \ell \geq \bar{\ell}. \tag{6.14}$$

Then, for small $\epsilon > 0$, we have that $\bar{\ell}$ is an upper bound for ℓ since $\mathfrak{R}^{\mathfrak{p}}_{\bar{\ell}}$ would give the same recommendation independently from the disease considered, hence being uninformative for repurposing scope.

However, the convergence to a unique stationary distribution $\boldsymbol{\pi}$ enforces the stability of the method. In fact, slightly modifying $\mathcal{R}$ (*i.e.* partially modifying the initial conditions) we are expected to reach a *similar* $\boldsymbol{\pi}'$, *i.e.* $\|\boldsymbol{\pi} - \boldsymbol{\pi}'\|$ is negligible.

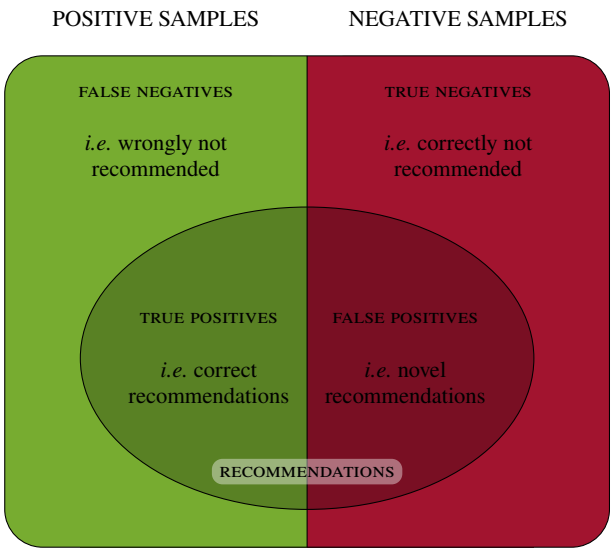


Fig. 6.4: Confusion matrix associated with the DR recommender system. $\mathfrak{R}_\ell^{\mathfrak{p}}$ recommendations are represented by the shaded ellipse. False positives (shaded red) are to be minimised but, concurrently, they represent also novel recommendations.

6.4 Performance evaluation

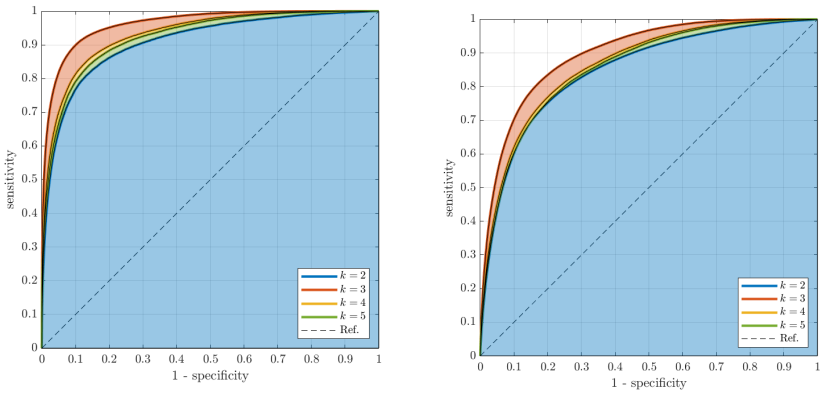
In this section, we provide an analysis of our method by means of the commonly used performance indicators based on the confusion matrix obtained from the method's execution. We firstly analyse the method on its own, hence providing parameters tuning by means of receiver operating characteristic (ROC) curves. Later, we apply our methodology to cross-validation datasets provided by other literature recommender systems.

6.4.1 Parameters tuning

As pointed out in Section 6.3.3, the recommender system $\mathfrak{R}_\ell^{\mathfrak{p}}$ can be tuned by means of two parameters:

(*i*) The length of the Markov Process $1 < \ell \ll \bar{\ell}$ (discrete)
(*ii*) The recommendations percentage $0 < \mathfrak{p} < 1$ (continuous)

One of the most used tools in parameter tuning is the ROC curve (true-positive rate vs. false-positive rate) constructed via the confusion matrix (cf. Figure 6.4) at the varying of the parameters. The objective is to reduce the false-positive rate. As it is common practice in the recommender systems literature, even if little or no *a priori* information on the ground truth (GT) negatives is available (the absence of a



(a) **(b)**

Fig. 6.5: The ROC curves obtained by the recommender system with various path lengths ℓ. The axis represents false- and true-positive ratios. Curves are sampled with a rate of $\Delta\mathfrak{p} = 0.01$. Numeric values of the AUC (integral of the curve) are reported in Table 6.2. **(a)** true- and false-positive ratios are evaluated with regard to the complete dataset (unweighted). **(b)** true- and false-positive ratios are firstly evaluated for each drug independently and then the average is computed (weighted).

ℓ	unweighted mean AUC	weighted mean AUC
2	0.908833	0.847932
3	0.961536	0.897935
4	0.931982	0.863702
5	0.924096	0.856561

Table 6.2: AUC of the ROC curves from Figure 6.5. Indicators are evaluated *w.r.t.* the complete dataset (unweighted) or drug-by-drug and then averaged (weighted). Values are computed via the Newton-Cotes numerical approximation formula.

given recommendation might be due to its effectiveness not being assessed, yet), it is assumed that only a small fraction of the negatives are actually to be recommended.

For small values of ℓ, we can then build a ROC curve on the parameter $\mathfrak{p}$, varying on $0 \leq \mathfrak{p} \leq 1$. Figure 6.5 represents such ROC curves with $1 < \ell \leq 5$ and Table 6.2 reports the corresponding area under the curve (AUC). Results are listed with two indicators: *unweighted* rates are evaluated with regard to the complete dataset (without taking into account the difference between the drugs); *weighted* rates, on the contrary, are first evaluated regarding the drugs individually and then averaged amongst them.

Weighted AUC values being smaller than unweighted ones (as expected) implies that drugs associated with fewer *a priori* known-recommendations are more unreliable in the system. In fact, they have a stronger negative impact on the mean accuracy than those with more recommendations available.

