



ESSAYS ON NETWORK DATA ANALYSIS THROUGH THE BAG-OF-PATHS FRAMEWORK

Sylvain COURTAÏN

*Thesis submitted in partial fulfillment of the requirements for the degree
of Doctor of Philosophy in Economics and Management Sciences*

November 2022

UCLouvain - LSM - LouRIM
Louvain Research Institute in Management and Organizations
Louvain School of Management
Université catholique de Louvain

Examination Committee:

Prof. François BAVAUD
Dr. Olivier CAELEN
Prof. François FOUSS (President)
Dr. Bertrand LEBICHOT
Prof. Marco SAERENS (Advisor)

Université de Lausanne, Switzerland
R&D ML and AI Researcher, Worldline Inc.
UCLouvain/LouRIM, Belgium
Université du Luxembourg, Luxembourg
UCLouvain/ICTEAM, Belgium

*Information is the oil of the 21st century,
and analytics is the combustion engine.*
PETER SONDERGAARD

Abstract

Since the rapid growth of the Internet and the advent of social networks in the 2000s, the amount of available network data is quickly increasing, leading to the development of new network analysis methods. Nowadays, these network analysis methods have spread to various fields, including, among others, marketing, supply chain, finance, and biology, as essential analysis and prediction tools. This thesis focuses on the development of one of these methods, called the bag-of-paths framework. This framework has the specificity to define a family of dissimilarity measures between nodes of the network that extrapolate between an optimal exploitation of the graph structure (optimal behavior - shortest path distance) and a random exploration of the graph (random behavior - commute time distance) via a parameter that controls the desired degree of randomness/exploration. Throughout this thesis, we propose several theoretical and practical extensions of the bag-of-paths framework. Regarding the theoretical contributions, we incorporate capacity constraints on edges, marginal constraints on input and output flows, and a Poisson distribution weighting and constraining path lengths, into the bag-of-paths framework. Furthermore, we expose the applicability of this framework through graph-based semi-supervised classification tasks and a real-life fraud detection case.

Acknowledgments

The first person I would like to thank is my advisor Prof. M. Saerens, without whom this thesis would never have existed. His advice, his patience, his availability, his listening and above all his kindness have been extremely precious assets during the realization and the finalization of this thesis.

I would also like to thank Prof. F. Fouss and Dr. B. Lebichot for having accepted to be part of my supervisory committee and my jury, as well as for their follow up and advice during the different stages of this thesis. I also thank Prof. F. Bavaud and Dr. O. Caelen for having accepted to join the jury and for their comments which helped me to finalize this thesis. Finally, I would like to thank Prof. F. Fouss for accepting to be the president of the jury.

Then, I would like to thank my co-authors Prof. M. Saerens, Dr. G. Guex, Dr. I. Kivimäki, Dr. B. Lebichot and P. Leleux who helped me to achieve the various scientific articles which compose this thesis. I also thank Worldline SA/NV, R&D, for providing me with real data and their expertise in fraud detection.

Another part of my thanks goes to my colleagues of the UCLouvain: Prof. M. Kolp, Prof. C. Vande Kerckhove, Prof. J. Vanderdonckt, Dr. N. Burny, Dr. H. Haidar, Dr. S. Heng, Dr. I. Khaddam, Dr. S. Kiv, Dr. M. Madany, Dr. T.-D. Nguyen, Dr. M. Zen, G. Aghakhani, S. Baudine, J. Charles, H. Dehaybe, B. Dupont, M. Lebrun, P. Leleux, M. Ousmer, H. Rämer, P. Rodberg, F. Vancompernelle Vromman, L. Vecchiato and many others with whom I had the opportunity to share many discussions, not always very serious, as well as many pleasant team-buildings.

Furthermore, I would like to thank my family and friends for their support and encouragement during these five years. More specifically, I would like to thank my parents who gave me the opportunity to do this doctorate by enabling me to attend university education and who have always supported me no matter what my choices were.

Finally, I would like to thank my beloved partner Alice for having been by my side during all this adventure in the joyful and especially in the difficult moments. I cannot thank her enough for having proofread this manuscript many times while carefully avoiding, according to her own words, the parts composed of hieroglyphs.

Contents

1	Introduction	1
1.1	List of publications	3
1.1.1	Core publications	3
1.1.2	Related publications	6
1.2	Structure of the thesis	7
2	Graphs and networks framework	9
2.1	Basic concepts	10
2.1.1	Typology of graphs	10
2.1.2	Terminology of graphs	11
2.2	Common associated matrices	12
2.2.1	Adjacency matrix	12
2.2.2	Cost matrix	13
2.2.3	Degree matrices	14
2.2.4	Reference transition probability matrix	14
2.2.5	Laplacian matrices	15
2.2.6	Modularity matrix	16
2.2.7	An illustrative example	17
2.3	Similarity and dissimilarity measures	19
2.3.1	Basic concepts	19
2.3.2	Shortest path distance	20
2.3.3	Commute time distance	20
2.4	Kernels and derived distances	22
2.4.1	Basic concepts	22
2.4.2	Distances from kernels and vice versa	23
3	Bag-of-paths framework	25
3.1	Bag-of-paths model	26
3.2	Bag-of-hitting-paths model	30
3.3	Randomized shortest path model	31
3.4	Derived dissimilarity measures	34
3.4.1	Surprisal distance	34
3.4.2	Free energy distance	35
3.4.3	Randomized shortest path dissimilarity	36

4	Randomized shortest path with capacity constraints	39
4.1	Basic notions	40
4.1.1	Background and notation	40
4.1.2	Quantities of interest	41
4.2	Dealing with edge flow capacity constraints	43
4.2.1	Lagrange function in case of capacity constraints	44
4.2.2	Exploiting Lagrangian duality	46
4.2.2.1	Computing the dual function	47
4.2.2.2	Maximizing the dual function	47
4.2.3	Resulting algorithm and discussion	48
4.3	Dealing with net flow capacity constraints	50
4.3.1	Lagrange function in case of net flow capacity constraints	52
4.3.2	Resulting algorithm and discussion	53
4.4	Experiments	54
4.4.1	First illustrative example	54
4.4.2	Second illustrative example	56
4.4.3	Scalability experiment	57
4.5	Conclusion	59
5	Relative entropy-regularized optimal transport on a graph	63
5.1	Related work	65
5.2	Notation, problem statement, and preliminaries	66
5.2.1	Notation	66
5.2.2	Problem statement	67
5.2.3	Definition of an extended, single-source and target, graph	67
5.2.3.1	Adjacency matrix of the extended graph	68
5.2.3.2	Transition matrix of the natural random walk on the extended graph	69
5.2.3.3	Computing the values of α	71
5.3	Optimal transport on a graph in a constrained RSP model	72
5.3.1	Lagrange function	73
5.3.2	Exploiting Lagrangian duality	75
5.3.2.1	Computing the dual function	76
5.3.2.2	Maximizing the dual function	76
5.3.3	Resulting algorithm	79
5.3.4	Computation time investigation	80
5.3.5	A distance measure between nodes	82
5.3.5.1	Coupling matrix	82
5.3.5.2	Margin-constrained bag-of-paths surprisal distance	83

5.4	Optimal transport on a graph from a constrained RSP model with capacity constraints	84
5.5	Case study: application to transportation problems	85
5.5.1	Theoretical background	85
5.5.2	Illustrative example	89
5.5.3	Empirical comparisons	92
5.6	Conclusion	96
6	Bag-of-paths with path lengths weighted by a Poisson distribution	99
6.1	Basic notions	100
6.2	Bag-of-paths weighted by path lengths	102
6.2.1	A Poisson distribution on path lengths	102
6.2.2	Computing the joint probability of drawing a path starting in i and ending in j	103
6.2.3	A derived distance measure between nodes	104
6.2.4	Weighting start and end nodes	105
6.3	Bag-of-paths with Poisson-distributed path lengths	106
6.3.1	BoP model with Poisson-distributed path lengths	106
6.3.2	Computing the joint probability of drawing a path starting in i and ending in j	108
6.3.3	A derived distance measure between nodes	109
6.3.4	Correlation analysis	110
6.4	Conclusion	110
7	Graph-based semi-supervised classification	113
7.1	Investigated methods	115
7.1.1	Bag-of-paths dissimilarities and similarities	115
7.1.2	Covariance and correlation kernels	117
7.1.3	Laplacian and adjacency based dissimilarities and similarities	118
7.1.4	Commute time-based kernels	119
7.1.5	Diffusion-based similarities	120
7.2	Datasets	124
7.3	Experimental methodology	124
7.4	Results and discussion	126
7.4.1	Weights and priors analysis	129
7.4.2	Per-category analysis	132
7.4.2.1	First category	133
7.4.2.2	Second category	137
7.4.2.3	Third category	141
7.4.2.4	Fourth category	145