According to ROCs, the best choice is $\ell^\star = 3$. Hence, recommendations derive from exactly four random walks patterns of length 3 $X = (x_0, x_1, x_2, x_3)$ sampled from one of the following

$$\begin{array}{ll} V_{\text{drg}} \times V_{\text{drg}} \times V_{\text{drg}} \times V_{\text{dis}} & V_{\text{drg}} \times V_{\text{drg}} \times V_{\text{dis}} \times V_{\text{dis}} \\ V_{\text{drg}} \times V_{\text{dis}} \times V_{\text{drg}} \times V_{\text{dis}} & V_{\text{drg}} \times V_{\text{dis}} \times V_{\text{dis}} \times V_{\text{dis}} \end{array} \tag{6.15}$$

and nodes from the same set can potentially coincide.

With $\ell = \ell^\star$, we evaluate the best value of $\mathfrak{p}$ as the parameter generating the furthest point from the reference diagonal

$$\underset{\text{params}}{\text{argmax}}\{\text{sensitivity} + \text{specificity}\}\,, \tag{6.16}$$

hence, obtaining $\mathfrak{p}^\star = 12.5\%$. In our setting, such a value corresponds to 213 recommendations per disease.

Analogous results are obtained also with leave-one-out 10-fold cross-validation tests, where the AUC results in a mean value of 0.930174 and 0.862815 for weighted and unweighted approaches respectively, both with a standard deviation of order 10^{-4}.

6.4.2 Comparison with other methods

To provide more insightful information, we compare our method against the four benchmark datasets *RDB*, *ID*, *LL*, and *SLAMS* introduced in Section 6.2. To keep recommendations consistent with our method, we restricted such datasets by filtering the diseases and the drugs available in our training and then we collect the result regarding their connections. In situations like these ones, two main approaches are possible: (i) a gentle re-tuning of the method with the new dataset in order to prove the model's flexibility to different data or (ii) a direct usage of the dataset as a cross-validation set. While the first approach consists in performing parameter analysis again on the novel dataset, hence *forgetting* the original set of data and therefore taking the best results of the methodology over the benchmarks, the second approach – the one we adopted – tests the benchmarks over the parameter gathered from the original dataset. In this sense, we compared the recommendations our method proposes with the ground truth obtained from the four restricted datasets. However, to keep the environment of the test as close to reality as possible, we did not restrict our method's recommendation set to the drugs available in each benchmark dataset, meaning that we made our method infer no information about the validation set it was tested against; if this was the case, in fact, sensitivity and specificity values would grow above $(0.99, 0.9)$ due to the high number of recommendations, hence being uninformative.

In Figure 6.6 we report, with respect to the Unweighted-mean ROC curve, the true- and false-positive rates achieved on the benchmark datasets with $\ell = \ell^\star$.

Dataset	AUC	Accuracy	Sensitivity	Specificity	Shared recoms
RDB	0.875031	0.8417	0.660791	0.875903	51%
ID	0.873807	0.7811	0.486318	0.881105	16%
LL	0.852050	0.8415	0.646048	0.853405	78%
SLAMS	0.873245	0.7701	0.469907	0.875486	23%
MC	0.892347	0.899491	0.899547	0.962165	100%

Table 6.3: $\mathfrak{R}_3^{\mathfrak{p}}$ AUCs and performance indicators for the benchmark datasets (cf. Figure 6.6). Indicators are evaluated considering the best parameter $\mathfrak{p}^\star = 12.5\%$. Results for the main dataset are reported as reference (*MC*).

Corresponding AUCs and numeric indicators extracted with $\mathfrak{p}^\star$ are reported in Table 6.3.

As it is expected, better sensitivity and specificity results are obtained on *LL* and *RDB* since they share more than 50% of the recommendation with our training set *MC* (cf. Section 6.2.2). If we consider *SLAMS* and *ID*, while having less than 1/4 of common data with *MC*, they achieve anyway a comparable accuracy and specificity. Such comparable results provide a further clue of our method's stability.

6.5 Case study: rheumatoid arthritis

In addition to the performances assessed in Section 6.4, we here explore in more detail the results over an exemplar non-communicable disease (NCD), *i.e.* rheumatoid arthritis (RA), a worldwide threat associated with spreading chronic inflammation [249]. NCDs provoke, in fact, more than 44 million deaths per year [217] [261] and it is estimated that RA, amongst the others, affects around 1% of the world population [267]. RA's incidence and representativeness make it an interesting case study to explore the relevance of the results offered by the recommender system also being its aetiology complex and not fully elucidated, since it includes both genetic and environmental factors [259], [252].

Standard therapy (true positives) for RA was extracted from the 2021 update of the American College of Rheumatology (ACR, [236]). Then, the relevance of the findings towards potential clinical translation was manually curated for the top-ranking novel results, using two sources of knowledge to deepen the information retrieved by each entry. The first source is represented by PubMed [255], the most comprehensive medical database. The second one is the well-known repository ClinicalTrials [229] collecting concluded and ongoing clinical trials. In both cases, the search terms were represented by the disease (rheumatoid arthritis) and the drug, as recovered by the recommender system.

The top ten novel (*w.r.t.* the dataset) recommendations are reported in Table 6.4. Score, drug name and CAS number are the output of the recommender system,

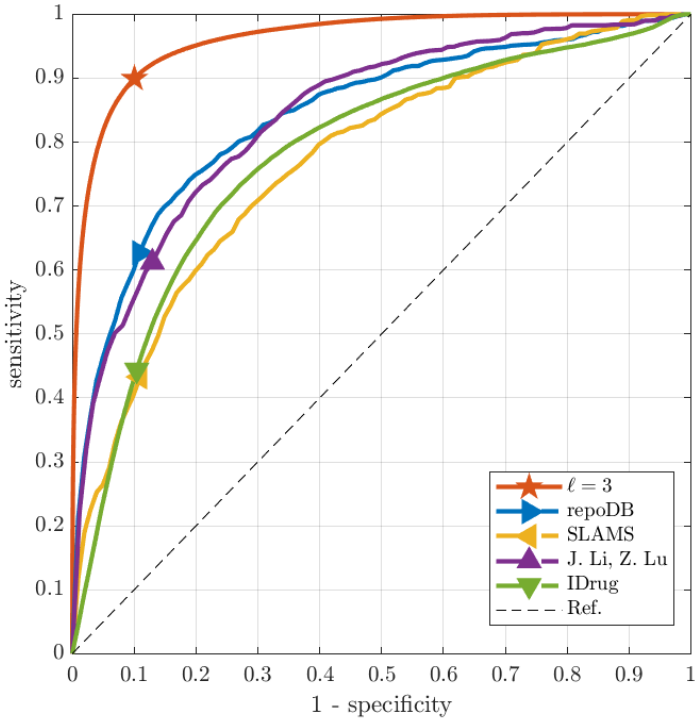


Fig. 6.6: ROC curves obtained by testing $\mathfrak{R}_3^{\mathfrak{p}}$ against the benchmark datasets. Indicators are evaluated *w.r.t.* the best setting $\mathfrak{p}^\star = 12.5\%$ and plotted over corresponding ROCs (numeric values are reported in Table 6.3). Reference unweighted ROC is plotted as well (cf. Figure 6.5(**b**)).

while clinical trials and related publications are the results of our manual analysis of the findings. Recommendations are divided according to the associated evidence (publication or clinical study), into three sets:

approved Are drugs already employed in the literature (see [236]) but not present in our training set *MC*. Being able to predict them offers a measure of the validation of the system itself.

potential Are experimentally promising drugs according to recent publications (reported in the table as well). They appear to be mainly drugs whose validation in clinical trials is not (yet) engaged, but whose usage in animal models (mostly collagen-induced arthritis – CIA – rodents) has recently offered promising results. This is the case for *Gemcitabine* (`95058-81-4`) [231] and *Lenalidomide* (`191732-72-6`) [247], whose identification by our system is particularly interesting due to their recent developments (*i.e.* after 2017).

Score [%]	Drug Name	cas-number	# Trials	Relevance
3.8830	Dihydrofolic Acid	4033-27-6	0	open
3.3545	Folic acid	59-30-3	307	approved
3.0627	Cyclophosphamide	50-18-0	7	approved
2.1424	Gemcitabine	95058-81-4	0	potential [231]
2.0943	Doxorubicin	23214-92-8	0	open
2.0700	Tretinoin	302-79-4	2	open
1.9034	Farletuzumab	896723-44-7	0	potential [235]
1.8320	Cisplatin	15663-27-1	0	open
1.7145	Lenalidomide	191732-72-6	0	potential [247]
1.7101	Docetaxel	114977-28-5	0	open

Table 6.4: Top ten drug recommendations for RA with $\ell = \ell^\star = 3$ iterations of the recommender system. Score, drug name and CAS number are outputs of the recommender system, while clinical trials and related publications are the results of our manual curation of the findings. Recommendations are divided into *approved* (already employed), *potential* (experimentally promising) and *open* (little or no recent works). The most recent publication is reported for each *potential* result, while [236] serve as a reference for *approved* ones.

open Are the ones supported by no hints of recent/ongoing related investigation. They could potentially be the most important recommendations of the system, providing a starting point for novel studies. However, they should be validated by pharmacologists, a topic which is out of the scope of the present work.

Further inspection can be carried out thanks to the explainability of the output provided by the Markov Process representation of the DR system. As an example, Figure 6.7 reports the 25 paths of length $\ell = \ell^\star$ that are contributing most to *Gemcitabine* novel recommendation, hence giving hints on the connections found. The connection with the highest value is $\{\text{RA} \rightarrow \text{Fludarabine} \rightarrow \text{Clofarabine} \rightarrow \text{Gemcitabine}\}$. More in general, all these 25 connections share the same few drugs (14) and diseases (5) with an interesting relevance in Fludarabine (`cas-number` = `21679-14-1`, present in five paths), Fluorouracil (`cas-number` = `51-21-8`, present in nine paths), and Methotrexate (`cas-number` = `59-05-2`, present in nine paths); these paths consists in fact in 21 out of the 25 total (do note that such drugs are present in the same path twice). All of these anti-neoplastic drugs can be found at a distance of one from RA, meaning that they are present in *MC* dataset.

The category *open*, having no current support in the literature, can represent both (expected) noise offered by such an automated system building on a meaningful but limited database or the most innovative opportunities. In this sense, the explainability of the recommendation represents a significant added value, offering pharmacologists and rheumatologists a clear starting point for further consideration as well as the feasibility of (early) translation into clinical testing.

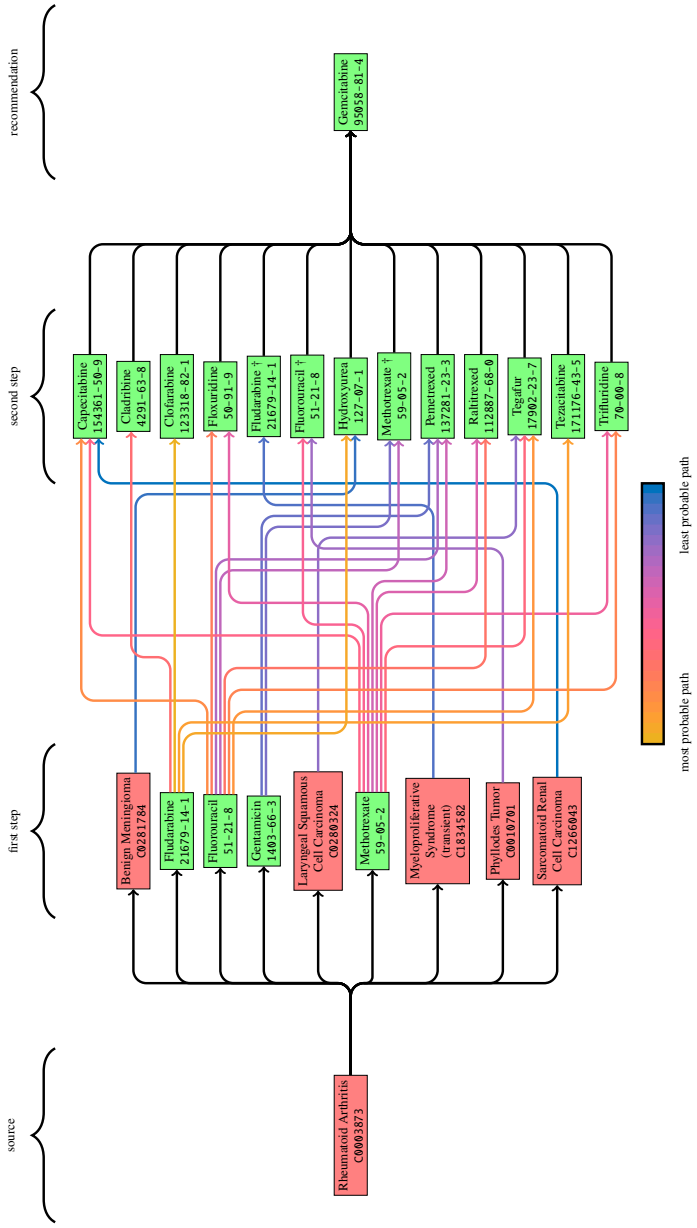


Fig. 6.7: Depiction of the recommendation process of the drug Gemcitabine in the case study of RA (top 25 path contributions shown, coloured from yellow, as the most probable path, to blue, as the least probable one). From left to right, the disease RA (source) is linked in the first step of the process with similar diseases (in red) according to σ-scores and approved drugs (in green) through known associations. Then, a second step is performed and a different set of drugs is reached (some of the nodes are returning from the first step, marked with †). Finally, the paths reach Gemcitabine with a third (and last, $\ell = 3$) step, generating the recommendation.