7.4.2.5	Fifth category	149
7.4.3	Best-ranked methods analysis	153
7.4.4	Comparison of the impact of the different extracted feature sets	158
7.4.5	Investigation of the best performing method	161
7.5	Conclusion	162
8	Graph-based fraud detection with the free energy distance	165
8.1	Related work	168
8.2	Free energy distance	174
8.2.1	Background	174
8.2.2	Computing the directed free energy distance on large graphs	175
8.2.3	Application to the fraud detection problem	177
8.3	Experimental comparisons and discussion	178
8.3.1	Real-world fraud detection study case	178
8.3.2	Impact analysis of the number of iterations	182
8.3.3	Time complexity analysis	183
8.4	Conclusion	184
9	Conclusion	187
9.1	Content and contributions	187
9.2	Limitations	190
9.3	General discussion	192
A	Randomized shortest path model: a flow approach	195

List of Figures

2.1	Illustration of the seven bridges of Königsberg and Euler's solution [31].	9
2.2	A small undirected and unweighted graph G	17
4.1	Net expected number of visits to nodes $\bar{\mathbf{j}}$ (see Eq. 4.18) obtained with the capacity-constrained RSP Algorithm 1 on a 2D grid with several obstacles between two nodes. The source node s (upper left corner) and the target node t (lower right corner) are highlighted in green. All the edges around the obstacles are constrained to low capacities. Red represents a higher $\bar{\mathbf{j}}$ and blue a lower one. The trajectory of $\bar{\mathbf{j}}$ always follows the shortest path between the two nodes, taking the obstacles into account. The $\bar{\mathbf{j}}$ is more dispersed when θ is small and more concentrated when θ is large.	55
4.2	A small undirected graph composed of eight nodes [227]. The values on the edges represent capacities; moreover, all edge costs are set to 1, except the added shortcut edge (dashed line) whose cost is 10. The maximum possible flow between the source node s and the target node t is 12, and is equal to the min-cut between nodes $\{a, b, c\}$ and $\{d, e\}$	56
4.3	Representation of the flow from s to t depending on the value of θ , for the capacity-constrained RSP and the graph appearing in Fig. 4.2. The thickness of the arrows is scaled with respect to the largest net flow present in the graph. The flow between s and t increases with θ and almost reaches its maximum value of 12 when $\theta \geq 1$	58
4.4	Evolution of the flow between nodes $\{f, g, h\}$ and t satisfying the capacity constraints (total flow in the original graph G , without the shortcut edge), provided by Algorithm 1, in terms of the θ parameter, with a gradient step $\alpha = 1/\theta$. The intersection between the curve and the y -axis when $\theta \rightarrow 0^+$ is 4.699. Note that the x -axis is scaled logarithmically. The flow increases with θ and reaches its maximum value of 12 when $\theta > 1$	59

4.5	Evolution of the mean computation times of all methods (left) and of the three operations research algorithms (right) on directed graphs of increasing size n . Note that the scale of the two graphs is not the same. The linear solver CVX is slower than all other methods. All three operations research algorithms are faster than the cRSP, especially the PR.	60
5.1	On the left, a small undirected graph G with two input nodes $In = \{2, 3\}$ (in green) and two output nodes $Out = \{7, 8\}$ (in red). On the right, the extended graph G_{ext} of this small undirected graph G with one source supernode 1 (in dark green) connected to all the input nodes and one target supernode $n = 9$ (in dark red) connected to all the output nodes. Therefore, on this extended graph, the source supernode is node 1 and the target supernode is node n (the total number of nodes in G_{ext}).	68
5.2	Evolution of the mean computation times of the RSPOT, the cBoP, the cBoPH, the Coin-or CLP, and the CVX GLPK on lattices of increasing size n , containing $\lfloor n/3 \rfloor$ sources and $\lfloor n/3 \rfloor$ targets with uniform weights. The linear solvers CLP and CVX are slower than all other methods.	82
5.3	Network representation of transportation problem.	88
5.4	Representation of the solutions of the transportation problem presented in Table 5.2 provided by the Algorithm 2 for several θ values. The optimal total transportation cost of 4525 is reached for all $\theta > 2$	90
5.5	Representation of the solutions of the transportation problem presented in Table 5.2 with capacity constraint of 150 on each road provided by the Algorithm 3 for several θ values. The optimal total transportation cost of 4625 is reached for all $\theta \geq 2$	91
5.6	Computation time required to solve the BTPs for each method. The linear solver CVX is the slowest for all BTPs, except for BTP5 where it is the MODI - NWCM.	95
5.7	Computation time required to solve the UTPs for each method. The linear solver CVX is the slowest for all UTPs.	95
6.1	Pearson correlation coefficients ρ between the PSurp and the PWSurp for several θ and λ values. The two distances are highly correlated regardless of the chosen parameters.	111

- 7.1 Representation of the experimental design used on our experiment. For each dataset and method, 10 runs of a 5×5 nested cross-validation methodology with different folds of labeled/unlabeled nodes on each run is performed. In each external fold, 20% of the labels are used to train the model and the remaining 80% are hidden to compute the accuracy score of the classifier. In each internal fold, the parameters are tuned on the external training fold by keeping 80% of the nodes labeled. The final accuracy is obtained by averaging the 50 accuracy score obtained on the external folds across the 10 runs. 128
- 7.2 Mean ranks and 95% Nemenyi confidence intervals for the 6 margin-constrained bag-of-paths methods and the 3 Poisson weighted surprisal distances with priors across the 14 datasets for feature sets 5%F, 10%F, 20%F and Ker. Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are always obtained with the external parameter L1-normalized degree. 131
- 7.3 Mean ranks and 95% Nemenyi confidence intervals for the methods of the first category across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the PWSurp for all the feature sets, except for 10%F where it is the cBoPH. 135
- 7.4 Mean ranks and 95% Nemenyi confidence intervals for the methods of the second category across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the nCor for all the feature sets, except for 10%F where it is the nCorH. 139

- 7.5 Mean ranks and 95% Nemenyi confidence intervals for the methods of the third category across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the LF for all the feature sets. 143
- 7.6 Mean ranks and 95% Nemenyi confidence intervals for the methods of the fourth category across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the SCCT for all the feature sets. 147
- 7.7 Mean ranks and 95% Nemenyi confidence intervals for the methods of the fifth category across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the LogCom for all the feature sets. 151
- 7.8 Mean ranks and 95% Nemenyi confidence intervals for the best-ranked methods across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the PWSurp for all the feature sets, except for 10%F where it is the SCCT. 156
- 7.9 Mean ranks and 95% Nemenyi confidence intervals for the eight best-ranked method of the first and third categories across the four feature sets (5%F, 10%F, 20%F, and Ker). Two feature sets are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the feature sets: the higher the rank, the best the results obtained on the feature set. Each time, the best feature set highlighted in blue is either the Ker or the 5%F. 159

7.10	Mean ranks and 95% Nemenyi confidence intervals for the eight best-ranked method of the second, fourth and fifth categories across the four feature sets (5%F, 10%F, 20%F, and Ker). Two feature sets are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the feature sets: the higher the rank, the best the results obtained on the feature set. Each time, the best feature set highlighted in blue is either the Ker or the 5%, except for the SCCT where it is 10%F.	160
8.1	A scheme, strongly inspired from [72], illustrating the layers of control in a FDS.	166
8.2	A small toy example of a tripartite graph. Left sub-figure shows a graph composed of five transactions (in red), including two fraudulent (risk score = 1), between three cardholders (in green) and three merchants (in orange). Central sub-figure represents the risk score obtained after running APATE with a $\alpha = 0.5$. A cardholder (risk score = 0.32) is identified as more fraudulent than the others. Right sub-figure represents the risk score obtained after running FraudsFree with a $\theta = 1$. A cardholder (risk score = 0.28) and two merchants (risk score = {0.22, 0.31}) are identified as more fraudulent than the others. Those risk scores are then used as features in a random forest for both methods.	170
8.3	Real-world FDS scenario composed of three set of data. A training set of 15 days where the fraud tags are known. A gap set of 7 days where the fraud tags are unknown. A test set of 1 day where the fraud tag is predicted. This scenario is repeated each day until the last day.	173
8.4	The directed lattice derived from the original graph. It only considers walks of length up to ℓ	177
8.5	Mean ranks and 95% Nemenyi confidence intervals for the 11 methods (see Table 8.3) based on the Card Pr@100. Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best method highlighted in blue is the RCTK SSL+FB. . .	181
8.6	Variables selected by the random forest classifier for the RCTK SSL+FB model for all days. The four merchant risk scores are the most selected features. See Table 8.1 for acronyms.	181

- 8.7 Variables selected by the random forest classifier for the FF SSL+FB model for all days. The four merchant risk scores are the most selected features. See Table 8.1 for acronyms. 182
- 8.8 Evolution of the mean Card Pr@100 score provided by the RCTK SSL+FB (left) and the FF SSL+FB (right) with the number of iterations for each alpha and theta parameter values between Day 30 and Day 40. The maximum global mean Card Pr@100 is reached for both methods at five iterations. 183
- 8.9 Evolution of the time required to create the tripartite graph and extract the 12 features depending on the number of transactions n_{Tx} for the FF ($\theta = 0.5$) and the RCTK ($\alpha = 0.7$). The time complexity of both methods is of the order of $\mathcal{O}(n_{\text{Tx}})$. . . 184