6.6 Discussion and conclusions

Drug recommendation processes encompass experimental and computational approaches, with the latter potentially leveraging on computational resources as well as algorithmic advances, especially in the area of machine learning [256]. The power and flexibility of such tools created high expectations in the field of *personalised medicine*, a twenty years-long effort in the evolution of the medical paradigm [251], whose latest interpretations rely on machine learning for applications ranging from diagnosis to therapy. In this setting, however, high-level transparency and accountability are mandatory, hence promoting explainable machine learning methods [264].

In this perspective, we here proposed a similarity-based approach that exploits established associative knowledge on experimental drug-gene molecular interactions, disease-drug and disease-gene associations, and pharmacological (disease-drug) indications to build a predictive DR system.

The way the proposed approach is built grants several advantages. Amongst the others, one is the standardisation, interoperability, quality and up-to-date information obtained thanks to the choice of robust, binary-type data. Moreover, the use of associative binary data (*i.e.* drug-disease, gene-disease, drug-gene) allows to overcome the problem of the trustworthiness and sensibility to experimental settings of quantitative data (such as *e.g.* gene expression data, valuable and extremely refined but also sensitive to noise [258]), and to take advantage of prior knowledge characterised by high robustness, reliability and stability. Another key feature is represented by the generation of explainable prediction, which can be obtained, once again, since the choice made here to integrate several binary associative datasets from reliable sources (DrugBank, DisGeNET, MalaCards) allowed to build a robust and well-grounded set of data that the recommender system is able to work on with a transparent approach. The output is hence completely traceable, from the processing of the initial input sources (diseases, drugs) up to the formulation of the drug repurposing proposal, hence promoting it as an explainable artificial intelligence method.

On the other hand, typical issues of associative data generate DR shortcomings, including noisy recommendations (*e.g.* false positive interactions) or incompleteness in the datasets (false negative interactions). Here, the incorporation of sources containing *negative* data (*e.g.* experimentally validated non-interacting pairs such as the Negatome for protein-protein interactions [220]) or important pharmacological side effects that impair drug usage (*e.g.* data from SIDER [243]) may help in refining the output for practical, clinical use. Multiple, redundant or complementary data sources (*e.g.* the Drug Repurposing Knowledge Graph [241]) will also help in this direction to cover missing information as well as possible.

In future implementations, we plan to extend the capability of managing information about drug–drug interactions, hence performing a step forward in the problem of synergistic drug combinations while tackling the repurposing proposals issues. To this scope, we plan to include the resources for drug synergy (*e.g.* DrugCombDB [246]) and adversary effects (*e.g.* from DrugBank).

We also plan to include micro-RNA interactions, a growing field of interest in diagnostic and therapeutic pharmacology (see [225]), to further strengthen our

knowledge-graph; in fact, miRNA plays an important role similar to target genes both for what concerns diseases (see [238]) and drugs (see [228]).

References

[217] D. Abegunde and A. Stanciole. "An estimation of the economic impact of chronic noncommunicable diseases in selected countries". In: *World Health Organization, Department of Chronic Diseases and Health Promotion* 2006 (2006). RG:253888397 (cit. on p. 229).

[218] J. S. Amberger and A. Hamosh. "Searching Online Mendelian Inheritance in Man (OMIM): A Knowledgebase of Human Genes and Genetic Phenotypes". In: *Curr Protoc Bioinformatics* 58 (June 2017), pp. 1–1. DOI: 10.1002/cpbi.27 (cit. on pp. 214, 215).

[219] *Repurposing drugs to treat COVID-19: interview with David Fajgenbaum, 2021*. URL:ascodaily.libsyn.com (cit. on p. 214).

[4] L. S. Blackford, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, et al. "An updated set of basic linear algebra subprograms (BLAS)". In: *ACM Transactions on Mathematical Software* 28.2 (2002). NIST:50982, pp. 135–151 (cit. on pp. 20, 215).

[220] P. Blohm, G. Frishman, P. Smialowski, F. Goebels, B. Wachinger, A. Ruepp, and D. Frishman. "Negatome 2.0: a database of non-interacting proteins derived by literature mining, manual annotation and protein structure analysis". In: *Nucleic Acids Res* 42.Database issue (Jan. 2014), pp. 396–400. DOI: 10.1093/nar/gkt1079 (cit. on p. 233).

[221] O. Bodenreider. "The Unified Medical Language System (UMLS): integrating biomedical terminology". In: *Nucleic Acids Res* 32.Database issue (Jan. 2004), pp. D267–270. DOI: 10.1093/nar/gkh061 (cit. on p. 215).

[222] A. S. Brown and C. J. Patel. "A standard database for drug repositioning". In: *Scientific data* 4.1 (2017). URL:unmtid-shinyapps.net/repodb/, pp. 1–7 (cit. on pp. 217, 220).

[223] *CAS REGISTRY [ONLINE]*. FAQ:cas.org. American Chemical Society (cit. on p. 215).

[224] H. Chen, F. Cheng, and J. Li. "iDrug: Integration of drug repositioning and drug-target prediction via cross-network embedding". In: *PLoS computational biology* 16.7 (2020). GITHUB:Case-esaC/iDrug, e1008040. DOI: 10.1371/journal.pcbi.1008040 (cit. on pp. 217, 220).

[225] X. Chen, N.-N. Guan, Y.-Z. Sun, J.-Q. Li, and J. Qu. "MicroRNA-small molecule association identification: from experimental results to computational models". In: *Briefings in bioinformatics* 21.1 (2020), pp. 47–61. DOI: 10.1093/bib/bby098 (cit. on p. 233).

[226] X. Chen, B. Ren, M. Chen, Q. Wang, L. Zhang, and G. Yan. "NLLSS: predicting synergistic drug combinations based on semi-supervised learning".

In: *PLoS computational biology* 12.7 (2016), e1004975. DOI: 10.1371/journal.pcbi.1004975 (cit. on p. 214).

[227] X. Chen, C. C. Yan, X. Zhang, X. Zhang, F. Dai, J. Yin, and Y. Zhang. "Drug–target interaction prediction: databases, web servers and computational models". In: *Briefings in bioinformatics* 17.4 (2016), pp. 696–712. DOI: 10.1093/bib/bbv066 (cit. on pp. 214, 215).

[228] X. Chen, C. Zhou, C.-C. Wang, and Y. Zhao. "Predicting potential small molecule–miRNA associations based on bounded nuclear norm regularization". In: *Briefings in Bioinformatics* 22.6 (2021), bbab328. DOI: 10.1093/bib/bbab328 (cit. on p. 234).

[229] *ClinicalTrials [Internet]*. URL: clinicaltrials.gov/. Bethesda (MD): National Library of Medicine (US) (cit. on pp. 220, 229).

[230] *Corona Project*. URL: cdcn.org/corona/. Oct. 2021 (cit. on p. 214).

[231] A. F. DAĞLI, A. KARATAŞ, C. ORHAN, M. TUZCU, M. ÖZGEN, K. ŞAHİN, and S. S. KOCA. "Antiinflammatory and antioxidant effects of gemcitabine in collagen-induced arthritis model". eng. In: *Turk J Med Sci* 47.3 (June 2017), pp. 1037–1044. ISSN: 1300-0144. DOI: 10.3906/sag-1606-80 (cit. on pp. 230, 231).

[232] A. P. Davis, C. J. Grondin, R. J. Johnson, D. Sciaky, J. Wiegers, T. C. Wiegers, and C. J. Mattingly. "Comparative toxicogenomics database (CTD): update 2021". In: *Nucleic acids research* 49.D1 (2021), pp. D1138–D1143. DOI: 10.1093/nar/gkaa891 (cit. on p. 220).

[233] *DrugCentral [Internet]*. Division of Translational Informatics at University of New Mexico. URL: https://drugcentral.org (cit. on p. 220).

[234] L. Ehrlinger and W. Wöss. "Towards a Definition of Knowledge Graphs". In: *SEMANTICS 2016: Posters and Demos Track* (Sept. 2016). RG:323316736 (cit. on p. 215).

[235] Y. Feng, J. Shen, E. D. Streaker, M. Lockwood, Z. Zhu, P. S. Low, and D. S. Dimitrov. "A folate receptor beta-specific human monoclonal antibody recognizes activated macrophage of rheumatoid patients and mediates antibody-dependent cell-mediated cytotoxicity". In: *Arthritis Research & Therapy* 13.2 (Apr. 2011), R59. ISSN: 1478-6354. DOI: 10.1186/ar3312. (Visited on 02/25/2022) (cit. on p. 231).

[236] L. Fraenkel et al. "2021 American College of Rheumatology Guideline for the Treatment of Rheumatoid Arthritis". eng. In: *Arthritis Care Res (Hoboken)* 73.7 (July 2021), pp. 924–939. ISSN: 2151-4658. DOI: 10.1007/s13167-019-00195-w (cit. on pp. 229–231).

[237] A. Gottlieb, G. Y. Stein, E. Ruppin, and R. Sharan. "PREDICT: a method for inferring novel drug indications with application to personalised medicine". In: *Molecular Systems Biology* 7.1 (2011), p. 496. DOI: 10.1038/msb.2011.26 (cit. on p. 220).

[238] C. Gu, B. Liao, X. Li, L. Cai, H. Chen, K. Li, and J. Yang. "Network-based collaborative filtering recommendation model for inferring novel disease-related miRNAs". In: *RSC Adv.* 7 (71 2017), pp. 44961–44971. DOI: 10.1039/C7RA09229F (cit. on pp. 216, 234).

[239] A. D. Hingorani et al. “Improving the odds of drug development success through human genomics: modelling study”. In: *Sci Rep* 9.1 (Dec. 2019), p. 18911. DOI: 10.1038/s41598-019-54849-w (cit. on p. 214).

[240] W. H. Organization. *ICD-11 Reference Guide: International Classification of Diseases for Mortality and Morbidity Statistics*. GUIDE:icd.who.int (cit. on p. 215).

[241] V. N. Ioannidis, X. Song, S. Manchanda, M. Li, X. Pan, D. Zheng, X. Ning, X. Zeng, and G. Karypis. *DRKG - Drug Repurposing Knowledge Graph for Covid-19*. GITHUB:gnn4dr/DRKG. 2020 (cit. on p. 233).

[242] M. J. Keiser et al. “Predicting new molecular targets for known drugs.” In: *Nature* 462.7270 (Nov. 2009), pp. 175–181. ISSN: 1476-4687 (Electronic); 0028-0836 (Print); 0028-0836 (Linking). DOI: 10.1038/nature08506 (cit. on p. 214).

[243] M. Kuhn, I. Letunic, L. J. Jensen, and P. Bork. “The SIDER database of drugs and side effects”. In: *Nucleic Acids Res* 44.D1 (Jan. 2016), pp. D1075–1079. DOI: 10.1093/nar/gkv1075 (cit. on p. 233).

[244] J. Li and Z. Lu. “A new method for computational drug repositioning using drug pairwise similarity”. In: *2012 IEEE international conference on bioinformatics and biomedicine*. IEEE. 2012, pp. 1–4. DOI: 10.1109/BIBM.2012.6392722 (cit. on pp. 217, 220).

[245] C. E. Lipscomb. “Medical subject headings (MeSH)”. In: *Bulletin of the Medical Library Association* 88.3 (2000). PMCID:35238, PMID:10928714, p. 265 (cit. on p. 214).

[246] H. Liu, W. Zhang, B. Zou, J. Wang, Y. Deng, and L. Deng. “DrugCombDB: a comprehensive database of drug combinations toward the discovery of combinatorial therapy”. In: *Nucleic Acids Res* 48.D1 (Jan. 2020), pp. D871–D881. DOI: 10.1093/nar/gkz1007 (cit. on p. 233).