List of Tables

2.1	Summary of the various types of graphs.	11
5.1	Table representation of transportation problem.	87
5.2	Table representation of a small example of transportation problem.	89
5.3	Five balanced transportation problems used in our comparisons.	93
5.4	Five unbalanced transportation problems used in our comparisons.	94
5.5	Total transportation cost for the various methods obtained on the different transportation problems presented in Tables 5.3 and 5.4. The optimal total transportation costs are highlighted in boldface for each transportation problem. The RSPOT, the MODI, and the linear solver CVX are reaching the optimal total transportation costs for all TPs.	94
7.1	The methods investigated in our experiment with their acronym and their measure type.	122
7.2	Parameter ranges for the investigated methods by category. The symbol ‘/’ means that the method has no parameter. . . .	123
7.3	Main characteristics of the datasets used in our experiment. .	125
7.4	The best combination of kernel transformation and feature extraction options for each method according to the Nemenyi tests rankings. The symbol ‘/’ means that no transformation is applied.	127
7.5	Classification accuracy in percent $\pm$ standard deviation for the 6 margin-constrained bag-of-paths methods and the 3 Poisson weighted surprisal distances with priors, obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing weight parameter or prior vector is highlighted in boldface for each method, each dataset and each feature set. The L1-normalized degree weights (the cBoPDegree and the cBoPHDegree) and priors (the PWSurpDegree) obtain most of the time the best results across all the feature sets.	130

- 7.6 Classification accuracy in percent $\pm$ standard deviation for the methods of the first category obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. Most of the best results across all the feature sets are obtained by the PWSurp in Ker. 134
- 7.7 The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the first category, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface. 136
- 7.8 The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the first category, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface. . . . 136
- 7.9 Classification accuracy in percent $\pm$ standard deviation for the methods of the second category obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. The best results across all the feature sets are obtained either in Ker or in 5%F, but the methods providing these results differ from one dataset to another. 138
- 7.10 The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the second category, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface. 140
- 7.11 The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the second category, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface. . . . 140

- 7.12 Classification accuracy in percent $\pm$ standard deviation for the methods of the third category obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. The LF stands out from the other methods by providing most of the highlighted results. 142
- 7.13 The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the third category, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface. 144
- 7.14 The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the third category, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface. . . . 144
- 7.15 Classification accuracy in percent $\pm$ standard deviation for the methods of the fourth category obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. Most of the best results across all the feature sets are obtained by the SCCT. . . . 146
- 7.16 The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the fourth category, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface. 148
- 7.17 The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the fourth category, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface. . . . 148

7.18	Classification accuracy in percent $\pm$ standard deviation for the methods of the fifth category obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. The LogCom stands out from the other methods by providing most of the highlighted results.	150
7.19	The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the fifth category, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.	152
7.20	The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the fifth category, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface. . . .	152
7.21	The best-ranked methods selected according to the Nemenyi tests rankings in each category and in each feature set. . . .	153
7.22	Classification accuracy in percent $\pm$ standard deviation for the two best-ranked methods of each category in each feature set (5%, 10%, 20%, and Ker) obtained on the different datasets. The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. The best results across all the feature sets are obtained either in Ker or in 5%F, but the methods providing these results differ from one dataset to another.	154
7.23	The p -values provided by pairwise Wilcoxon signed-rank tests on the best-ranked methods, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.	157
7.24	The p -values provided by pairwise Wilcoxon signed-rank tests on the best-ranked methods, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.	157
7.25	The p -values provided by pairwise Wilcoxon signed-rank tests for the 16 methods. Each table contains the p -values of two methods, for the first method in the upper right triangle and the second method in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface. . . .	161

7.26	Ranking of the 16 best-ranked methods and the three baseline methods according to Nemenyi test across all the data and the feature sets (the higher the mean rank, the better). The best-ranked method overall is the PWSurp in Ker.	163
8.1	Features used by the random forest classifier. Notice that each transaction is linked with a cardholder and with a merchant at a certain date, but this information is only used to build the tripartite graph [170].	172
8.2	Summary of all the acronyms presented in Section 8.1, 8.2 and 8.3.	179
8.3	Mean Card Pr@100 between Day 41 and Day 139 for each of the 12 methods (see Table 8.2 for acronyms). The average time in seconds required to create the tripartite graph and extract the 12 features for each day is also reported for each method except for the first one. The FF SSL+FB obtains the best Card Pr@100 and the RCTK SSL+FB 5 ITER is the fastest.	180
9.1	Summary of the contributions of the thesis.	190

Introduction

| 1

Nowadays, with the development of information and communications technologies, more than two quintillion (10^{30}) bytes of data are generated every day around the world [244]. This data, although abundant, is gaining more and more importance within companies to the point of being recognized as a resource equivalent to black gold by some of them [38].

Data analytics is a strategic tool for companies that allows them to optimize their production tools, to improve their after-sales services, to predict the needs and buying behaviors of their customers, to avoid churn, to detect fraudulent credit card transactions, and much more.

In order to carry out their analyses, companies have a wide range of models and algorithms at their disposal. These are the objects of study of applied statistics, machine learning and data mining [7, 195, 304].

More precisely, machine learning is a field of artificial intelligence that studies algorithms and models based on mathematical and statistical approaches giving computers the ability to learn from data or their environment. In other words, the computers improve their own performance in solving tasks without being explicitly programmed for each of these tasks. More broadly, machine learning encompasses the design, implementation, optimization, development, and analysis of such algorithms and models [7, 195]. For its part, data mining is sometimes defined as the application of machine learning methods to already available large datasets [7].

Since the rapid growth of the Internet and the advent of social networks in the 2000s, a specific type of data is increasingly available from the mass of data generated every day, **network data** also called **graph-based data**. Indeed, networks or graphs are more and more used in many areas including, among others, chemistry, biology, physics, human sciences, and engineering [100, 103]. This increase in the availability of network data has led to the development of new network analysis methods and, more broadly, **network science**, also studied in machine learning dealing with graphs.

One of the best-known network is the World Wide Web, within which web pages are interconnected by hyperlinks. This famous network is used by search engines as Google, Yahoo, Bing, etc. to provide billions of search results to their users every day. More precisely, the search engines results are web page rankings obtained by calculating indicators, scores, such as the Page Rank [214] on the World Wide Web. These indicators are prestige measures

that quantify the importance of web pages composing the graph [103].

Other examples of networks are social networks such as Facebook, LinkedIn, Instagram, etc. where people are connected with each other on the basis of friendships, work relations, common interests, or any other characteristic which they are likely to share. The structure of these networks is the study object of social network analysis (SNA) [263]. An example of SNA is predicting the next friend of a Facebook user using link prediction on the social network.

Still other examples of networks are transportation networks, which include among others road, sea and air networks. These networks are used to solve transportation problems [83, 141] such as moving a quantity of goods from factories to retailers based on supply and demand at minimum cost. This type of problem is usually called a constrained graph flow optimization problem.

One last example of graphs, more in line with the news of the last three years, is virus graphs. These graphs are used to model the spread of viruses like COVID-19 [28] via diffusion models and detect suspected cases at an early stage [286].

In this thesis, we focus on one of these **network analysis models** called the **bag-of-paths framework** (BoP) [105, 156, 238] which applies to graph-based data. This framework has the particularity of extrapolating between optimal and random behavior via a parameter that controls the desired degree of randomness.

More precisely, the BoP framework considers a probability distribution on the set of all possible paths between pairs of nodes of a graph, which depends on an inverse temperature parameter. This parameter allows to monitor the balance between *exploration* (random behavior) and *exploitation* (optimal behavior). Indeed, when this parameter is close to zero, the likelihood of selecting a path is reduced to the probabilities generated by a random walk on the graph (pure exploration). On the other hand, when this parameter is large, the probability distribution is biased, and only the least-cost paths, also called shortest paths, are considered (pure exploitation) [105].

Including randomness into network analysis models allows, among others, to design models able to predict the behavior of human beings, or more broadly of living beings, which is not an easy task. Indeed, unlike machines, living beings make decisions based on a large number of different factors, many of which are not measurable, and therefore do not have optimal reasoning and are sometimes even irrational [76]. Moreover, when making decisions, living beings generally do not have a complete view of the problem and thus only possess part of the information. It is therefore important that models take this into account by incorporating an irrationality part or, in other words, randomness.

Moreover, the BoP framework allows to define a set of graph measures which interpolates between these two traditional paradigms that are the shortest paths between nodes of a graph and the random walks on the graph. Within this set, we find among others a family of dissimilarity measures that captures the notion of relative accessibility between nodes, in other words, a combination of both proximity and amount of connectivity in the graph [105].