[247] B. Lopez-Millan et al. “Therapeutic effect of the immunomodulatory drug lenalidomide, but not pomalidomide, in experimental models of rheumatoid arthritis and inflammatory bowel disease”. eng. In: *Exp Mol Med* 49.2 (Feb. 2017). PMCID:PMC5336556, PMID:28154372, e290. ISSN: 2092-6413. DOI: 10.1038/emm.2016.143 (cit. on pp. 230, 231).

[248] H. Luo, J. Wang, M. Li, J. Luo, X. Peng, F.-X. Wu, and Y. Pan. “Drug repositioning based on comprehensive similarity measures and Bi-Random walk algorithm”. In: *Bioinformatics* 32.17 (May 2016). code available on GITHUB : bioinfomaticsCSU/MBiRW, pp. 2664–2671. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw228 (cit. on p. 214).

[249] M. G. Maturo, M. Soligo, G. Gibson, L. Manni, and C. Nardini. “The greater inflammatory pathway-high clinical potential by innovative predictive, preventive, and personalized medical approach”. eng. In: *EPMA J* 11.1 (Mar. 2020). PMCID:PMC7028895, PMID:32140182, pp. 1–16. ISSN: 1878-5077. DOI: 10.1007/s13167-019-00195-w (cit. on p. 229).

[250] Y. Nam, M. Kim, H.-S. Chang, and H. Shin. “Drug repurposing with network reinforcement”. In: *BMC Bioinformatics* 20.13 (July 2019), p. 383. ISSN: 1471-2105. DOI: 10.1186/s12859-019-2858-6 (cit. on p. 214).

[251] C. Nardini, V. Osmani, P. G. Cormio, A. Frosini, M. Turrini, C. Lionis, T. Neumuth, W. Ballensiefen, E. Borgonovi, and G. D'Errico. "The evolution of personalized healthcare and the pivotal role of European regions in its implementation". In: *Future Medicine* (Apr. 2021). DOI: 10.2217/pme-2020-0115 (cit. on p. 233).

[252] Y. Okada, D. Wu, G. Trynka, T. Raj, C. Terao, K. Ikari, Y. Kochi, K. Ohmura, A. Suzuki, S. Yoshida, et al. "Genetics of rheumatoid arthritis contributes to biology and drug discovery". In: *Nature* 506.7488 (2014), pp. 376–381. DOI: 10.1038/nature12873 (cit. on p. 229).

[253] M. G. Ozsoy, T. Özyer, F. Polat, and R. Alhajj. "Realizing drug repositioning by adapting a recommendation system to handle the process". In: *BMC Bioinformatics* 19.1 (Apr. 2018), p. 136. ISSN: 1471-2105. DOI: 10.1186/s12859-018-2142-1 (cit. on p. 214).

[254] J. Piñero, J. M. Ramírez-Anguita, J. Saüch-Pitarch, F. Ronzano, E. Centeno, F. Sanz, and L. I. Furlong. "The DisGeNET knowledge platform for disease genomics: 2019 update". In: *Nucleic Acids Research* 48.D1 (Nov. 2019), pp. D845–D855. ISSN: 0305-1048. DOI: 10.1093/nar/gkz1021 (cit. on pp. 215–218).

[255] *PubMed [Internet]*. URL: ncbi.nlm.nih.gov/pubmed/. Bethesda (MD): National Library of Medicine (US) (cit. on p. 229).

[256] S. Pushpakom et al. "Drug repurposing: progress, challenges and recommendations". In: *Nature Reviews Drug Discovery* 18.1 (Jan. 2019), pp. 41–58. ISSN: 1474-1784. DOI: 10.1038/nrd.2018.168 (cit. on p. 233).

[257] N. Rappaport, M. Twik, I. Plaschkes, R. Nudel, T. Iny Stein, J. Levitt, M. Gershoni, C. P. Morrey, M. Safran, and D. Lancet. "MalaCards: an amalgamated human disease compendium with diverse clinical and genetic annotation and structured search". In: *Nucleic Acids Research* 45.D1 (Nov. 2016), pp. D877–D887. ISSN: 0305-1048. DOI: doi.org/10.1093/nar/gkw1012 (cit. on pp. 215, 217, 218).

[258] J. M. Raser and E. K. O'Shea. "Noise in gene expression: origins, consequences, and control". In: *Science* 309.5743 (Sept. 2005), pp. 2010–2013. DOI: 10.1126/science.1105891 (cit. on p. 233).

[259] G. B. Rogers. "Germs and joints: the contribution of the human microbiome to rheumatoid arthritis". In: *Nature medicine* 21.8 (2015), pp. 839–841. DOI: 10.1038/nm.3916 (cit. on p. 229).

[260] R. Roscher, B. Bohn, M. F. Duarte, and J. Garcke. "Explainable machine learning for scientific insights and discoveries". In: *Ieee Access* 8 (2020), pp. 42200–42216. DOI: 10.1109/ACCESS.2020.2976199 (cit. on p. 216).

[261] A. Saha and G. Alleyne. "Recognizing noncommunicable diseases as a global health security threat". In: *Bulletin of the World Health Organization* 96.11 (2018), p. 792. DOI: 10.2471/BLT.17.205732 (cit. on p. 229).

[262] Z. Tanoli, U. Seemab, A. Scherer, K. Wennerberg, J. Tang, and M. Vähä-Koskela. "Exploration of databases and methods supporting drug repurposing: a comprehensive survey". In: *Briefings in Bioinformatics* 22.2 (Feb.

2020), pp. 1656–1678. ISSN: 1477-4054. DOI: 10.1093/bib/bbaa003 (cit. on p. 215).

[263] P. Tieri and C. Nardini. "Signalling pathway database usability: lessons learned". In: *Mol Biosyst* 9.10 (Oct. 2013), pp. 2401–2407. DOI: 10.1039/C3MB70242A (cit. on p. 215).

[264] E. Tjoa and C. Guan. "A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.11 (2021), pp. 4793–4813. DOI: 10.1109/TNNLS.2020.3027314 (cit. on p. 233).

[265] P. Venkatesan. "Repurposing drugs for treatment of COVID-19". In: *Lancet Respir Med* 9.7 (July 2021), e63. DOI: 10.1016/S2213-2600(21)00270-8 (cit. on p. 214).

[266] D. S. Wishart et al. "DrugBank 5.0: a major update to the DrugBank database for 2018." In: *Nucleic Acids Res.* (Nov. 2017). DOI: doi.org/10.1093/nar/gkx1037 (cit. on pp. 214–218).

[267] Y. Xu and Q. Wu. "Prevalence Trend and Disparities in Rheumatoid Arthritis among US Adults, 2005–2018". In: *Journal of clinical medicine* 10.15 (2021), p. 3289. DOI: 10.3390/jcm10153289 (cit. on p. 229).

[268] C. Zhang and G. Yan. "Synergistic drug combinations prediction by integrating pharmacological data". In: *Synthetic and systems biotechnology* 4.1 (2019), pp. 67–72. DOI: 10.1016/j.synbio.2018.10.002 (cit. on p. 214).

[269] P. Zhang, P. Agarwal, and Z. Obradovic. "Computational drug repositioning by ranking and integrating multiple data sources". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2013, pp. 579–594. DOI: 10.1007/978-3-642-40994-3_37 (cit. on pp. 217, 220).

Conclusions

In the present work, we have undertaken the task of collecting, extending, and harmonising the contributions in the field of graph theory, machine learning, and data analysis the candidate produced during his Ph.D. in mathematics at the *"Università degli Studi Roma Tre"*.

The majority of the work presented here is derived from published articles in reputable conferences and journals, such as the Elsevier Journal of Computational Sciences (IF 3.817, cite score 5.9, Q1 in modelling and simulation) and IEEE/ACM Transactions on Computational Biology and Bioinformatics (IF 3.702, cite score 7.5, Q2 in applied mathematics).

Throughout this work, we have introduced several innovative methodologies and algorithms that are applicable across a broad spectrum of fields, namely, big data analysis, vehicular and pedestrian dynamics, as well as computational medicine and immunology. Despite the heterogeneity of these fields (which might appear quite far from one another), we were capable to capture the subtle mathematical underlying backbone. Consequently, we have been able to contribute, albeit with some variation, to all of these fields by employing similar techniques. These techniques are the result of the journey undertaken by the candidate that started at *"Università degli Studi Roma Tre"* nine years ago and led him to collaborate with (and thus learn from) many experts in heterogeneous fields. Notable are the collaborations with researchers affiliated with the *"Istituto per le Applicazioni del Calcolo"* which have been instrumental in expanding the candidate's knowledge and understanding.

These collaborations have demonstrated that, even in seemingly unrelated fields, a common mathematical foundation can often be found. As a result, the theories studied have multiple wide-ranging significant applications.

In particular, we have shown in Chapter 3 a fast and reliable algorithm for reducing in terms of size graphs equipped with an attribute-based colouring. This methodology hence enables the extraction of smaller yet meaningful representations from large datasets organised as graphs, as it is a common case in big data. An application example is discussed in the context of social networks.

A similar approach was employed, albeit on a different scale, in the context of cultural heritage, specifically in extracting information from graph representations of

museum-like environments. In this sense, Chapter 4 not only introduces a novel way to represent museums as coloured graphs (with colours modelling expert knowledge), but also investigates the complex behaviour of pedestrian traffic within exhibition spaces. Several novel approaches, some of which are in the field of machine learning, are introduced in this chapter, encompassing data gathering and analysis as well as visitor flow modelling and optimisation. Notably, the introduction of a digital twin allowed for the identification of changes with proven impacts on the daily operations of renowned museums such as the museum of Galleria Borghese in Rome and the Peggy Guggenheim Collection in Venice.

Chapter 5 shifts the focus from pedestrian to vehicular traffic, following the trend of flow management through mathematical modelling. Within this context, we address some of the problems related to real-time simulation and we propose two machine learning methods to assist the mathematical model in making "appropriate decisions". Firstly, we tackle the issue of incompatible flux data obtained from fixed sensors along highways and present a novel method for inverting the fundamental diagram, thereby enabling the recovery of density at these sensors. In detail, the method is built upon an LSTM-based artificial neural network capable of time-series analysis, effectively distinguishing between freeway and congested situations. Secondly, we address the problem of determining future-time boundary conditions for the mathematical traffic model. Once again, the mindful use of an artificial neural network proves to be effective in this regard.

Finally, in Chapter 6, we shifted our focus back to graph theory and we introduced a novel drug repurposing recommender system based on biased random walks. In particular, the system is based on a similarity-score entropy-like measure to construct a transition matrix over a drug-disease knowledge graph. The resulting Markov chain exhibits useful mathematical properties which enforce stability, consistency, and explainability of the recommendation process.

Overall, the present work has contributed to the fields of graph theory, machine learning, and data analysis by presenting novel methodologies, algorithms, and applications. The exploration of different domains showed the flexibility of mathematics as a unifying field, allowing for meaningful contributions across heterogeneous applications. Through collaborations and an interdisciplinary approach, we have showcased the versatility and impact of the techniques developed, paving the way for further advancements in these fields.

Author's contributions

Apart from the works introduced in this thesis, the candidate also published four other articles during his Ph.D.: three in the field of cryptography ([290, 272, 281]) with the Cryptography and Cybersecurity laboratory of *"Università degli Studi Roma Tre"* as a natural follow-up of the candidate's bachelor and master thesis work and one in the field of computational topology ([275]) with the Computational Visual Design Laboratory (CVD-Lab) of *"Università degli Studi Roma Tre"*.

At the time of writing, two more works by the candidate have been accepted for publication ([277, 273]) and one is currently under preparation ([288]), all in the field of cryptography.

Concurrently to the work presented in Chapter 6, the candidate is also collaborating with the European Project *ERA4TB* (European Regimen Accelerator for Tuberculosis). Here, again with the *"Digital Biology Unit"* of the IAC-CNR, the candidate is developing an agent-based system for simulating *in silico* Mycobacterium Tuberculosis (M.Tb.) pulmonary infection in a 2D/3D environment capable of reproducing both *in vivo* and *in vitro* experimental data. The model itself will be the object of a future paper currently under development [286].

The main objective of the research is to study the effect of novel drugs on M.Tb. infection. In this setting, validation and application of the model are currently undertaken in collaboration with the research laboratory of both the University of Padova and Pavia which provide results of *in vitro* simulations.