Recently, several works have already proposed theoretical and practical extensions of the BoP framework. Among them, we find the definition of new betweenness measures [155, 172], the addition of capacity constraints on nodes of the graph [175], the development of new node criticality measure [173], the modeling of animal behavior [217] and the application to collaborative recommendation problems [175].

This effort is pursued in this thesis by proposing three theoretical and two practical extensions of the BoP framework. More specifically, we propose to incorporate capacity constraints on edges (Chapter 4), marginal constraints on input and output flows (Chapter 5), and a Poisson distribution weighting and constraining path lengths (Chapter 6) into the BoP framework. Regarding the applications, we present a graph-based semi-supervised classification task (Chapter 7) and a real-world fraud detection case (Chapter 8).

The remainder of this chapter is devoted to present the publications and collaborations achieved during this thesis in Section 1.1 and the structure of this thesis in Section 1.2.

1.1 List of publications

During the completion of this thesis, I had the opportunity to publish several scientific articles as first author and also collaborate on others as co-author. The list of publications alongside with my contributions to each of them is presented in the two following subsections. Subsection 1.1.1 introduces publications directly related to some chapters of this thesis, while Subsection 1.1.2 lists publications that are not included in this thesis to which I collaborated.

1.1.1 Core publications

- (1) **S. Courtain, B. Lebichot, I. Kivimäki, and M. Saeuens.** “Graph-based fraud detection with the free energy distance”. In: *Proceedings of the 8th International Conference on Complex Networks and their Applications (CNA’19)*. Springer, 2019, pp. 40–52

This paper investigates a real-world application of the free energy distance between nodes of a graph defined in the BoP framework [105] by

proposing an improved extension of the existing fraud detection system named APATE [281]. It relies on a new way of computing the free energy distance based on paths of increasing length, and scaling on large, sparse, graphs. This new approach is assessed on a real-world large-scale e-commerce payment transactions dataset obtained from a major Belgian credit card issuer, Worldline SA/NV. The experimental results show that the free-energy based approach reduces the computation time by one half while maintaining state-of-the-art performance.

My contribution to this paper was to adapt, with the help of Dr. B. Lebichot, the theory developed by Dr. I. Kivimäki and Prof. M. Saerens to a fraud detection system. In addition, I was in charge of all the other aspects of this paper, except for the implementation that was partially provided by Dr. B. Lebichot.

- (2) **S. Courtain, P. Leleux, I. Kivimäki, G. Guex, and M. Saerens.** “Randomized shortest paths with net flows and capacity constraints”. In: *Information Sciences* (2020), pp. 341–360

This paper extends the randomized shortest path (RSP) model [156, 238, 300], which is a specific interpretation of the BoP model, by investigating the net flow RSP and adding capacity constraints on edge flows. In its first part, the present work first develops the net flow RSP model considering that edge flows in opposite directions neutralize each other (as in electric networks), and proposes an algorithm to compute the expected routing costs between all pairs of nodes. This quantity is called the net flow RSP dissimilarity measure between nodes. Experimental comparisons on node clustering tasks indicate that the net flow RSP dissimilarity is competitive with other state-of-the-art dissimilarities. In its second part, our work shows how to introduce capacity constraints on edge flows, and a procedure is developed to solve this constrained problem by exploiting Lagrangian duality.

I was responsible for the global supervision and the content of the capacity constraints on edge flows part of this paper. The theory behind this part was conjointly developed with Dr. I. Kivimäki and Prof. M. Saerens. Moreover, I also worked on the illustrative examples and participated in the entire writing and reviewing process.

- (3) **S. Courtain, G. Guex, I. Kivimäki, and M. Saerens.** “Relative entropy-regularized optimal transport on a graph: a new algorithm and an experimental comparison”. In: *Submitted for publication, currently under minor revision* (2021)

The present work investigates a new relative entropy-regularized algorithm to solve the optimal transport on a graph problem within the

RSP formalism [156, 238, 300] following [120, 123]. More precisely, a unit flow is injected into a set of input nodes and collected from a set of output nodes with specified marginals, while minimizing the expected transportation cost together with a paths relative entropy regularization term, providing a randomized routing policy. The main advantage of this new formulation compared to the one of [120, 123] is the fact that it can easily accommodate edge flow capacity constraints which commonly occur in real-world problems. The resulting optimal routing policy, i.e., the probability distribution of following an edge in each node, is Markovian and is computed after constraining the input and output flows to the prescribed marginal probabilities thanks to a variant of the algorithm developed in (2). In addition, experimental comparisons with other recently developed techniques show that the distance measure between nodes derived from the introduced model provides competitive results on semi-supervised classification tasks.

Also in this paper, part of the theory behind the model was developed in collaboration with Dr. I. Kivimäki, Dr. G. Guex and Prof. M. Saeuens. Furthermore, I was in charge of all the other aspects of this paper.

- (4) **S. Courtain and M. Saeuens. “A simple extension of the bag-of-paths model weighting path lengths by a Poisson distribution”. In: *Proceedings of the 10th International Conference on Complex Networks and their Applications (CNA’21)*. Springer, 2022, pp. 220–233**

This paper extends the BoP model [105] by introducing a weighting of the length of the paths in the network, provided by a Poisson probability distribution. The main advantage of this approach is that it allows to tune the mean path length parameter, which is most relevant for some applications. Various quantities of interest, such as the probability of drawing a path from the bag of paths, or the joint probability of sampling any path connecting two nodes of interest, can easily be computed in closed-form from this model. In this context, a new distance measure between nodes of a network, considering a weighting factor on the length of the paths, is defined. Experiments on semi-supervised classification tasks show that the introduced distance measure provides competitive results compared to other state-of-the-art methods. Moreover, a new interpretation of the logarithmic communicability similarity measure [134] is proposed in terms of this new model.

Apart from the theoretical part which was developed in collaboration with Prof. M. Saeuens, I was responsible for the content and all the other aspects of this paper.

1.1.2 Related publications

- (5) **P. Leleux, S. Courtain, G. Guex, and M. Sauerens. “Sparse randomized shortest paths routing with Tsallis divergence regularization”. In: *Data Mining and Knowledge Discovery* (2021), pp. 986–1031**

This paper rephrases the RSP formalism [156, 238, 300] in terms of Tsallis divergence regularization, instead of Kullback-Leibler divergence. The main consequence of this change is that the resulting routing policy (local transition probabilities) becomes sparse. In addition, two new dissimilarity measures between nodes are derived from this framework: the Tsallis RSP and free energy dissimilarities. Experimental comparisons on node clustering and semi-supervised classification tasks show that the two derived dissimilarity measures provide state-of-the-art results.

My responsibility in this paper was to perform the semi-supervised classification experiments and to write the section on these experiments. I was also involved in the implementation of the methods and the reviewing process.

- (6) **G. Guex, S. Courtain, and M. Sauerens. “Covariance and correlation kernels on a graph in the generalized bag-of-paths formalism”. In: *Journal of Complex Networks* 8.6 (2021), pp. 1–46**

This paper derives closed-form expressions computing the expectation of co-presence and of number of co-occurrences of nodes on paths sampled from a network according to general path weights (a bag of paths). The underlying idea is that two nodes are considered as similar when they often appear together on (preferably short) paths of the network. The different expressions are obtained for both regular and hitting paths and serve as a basis to compute new covariance and correlation measures between nodes, which are valid positive semi-definite kernels on a graph. Experiments on semi-supervised classification problems show that the introduced similarity measures provide competitive results compared to other state-of-the-art distance and similarity measures between nodes.

In this paper, I was responsible of the implementation and the writing of the experimental part. I also took part in the reviewing process and the implementation of the kernels.

- (7) **P. Leleux, S. Courtain, K. François, and M. Sauerens. “Design of biased random walks on a graph with application to collaborative recommendation”. In: *Physica A: Statistical Mechanics and***

its Applications 590 (2022), pp. 1–20

This paper investigates a paths-based statistical physics formalism, inspired by the BoP framework [105], for the design of random walks on a graph in which the transition probabilities (the policy) are biased in favor of some node features. In addition, the same formalism allows the introduction of capacity constraints on the expected node visit rates and an algorithm to compute the optimal policy subject to such capacity constraints is developed. Simulation results indicate that the proposed procedure can be effectively used in order to define a Markov chain driving the walk towards nodes having some specific properties, like seniority, education level or low node degree (hub-avoiding walk). An application relying on this last property is proposed as a tool to improve serendipity in collaborative recommendation, and is tested on the MovieLens data.

My contribution in this paper was mainly to provide comments and suggestions during the writing and reviewing process. I also collaborated with Prof. M. Saerens on the development of the theory of capacity constraints on the expected node visit rates.

1.2 Structure of the thesis

The remainder of this thesis is structured into eight chapters as follows:

- ▶ Chapter 2 introduces graph theory by setting out basic concepts and some relevant matrices.
- ▶ Chapter 3 presents the bag-of-paths framework.
- ▶ Chapter 4 is based on (2) and extends the randomized shortest path model considering the introduction of capacity constraints into the model.
- ▶ Chapter 5 is based on (3) and studies a flow-constrained randomized shortest path formulation to the relative entropy-regularized optimal transport problem on a graph with multiple inputs and outputs having fixed marginal probabilities.
- ▶ Chapter 6 is based on (4) and derives two extensions of the bag-of-paths framework including a Poisson probability distribution on the path lengths and on the probability of sampling a path.
- ▶ Chapter 7 presents a comprehensive comparison of 43 similarity and dissimilarity measures between nodes through graph-based semi-supervised classification tasks.

- ▶ Chapter 8 is based on (1) and focuses on automatically detecting fraudulent e-commerce transactions using network-related features and the free energy distance developed in the bag-of-paths framework.
- ▶ Chapter 9 concludes this thesis.

Graphs and networks framework

2

Graph theory, or **network science**, finds its roots in the resolution of the famous problem of the seven bridges of Königsberg in Prussia (now Kaliningrad, Russia) proposed by Leonhard Euler [93] (see Fig. 2.1). In 1735, Euler demonstrated by introducing the notion of graph that it is not possible to walk across all seven bridges without crossing the same one twice. The property illustrated in his demonstration is nowadays known as the notion of Eulerian paths in a graph [235, 290].

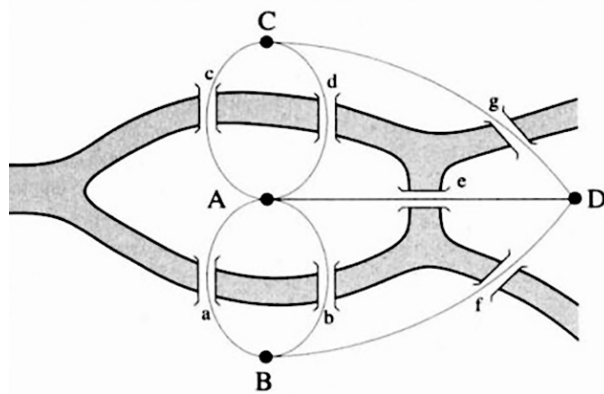


Figure 2.1: Illustration of the seven bridges of Königsberg and Euler's solution [31].

It was only in 1936, two centuries later, that D. König wrote the first book on graph theory [161]. Since then, graph theory has rapidly developed and has extended to a variety of well-known problems other than the Eulerian paths, such as traveling salesman problem, graph coloring, shortest path problem, to cite a few of them [235, 290]. As mentioned in Chapter 1, nowadays, graphs are increasingly used in many areas including, among others, chemistry, biology, physics, human sciences, and engineering [100, 103].

This chapter, largely inspired by [21, 103, 205], aims at introducing the graph theory by setting out basic concepts and some relevant matrices. However, the chapter does not provide an exhaustive summary of the graph theory and only primarily focuses on theoretical aspects that are relevant for

this thesis. For a wider review, the interested reader is advised to look at the available collection of textbooks covering this subject (see for example [2, 20, 21, 30, 103, 111, 205, 290]).

2.1 Basic concepts

A **network**, or **graph**, G is a mathematical structure containing a finite non-empty set $\mathcal{V}$ of n **nodes**, or **vertices**, and a set $\mathcal{E}$, which is a subset of $\mathcal{V} \times \mathcal{V}$, of m **edges**, or **links**, which pair nodes. Therefore, a graph is a collection of nodes linked by edges, or in mathematical expression $G(\mathcal{V}, \mathcal{E})$. Note that in this thesis, the terms network and graph are used interchangeably.

More generally, graphs are a convenient way of representing pairwise relations between objects. In the present case, these **objects**, or **entities** are the nodes while the existence of a **relation** between two objects is represented by the edges. As previously mentioned, a famous example of a graph is the network of friends on Facebook, where the nodes can be seen as people and the edges as the friendship relation between them.

2.1.1 Typology of graphs

In addition to being defined by a set of nodes and edges, the graphs are also characterized and named on the basis of four criteria related to the type of edges connecting their nodes. Nevertheless, there is no real consensus on these names because of the wide applicability of graph theory in many fields and also because the increasing interest for this theory is quite recent [235]. Therefore, this subsection and the next one present the typology and terminology of graphs that we are using in this thesis.

The first criterion concerns the number of links between two nodes. If there are multiple edges between two nodes, the graph is called a **multigraph** or a **multiple graph**. Whereas if the number of edges between two nodes is at most one, the graph is called a **simple graph**.

The second criterion is related to the presence of **self-loops**, or **self-edges**, in the graph. A self-loop is an edge that connects a node to itself. Generally, graphs are assumed to have no self-loops unless otherwise specified as ‘graphs with self-loops’. Furthermore, the term **pseudograph** is sometimes used to name a multigraph with self-loops.

The third criterion is based on the orientation of the edges. A graph is said to be **undirected**, or not oriented, if the edge (i, j) (i.e. the edge between a node i and a node j) is equivalent to edge (j, i) for all $(i, j) \in \mathcal{E}$. Otherwise, the graph is called a **directed**, or oriented, graph where the edges (i, j) and (j, i) are distinct. Moreover, an undirected edge between node i and node j

SIMPLE EDGES		Self-loops	No loops
Directed	Weighted	weighted directed simple graph with self-loops	weighted directed simple graph
	Unweighted	unweighted directed simple graph with self-loops	unweighted directed simple graph
Undirected	Weighted	weighted undirected simple graph with self-loops	weighted undirected simple graph
	Unweighted	unweighted undirected simple graph with self-loops	unweighted undirected simple graph
MULTIPLE EDGES		Self-loops	No loops
Directed	Weighted	weighted directed pseudograph	weighted directed multigraph
	Unweighted	unweighted directed pseudograph	unweighted directed multigraph
Undirected	Weighted	weighted undirected pseudograph	weighted undirected multigraph
	Unweighted	unweighted undirected pseudograph	unweighted undirected multigraph

Table 2.1: Summary of the various types of graphs.

is also commonly denoted by $i \leftrightarrow j$, whereas a directed edge is denoted by $i \rightarrow j$.

The fourth criterion concerns the weights on the edges. If the weights associated with each edge take binary values, 0 and 1, the graph is called an **unweighted graph**. Whereas if a nonnegative weight w_{ij} is associated with each edge, the graph is called a **weighted graph**. Furthermore, the interpretation of the weight differs according to whether or not the graph is directed. In the case of an **undirected weighted graph**, the weights are symmetric ($w_{ij} = w_{ji}$) and represent the degree of ‘affinity’ or ‘similarity’, or the ‘closeness’ between the nodes. However, the weights of a **directed weighted graph** usually represent the degree of endorsement, credit, reward, or dependency between the nodes.

All of these criteria can be combined to obtain a large number of graph types. The Table 2.1 summarizes all possible combinations. Finally, there are still other types of graphs defined according to the number of different types of nodes composing them.

A **bipartite graph** is a graph composed of two disjoint subsets $\mathcal{X}$ and $\mathcal{Y}$ of nodes such that no edge connects two nodes belonging to the same subset $\mathcal{X}$ or $\mathcal{Y}$. In other terms, each edge links a node in $\mathcal{X}$ to a node in $\mathcal{Y}$ and inversely. In a mathematical notation, this gives that for all of the edges $i \rightarrow j \in \mathcal{E}$, either $i \in \mathcal{X}$ and $j \in \mathcal{Y}$ or $i \in \mathcal{Y}$ and $j \in \mathcal{X}$, with $\mathcal{X} \cap \mathcal{Y} = \emptyset$ and $\mathcal{X} \cup \mathcal{Y} = \mathcal{V}$.

Following the same logic, a **tripartite graph** is a graph composed of three disjoint subsets $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ of nodes (i.e. $\mathcal{X} \cap \mathcal{Y} = \emptyset$, $\mathcal{X} \cap \mathcal{Z} = \emptyset$, $\mathcal{Y} \cap \mathcal{Z} = \emptyset$ and $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z} = \mathcal{V}$) such that each edge links two nodes of two different subsets $\mathcal{X}$, $\mathcal{Y}$ or $\mathcal{Z}$. A concrete example of tripartite graph is presented in Chapter 8. Moreover, this notion can theoretically be extended up to n subsets where each node of the graph is considered as a subset.

2.1.2 Terminology of graphs

Graphs are also accompanied by their own specific terminology. For example, two nodes connected by an edge are called **adjacent** and are considered **neighbors**. Furthermore, an edge connected to a node is said to be **incident**

to this node.