Further work was also carried out for what concern the characterisation of the Mt.B. disease in-host evolution and it is the object of an accepted paper [279].

The decision not to include those works relies mainly on their topics, which did not suit well with the cohesive view presented here. It is indeed the candidate's belief that these works comprise different distinct narratives in their own right and, consequently, they would have been superfluous within the scope of the present work.

Here is a comprehensive enumeration of the works accomplished by the candidate, organised according to the type of publication.

Journal papers

[270] F. Castiglione, C. Nardini, E. Onofri, M. Pedicini, and P. Tieri. "Explainable drug repurposing approach from biased random walks". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (July 2022). PMID:35839194, pp. 1009–1019. DOI: 10.1109/TCBB.2022.3191392 (cit. on pp. xiii, xiv, 213).

[271] P. Centorrino, A. Corbetta, E. Cristiani, and E. Onofri. "Managing crowded museums: visitors flow measurement, analysis, modeling, and optimization". In: *Journal of Computational Science* 53 (Apr. 2021), pp. 1–17. ISSN: 1877-7503. DOI: 10.1016/j.jocs.2021.101357 (cit. on pp. xiv, 103).

[272] M. Cianfriglia, E. Onofri, S. Onofri, and M. Pedicini. "Fourteen years of cube attacks". In: *Applicable Algebra in Engineering, Communication and Computing* (May 2023), pp. 1–43. DOI: 10.1007/s00200-023-00602-w (cit. on p. C-1).

[273] M. Cianfriglia, E. Onofri, and M. Pedicini. "mR$_{LWE}$-CP-ABE: a revocable CP-ABE for post-quantum cryptography". Accepted in Journal of Mathematical Cryptology (De Gruyter), ePRINT:2023/922. Dec. 2023 (cit. on p. C-1).

[274] F. Lombardi and E. Onofri. "Some results on colored network contraction". In: *Journal of Ubiquitous Systems and Pervasive Networks* 17.2 (Dec. 2022), pp. 91–98. DOI: 10.5383/JUSPN.17.02.006 (cit. on pp. xiv, 65).

[275] A. Paoluzzi, V. Shapiro, A. DiCarlo, G. Scorzelli, and E. Onofri. "Finite algebras for solid modeling using Julia's sparse arrays". In: *Computer-Aided Design* 155 (Nov. 2023), p. 103436. ISSN: 0010-4485. DOI: 10.1016/j.cad.2022.103436 (cit. on p. C-1).

Conference papers

[276] P. Centorrino, A. Corbetta, E. Cristiani, and E. Onofri. "Measurement and analysis of visitors' trajectories in crowded museums". In: *IMEKO TC-4*. IMEKO:2019-83. Florence, Italy: International Conference on Metrology for Archaeology and Cultural Heritage, Dec. 2019, pp. 423–428 (cit. on pp. xiv, 103).

[277] M. Cianfriglia, E. Onofri, and M. Pedicini. "mR$_{LWE}$-CP-ABE: a revocable CP-ABE for post-quantum cryptography". Long abstract accepted at CIFRIS23 conference. Dec. 2023 (cit. on p. C-1).

[278] F. Lombardi and E. Onofri. "Graph contraction on attribute-based coloring". In: *Procedia Computer Science* 201 (Apr. 2022). The 13th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 5th International Conference on Emerging Data and Industry 4.0 (EDI40), pp. 429–436. ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.03.056 (cit. on pp. xiv, 65).

[279] E. Mastrostefano, A. Ravoni, E. Onofri, P. Tieri, and F. Castiglione. "Harnessing computational models to uncover the role of the immune system in tuberculosis treatment". Accepted at the 7th International Workshop on Computational Methods for the Immune System Function (CMISF23–BIBM). Dec. 2023 (cit. on p. C-1).

[280] E. Onofri and A. Corbetta. "RSSi-based visitor tracking in museums via cascaded AI classifiers and coloured graph representations". In: *Collective Dynamics* 6 (Jan. 2022), pp. 1–17. DOI: 10.17815/CD.2021.131 (cit. on pp. xiv, 103).

[281] E. Onofri and M. Pedicini. "Novel notation on cube attack". In: *Collectio Ciphrarum, De Cifris Cryptanalysis* selected papers from the ITASEC2020 workshop (Mar. 2022). Ed. by Aracne, pp. 25–30. DOI: 10.53136/97912599486565 (cit. on p. C-1).

Submitted papers

[282] M. Briani, E. Cristiani, and E. Onofri. "Inverting the fundamental diagram and forecasting boundary conditions: how machine learning can improve macroscopic models for traffic flow". ARXIV:2303.12740. June 2023 (cit. on pp. xiii, xiv, 173).

Papers in preparation

[283] M. Caprolu, F. Lombardi, and E. Onofri. "Speculative Polkadot? an overview". In preparation (cit. on pp. xiii, 95).

[284] M. Catrambone, P. Centorrino, E. Cristiani, E. Onofri, and C. Riminesi. "On the pollution impact of visitors inside crowded museums". In preparation (cit. on pp. xiv, 166).

[285] F. Lombardi and E. Onofri. "Parallel graph contraction on attribute-based colouring". In preparation (cit. on p. xiii).

[286] E. Mastrostefano and E. Onofri. "PhysiTB: a custom module for M.Tb. infection simulation in PhysiCell". In preparation (cit. on p. C-1).

[287] E. Onofri. "On the theoretical aspects of colour contraction: γ-contraction". In preparation (cit. on pp. xiii, 65).

[288] E. Onofri and M. Pedicini. "Novel direction in cube attacks". In preparation (cit. on p. C-1).

Other contributions

[289] P. Centorrino, E. Cristiani, P. Ferrara, D. Macchion, and E. Onofri. *Measurement and analysis of the visitors behavior in the Peggy Guggenheim Collection*. Technical report. IAC–CNR, Feb. 2023, pp. 1–12 (cit. on pp. xiv, 103).

[290] M. Cianfriglia, E. Onofri, S. Onofri, and M. Pedicini. "Ten years of cube attacks". In: *IACR eprints* (Feb. 2022). ePrint:2022/137, pp. 1–30 (cit. on p. C-1).

[291] G. Del Monte, E. Onofri, G. Scorzelli, and A. Paoluzzi. "Local congruence of chain complexes". arXiv:2004.00046. May 2020.

Alphabetical Index

G

H

L

M