In the case of a directed edge $i \rightarrow j$, the node i is called the **predecessor** of node j or the **start node**, while the node j is called the **successor** of node i or the **end node**. Common notation for the set of predecessors and successors of node i are $\mathcal{Pred}(i) = \{j \mid i \rightarrow j \in \mathcal{E}\}$ and $\mathcal{Succ}(i) = \{j \mid j \rightarrow i \in \mathcal{E}\}$ respectively. In the case of an undirected graph, these two sets are equivalent and correspond to the set of neighbors of the node i denoted by $\mathcal{N}(i) = \{j \mid (i, j) \in \mathcal{E}\}$.

A **subgraph** H of a graph G is a graph consisting of a set of nodes and edges that are a subset of the original graph G (i.e. $\mathcal{V}(H) \subseteq \mathcal{V}(G)$ and $\mathcal{E}(H) \subseteq \mathcal{E}(G)$). A common notation for a subgraph H is $H \subseteq G$ meaning that G contains H .

A **connected** graph, or **strongly connected** graph for the directed case, is a graph in which each node is reachable from any other node. In other words, a graph is connected if it comprises only one **connected component**. A directed graph is called **weakly connected** if it becomes connected when all of its directed edges are replaced with undirected edges. Whereas a graph that is not connected is called **disconnected**. A disconnected graph is composed of a set of connected components which can be seen as a set of maximal connected subgraphs that are not connected to each other. The term maximal means that each subgraph contains the largest numbers of nodes and edges taking into account that there is no edge between two different subgraphs.

2.2 Common associated matrices

The use of matrices and the power of matrix algebra has played an important role in the rapid development of graph theory during recent decades [2, 20, 103, 290]. In this section, we present matrices and measures that are particularly relevant for the rest of this thesis. In the remainder of this thesis, the column vectors and matrices are represented by bold lowercase and uppercase letters respectively.

2.2.1 Adjacency matrix

The **adjacency matrix** $\mathbf{A}$ is a square matrix of size $n \times n$ that allows to represent the structure of a graph. In the case of undirected graphs, this matrix is symmetric ($a_{ij} = a_{ji}$ or $\mathbf{A} = \mathbf{A}^T$ where $\mathbf{A}^T$ denotes the transpose of matrix $\mathbf{A}$) which is generally not the case for a directed graph. When the graph is unweighted, the element a_{ij} of the matrix can only take binary values and thus can be represented as

$$a_{ij} = [\mathbf{A}]_{ij} \triangleq \begin{cases} 1 & \text{if there is an edge between nodes } i \text{ and } j, \\ 0 & \text{otherwise.} \end{cases} \quad (2.1)$$

The adjacency matrix could also be defined for a weighted graph as

$$a_{ij} = [\mathbf{A}]_{ij} \triangleq \begin{cases} w_{ij} & \text{if there is an edge between nodes } i \text{ and } j, \\ 0 & \text{otherwise.} \end{cases} \quad (2.2)$$

In the case of the absence of self-loops in the graph, the adjacency matrix $\mathbf{A}$ only contains 0's on its diagonal. When the graph is bipartite, the adjacency matrix takes the following form

$$\mathbf{A}_{bi} = \begin{bmatrix} \mathbf{0}_{\mathcal{X} \times \mathcal{X}} & \mathbf{A}_{\mathcal{X} \times \mathcal{Y}} \\ \mathbf{A}_{\mathcal{Y} \times \mathcal{X}} & \mathbf{0}_{\mathcal{Y} \times \mathcal{Y}} \end{bmatrix} \quad (2.3)$$

where $\mathbf{0}_{\mathcal{X} \times \mathcal{X}}$ and $\mathbf{0}_{\mathcal{Y} \times \mathcal{Y}}$ are two matrices full of 0s, $\mathbf{A}_{\mathcal{X} \times \mathcal{Y}}$ and $\mathbf{A}_{\mathcal{Y} \times \mathcal{X}}$ are two **biadjacency matrices**. The two matrices full of 0s indicate that no edge connects two nodes belonging to the same subset $\mathcal{X}$ or $\mathcal{Y}$ while the biadjacency matrices represent the links between elements of two different subsets.

More than providing an algebraic representation of the graph G , the adjacency matrix can be used to extract information, or measures, about the underlying graph and its structure. For example, one could measure the volume of the graph G , which represents the sum of all the elements of the adjacency matrix.

$$\text{vol}(G) = \sum_{i,j=1}^n a_{ij} = a_{\bullet\bullet} \quad (2.4)$$

2.2.2 Cost matrix

The **cost matrix** $\mathbf{C}$ can also be used to represent a graph. This matrix is based on costs associated with each edge of the graph rather than on the affinities between nodes as in the case with the adjacency matrix $\mathbf{A}$. This cost is non negative and one way to compute it is to use the elements of $\mathbf{A}$, as $c_{ij} = 1/a_{ij}$ or simply $c_{ij} = 1$. The cost matrix is defined as

$$[\mathbf{C}]_{ij} \triangleq \begin{cases} c_{ij} & \text{if there is an edge between nodes } i \text{ and } j, \\ \infty & \text{otherwise.} \end{cases} \quad (2.5)$$

Note that the costs can also be set independently from the adjacency matrix $\mathbf{A}$. In that case, the cost of a transition can be defined, for example, on the basis of some features of the nodes or the edges, depending on the problem to solve.

2.2.3 Degree matrices

The degree matrices are square and diagonal matrices of size $n \times n$ where the elements of the diagonal are the degrees of each node. However, the notion of degree takes different interpretations depending on whether the graph is directed or undirected.

In an undirected graph, the **degree** of a node is the number of links it has with the other nodes or equivalently the number of nodes adjacent to it. This number can be directly computed from the adjacency matrix. The degree d_i of a node i corresponds to the sum of row i , $\sum_{j=1}^n a_{ij} = a_{i\bullet}$, which is equal to the sum of the column i , $\sum_{j=1}^n a_{ji} = a_{\bullet i}$, in the case of an undirected graph. The **degree vector**, of size $n \times 1$, can also be defined from the adjacency matrix as $\mathbf{d} = \mathbf{A}\mathbf{e}$ where $\mathbf{e}$ is a column vector of length n containing 1s (i.e. $\mathbf{e} = [1, 1, \dots, 1]^T$). Finally, the **degree matrix** $\mathbf{D}$ can be defined alternatively from this vector as $\mathbf{D} = \text{Diag}(\mathbf{d})$, where **Diag** is an operator that, applied to a vector, allows to define a diagonal matrix with this vector as diagonal. When this operator is applied to a square matrix, it allows to define a diagonal matrix containing the diagonal of the square matrix.

In a directed graph, the notion of degree needs to be nuanced by introducing two different notions namely, the **indegree** and the **outdegree**. The indegree of a node is the number of edges incoming to it and the outdegree of a node is the number of outgoing edges. From the adjacency matrix, the indegree of a node i corresponds to the sum of column i , $\sum_{j=1}^n a_{ji} = a_{\bullet i}$, and the outdegree of a node i corresponds to the sum of row i , $\sum_{j=1}^n a_{ij} = a_{i\bullet}$. In matrix form, the $n \times 1$ **indegree vector** $\mathbf{d}_i$ and the $n \times 1$ **outdegree vector** $\mathbf{d}_o$ are respectively denoted by $\mathbf{d}_i = \mathbf{A}^T \mathbf{e}$ and $\mathbf{d}_o = \mathbf{A}\mathbf{e}$. Finally, from these vectors, the **indegree matrix** $\mathbf{D}_i$ and the **outdegree matrix** $\mathbf{D}_o$ can be defined as $\mathbf{D}_i = \text{Diag}(\mathbf{d}_i)$ and $\mathbf{D}_o = \text{Diag}(\mathbf{d}_o)$.

2.2.4 Reference transition probability matrix

The **reference transition probability matrix** $\mathbf{P}$ is a square matrix of size $n \times n$ defined as

$$\mathbf{P} = \mathbf{D}_o^{-1} \mathbf{A} \quad (2.6)$$

Note that this formulation is only valid for graphs that do not contain **dangling nodes**, that is, nodes with an outdegree equal to zero. This matrix has the property of being **row-stochastic** which means that each of its rows sums to 1.

Furthermore, the transition matrix is linked to the notion of natural **random walk** on a graph. A random walk is a Markov chain describing a sequence of nodes visited by a random walker where the probability that the random walker moves from a node, or a state, i to an adjacent node, or state,

j is known [102, 239]. More specifically, this probability, or reference probability, corresponds to the elements of the transition matrix p_{ij} which can also be defined elementwise as

$$p_{ij} \triangleq \frac{a_{ij}}{\sum_{j=1}^n a_{ij}} = \frac{a_{ij}}{a_{i\bullet}} \quad (2.7)$$

For more information about Markov chains and random walks, the interested reader is advised to consult the following standard textbooks [103, 118, 149, 212, 236].

2.2.5 Laplacian matrices

The Laplacian matrix is fundamental and used in many different contexts. Through these contexts, the Laplacian matrix generally appears in two different forms: the (unnormalized) Laplacian matrix and the normalized Laplacian matrix.

For an undirected graph without self-loops, the **Laplacian matrix** $\mathbf{L}$ [59] is a square matrix of size $n \times n$, defined as

$$\mathbf{L} \triangleq \mathbf{D} - \mathbf{A} \quad (2.8)$$

This matrix is positive semidefinite which means that all its eigenvalues are non-negative. Furthermore, the Laplacian matrix $\mathbf{L}$ is doubly centered; in other words, the sum of the elements for each column or row is equal to 0 ($\mathbf{L}\mathbf{e} = \mathbf{0}$ and $\mathbf{e}^T \mathbf{L} = \mathbf{0}^T$ where $\mathbf{0}$ is a column vector of length n full of 0s). Therefore, the matrix $\mathbf{L}$ is rank deficient which means that it has at least one zero eigenvalue. Moreover, the number of zero eigenvalues corresponds to the number of connected components of the graph G [59].

However, due to its rank deficiency, the inverse of the matrix $\mathbf{L}$ is not well defined. Briefly, one alternative solution is to use the **Moore-Penrose pseudoinverse** based on the singular value decomposition [9] of the Laplacian matrix $\mathbf{L}$. The Moore-Penrose pseudoinverse, or generalized inverse, extends the matrix inverse to cases where the normal inverse is not defined as rank-deficient matrices, or non-square matrices cases [6, 9, 22, 135, 229].

By denoting the eigenvalues and eigenvectors of $\mathbf{L}$ as $\{\lambda_k, \mathbf{u}_k\}$, the **pseudoinverse of the Laplacian matrix** $\mathbf{L}^+$ is

$$\mathbf{L}^+ = \sum_{k=1}^{n-r} \frac{1}{\lambda_k} \mathbf{u}_k \mathbf{u}_k^T \quad (2.9)$$

where the r zero eigenvalues are not taken into account.

Moreover, by associating any x_i value to each node, it is possible to define an autocorrelation measure from the Laplacian matrix $\mathbf{L}$ as

$$\mathbf{x}^T \mathbf{L} \mathbf{x} = \frac{1}{2} \sum_{i,j=1}^n a_{ij} (x_i - x_j)^2 = \frac{1}{2} \sum_{i=1}^n \sum_{j \in \mathcal{N}(i)} a_{ij} (x_i - x_j)^2 \quad (2.10)$$

where $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ is the vector containing the value x_i of each node. The autocorrelation measures the extent of the proximity of the value x_i of each node i to the values x_j associated with its direct neighborhood $j \in \mathcal{N}(i)$. The autocorrelation reaches a minimum value of zeros when the values x_i are constant. Therefore, the autocorrelation can be seen as a measure of the values x_i distribution's smoothness across the graph.

However, the formulation of the autocorrelation measure presented in Eq. 2.10 gives greater importance to high-degree nodes, also called hubs. Indeed, these nodes appear more frequently in the sum than nodes with a lower number of links. One solution to this effect is to rescale the values x_i using the **normalized Laplacian matrix** $\tilde{\mathbf{L}}$ instead of the Laplacian matrix $\mathbf{L}$ in Eq. 2.10.

The **normalized Laplacian matrix** $\tilde{\mathbf{L}}$ [59] is defined as

$$\tilde{\mathbf{L}} \triangleq \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \quad (2.11)$$

where $\mathbf{I}$ is the **identity matrix**, a matrix of size $n \times n$ with 1's on the main diagonal and 0's elsewhere. Furthermore, the matrix $\tilde{\mathbf{L}}$ has the same properties as the Laplacian matrix and its pseudoinverse.

2.2.6 Modularity matrix

The **modularity** is a measure of the quality of a particular division of a network into communities or clusters, which comes from the field of physics and was introduced by Newman and Girvan in [203, 206]. More precisely, the modularity corresponds to the difference between the observed quantity of within-cluster edges and the expected value of the same quantity in a network with the same division but where the nodes are randomly linked [204, 206]. This quantity is defined as

$$Q = \frac{1}{\text{vol}(G)} \sum_{i=1}^n \sum_{j=1}^n \left(a_{ij} - \frac{d_i d_j}{\text{vol}(G)} \right) \delta(\text{cl}(i) = \text{cl}(j)) \quad (2.12)$$

where $\text{cl}(i)$ is the cluster to which node i belongs, d_i is the element i of the degree vector (see Subsection 2.2.3), and $\delta(\text{cl}(i) = \text{cl}(j))$ is the Kronecker delta which is 1 when i and j belong to the same cluster ($\text{cl}(i) = \text{cl}(j)$) and 0 otherwise.

The modularity can only take values between $-1/2$ and 1 inclusive (see [39] for details). A modularity close to 1 implies a strong community structure. However, this is rarely reached. In practice, modularity greater than 0.3 is an indicator of significant community structure [61]. Whenever all the nodes of a network are grouped in a single cluster, the modularity is equal to 0. Finally, the modularity is negative whenever each node belongs to a separate cluster [21].

The **modularity matrix** $\mathbf{Q}$ can be defined as [103]

$$\mathbf{Q} = \mathbf{A} - \frac{(\mathbf{A}\mathbf{e})(\mathbf{e}^T\mathbf{A})}{\mathbf{e}^T\mathbf{A}\mathbf{e}} = \mathbf{A} - \frac{\mathbf{d}_o\mathbf{d}_i^T}{\text{vol}(G)} \quad (2.13)$$

where $\mathbf{d}_i$ is the indegree vector and $\mathbf{d}_o$ is the outdegree vector (see Subsection 2.2.3).

The matrix $\mathbf{Q}$ has the same property as the Laplacian matrix $\mathbf{L}$ introduced earlier. Both the rows and the columns of these matrices sum to zero, implying that they always have an eigenvector with an eigenvalue equal to zero [204]. Note that the modularity criterion can be rewritten based on this matrix $\mathbf{Q}$ as

$$Q = \frac{1}{\text{vol}(G)} \sum_{k=1}^{nc} \mathbf{v}_k^T \mathbf{Q} \mathbf{v}_k \quad (2.14)$$

where nc is the number of clusters and $\mathbf{v}_k$ is the binary membership vector for each cluster k . That is, v_{ik} is equal to 1 when node i is in cluster k and 0 otherwise.

2.2.7 An illustrative example

To wrap this section up, an example of most of the matrices defined in the previous subsections is illustrated based on a small undirected and unweighted graph presented in Fig. 2.2.

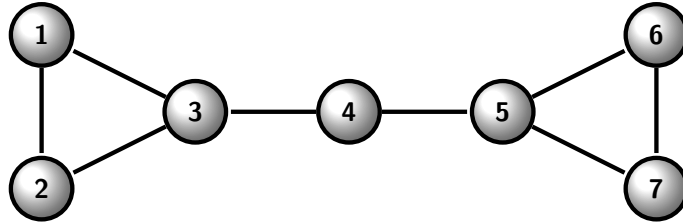


Figure 2.2: A small undirected and unweighted graph G .

The adjacency matrix $\mathbf{A}$ associated with this graph is

$$\mathbf{A} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{matrix}$$

Its degree vector $\mathbf{d}$ and degree matrix $\mathbf{D}$ are

$$\mathbf{d} = \begin{matrix} & \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{bmatrix} 2 \\ 2 \\ 3 \\ 2 \\ 3 \\ 2 \\ 2 \end{bmatrix} \end{matrix} \quad \mathbf{D} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \end{matrix}$$

Its reference transition probability matrix $\mathbf{P}$ is

$$\mathbf{P} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{bmatrix} 0 & 0.50 & 0.50 & 0 & 0 & 0 & 0 \\ 0.50 & 0 & 0.50 & 0 & 0 & 0 & 0 \\ 0.33 & 0.33 & 0 & 0.33 & 0 & 0 & 0 \\ 0 & 0 & 0.50 & 0 & 0.50 & 0 & 0 \\ 0 & 0 & 0 & 0.33 & 0 & 0.33 & 0.33 \\ 0 & 0 & 0 & 0 & 0.50 & 0 & 0.50 \\ 0 & 0 & 0 & 0 & 0.50 & 0.50 & 0 \end{bmatrix} \end{matrix}$$

Its Laplacian matrix $\mathbf{L}$ is

$$\mathbf{L} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix} \end{matrix}$$

Its pseudoinverse of the Laplacian matrix $\mathbf{L}^+$ is

$$\mathbf{L}^+ = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{bmatrix} 0.97 & 0.64 & 0.45 & -0.12 & -0.55 & -0.69 & -0.69 \\ 0.64 & 0.97 & 0.45 & -0.12 & -0.55 & -0.69 & -0.69 \\ 0.45 & 0.45 & 0.59 & 0.02 & -0.41 & -0.55 & -0.55 \\ -0.12 & -0.12 & 0.02 & 0.45 & 0.02 & -0.12 & -0.12 \\ -0.55 & -0.55 & -0.41 & 0.02 & 0.59 & 0.45 & 0.45 \\ -0.69 & -0.69 & -0.55 & -0.12 & 0.45 & 0.97 & 0.64 \\ -0.69 & -0.69 & -0.55 & -0.12 & 0.45 & 0.64 & 0.97 \end{bmatrix} \end{matrix}$$

Its modularity matrix $\mathbf{Q}$ is

$$\mathbf{Q} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{bmatrix} -0.25 & 0.75 & 0.63 & -0.25 & -0.38 & -0.25 & -0.25 \\ 0.75 & -0.25 & 0.63 & -0.25 & -0.38 & -0.25 & -0.25 \\ 0.63 & 0.63 & -0.56 & 0.63 & -0.56 & -0.38 & -0.38 \\ -0.25 & -0.25 & 0.63 & -0.25 & 0.63 & -0.25 & -0.25 \\ -0.38 & -0.38 & -0.56 & 0.63 & -0.56 & 0.63 & 0.63 \\ -0.25 & -0.25 & -0.38 & -0.25 & 0.63 & -0.25 & 0.75 \\ -0.25 & -0.25 & -0.38 & -0.25 & 0.63 & 0.75 & -0.25 \end{bmatrix} \end{matrix}$$

2.3 Similarity and dissimilarity measures

This section aims to introduce the concepts of similarity and dissimilarity in a graph, and two well-known dissimilarities, which are the shortest path distance and the commute time distance.

2.3.1 Basic concepts

Similarity has for many years been an important concept, which is widely used not only in network analysis but also in many fields of research. However, the definition of a similarity measure s between two nodes is little standardized through the literature [103].

Nevertheless, the most common way to define a similarity measure is based on some properties that should reasonably be displayed. First, it is a symmetric measure, that is $s_{ij} = s_{ji}$ for all i, j . Secondly, the similarity is positive, $s_{ij} \geq 0$, and increases with the similarity of the two nodes. Thirdly, the similarity between two nodes i, j is always lower or equal to the similarity measure of the node i or j itself (i.e. $s_{ij} \leq s_{ii}$ or $s_{ij} \leq s_{jj}$) [103, 114].

On the contrary, **dissimilarity**, the dual of the similarity, is a quite standard notion and is clearly defined through the literature [81, 114, 270]. The dissimilarity, measure, index, or coefficient over a set of objects, or in this thesis a set of nodes, is denoted by Δ . Furthermore, the dissimilarity between two nodes i and j is denoted by Δ_{ij} and must possess the following characteristics:

- **Nonnegativity:** $\Delta_{ij} \geq 0 \forall i, j$
- **Symmetry:** $\Delta_{ij} = \Delta_{ji} \forall i, j$
- **Reflexivity:** $\Delta_{ij} = 0$ if and only if $i = j$

If, in addition to these three characteristics, the dissimilarity satisfies the **triangle inequality**,

$$\Delta_{ij} \leq \Delta_{ik} + \Delta_{kj} \forall i, j, k \quad (2.15)$$

is then called a **distance metric**, a **distance measure**, or a **metric**. Moreover, a distance measure is called **Euclidean** if a configuration exists for any objects i and j in a Euclidean space that exactly keeps the initial distance between these two objects: i and j . Finally, an Euclidean distance can be defined in terms of similarity, inner products in the Euclidean space, as $\Delta_{ij} = (s_{ii} + s_{jj} - 2s_{ij})^{\frac{1}{2}}$ [103, 114].

2.3.2 Shortest path distance

As the name indicates, the **shortest path distance**, or **geodesic distance**, is closely related to the notion of path on a graph. A **path** φ is defined as a finite sequence of jumps between adjacent nodes in a graph, beginning at a starting node i and finishing at an ending node j . Generally, there are several paths of different **lengths** $\ell(\varphi)$ (number of jumps) between each pair of nodes. In an unweighted graph, the shortest path distance between two nodes is simply defined as the path with the smallest length between these two nodes. In the case of a weighted graph, the notion of length of a path corresponds to the total cost of a path $\tilde{c}(\varphi)$ that is to say the sum of the direct cost over all edges along the path φ . Therefore, the shortest path distance between two nodes can be defined as

$$\Delta_{ij}^{\text{sp}} \triangleq \min_{\varphi \in \mathcal{P}_{ij}} \{\tilde{c}(\varphi)\} \quad (2.16)$$

where $\mathcal{P}_{ij}$ is the set of all paths connecting node i to node j . If there is no path between two nodes then the distance between them is considered infinite. An infinite distance is a sign that the graph is disconnected.

However, the objective here is to compute the shortest path distance not only between two nodes but also between all pairs of nodes, and to record all the distances in a matrix Δ^{sp} called shortest-path distance matrix. The computation of this distance in a weighted directed graph is of fundamental importance [103]. Therefore, there exists a variety of techniques to compute a distance matrix depending on the specific features of the problem (i.e. negative weights, the sparsity of the graph, ...). Among these methods, **Floyd-Warshall algorithm** [97, 289] is one of the most well-known. This algorithm allows to compute the shortest path distance between all pairs of nodes with a relatively efficient compact matrix form [111, 151].

2.3.3 Commute time distance

As the reference transition probability matrix, the **commute time distance** is based on a random walk model. Intuitively, this distance represents the average time required for a random walker to go back and forth between a node i and a node j . In order to define the commute time distance, two

interesting quantities need to be introduced, namely the average first passage time and the average commute time.

The average first passage time $m(i, j)$ is the expected number of steps taken by a random walker to visit node j for the first time starting from a node or state i . This measure can be computed by the following formula [103]

$$\begin{cases} m(i, j) = 1 + \sum_{k=1}^n p_{ik} m(k, j) & \text{for } i \neq j \\ m(j, j) = 0 & \text{for } i = j \end{cases} \quad (2.17)$$

There exists another formulation of this quantity based on the matrix $\mathbf{L}^+$, presented in Subsection 2.2.5, for an undirected graph which is [103]

$$m(i, j) = \sum_{k=1}^n (l_{ik}^+ - l_{ij}^+ - l_{jk}^+ + l_{jj}^+) d_k \quad (2.18)$$

where d_k denotes the degree of node k .

The average commute time $n(i, j)$ is the average number of steps needed for a random walker to reach a node or state j from i for the first time and to go back to i . This measure is symmetric, as opposed to the average first passage time, and is defined as [103, 102, 239]

$$n(i, j) = m(i, j) + m(j, i) \quad (2.19)$$

This measure can also be defined based on the matrix $\mathbf{L}^+$ as [103]

$$n(i, j) = \text{vol}(G)(l_{ii}^+ + l_{jj}^+ - 2l_{ij}^+) \quad (2.20)$$

This quantity is a valid metric because it respects all the conditions seen in Subsection 2.3.1, and is commonly referred to as the commute time distance. This distance measure is closely related to the **effective resistance** [198, 248, 247] and the **resistance distance** [157]. The resistance distance is computed using the effective resistance between two nodes and the main difference with the commute time distance is the factor $\text{vol}(G)$ [47, 300].

The commute time distance matrix can now be defined as [103]

$$\Delta^{\text{CT}} = \text{vol}(G) (\mathbf{diag}(\mathbf{L}^+) \mathbf{e}^T + \mathbf{e}(\mathbf{diag}(\mathbf{L}^+))^T - 2\mathbf{L}^+) \quad (2.21)$$

where $\mathbf{diag}$ is an operator that, when applied to a square matrix, allows to define a column vector containing the diagonal of the square matrix. Furthermore, the elements of this matrix can be defined as

$$\Delta_{ij}^{\text{CT}} \triangleq n(i, j) = \text{vol}(G)(\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{L}^+ (\mathbf{e}_i - \mathbf{e}_j) \quad (2.22)$$

$\mathbf{e}_i$ and $\mathbf{e}_j$ are respectively the i th and the j th columns of the fundamental matrix $\mathbf{I}$.

2.4 Kernels and derived distances

This section aims to introduce the concept of kernel in a broad sense and to explain how to compute a kernel matrix from a Euclidean distance matrix and vice versa.

2.4.1 Basic concepts

A **kernel** is a function $k(i, j) : \Omega \times \Omega \rightarrow \mathbb{R}$ that returns a real number when given two objects $i, j \in \Omega$, defined in an **input space** Ω , that can be considered as a similarity measure (see Subsection 2.3.1 for details) between these two objects. These objects may take various forms such as sequences, graphs, documents, or, in the context of this thesis, nodes of a graph [103, 249, 304].

Concretely, the similarity $k(i, j)$ is equivalent to computing the inner product, also called dot product, of some mapping ϕ of the two objects i and j , such as $\mathbf{x}_i = \phi(i)$ and $\mathbf{x}_j = \phi(j)$, in an inner product space called **embedding space**. Therefore, $k(i, j) = \langle \phi(i), \phi(j) \rangle = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ where $\mathbf{x}_i$ and $\mathbf{x}_j$ are vector representations of the two objects i and j , called **object vectors** or **node vectors** if the objects are nodes of a graph, in the embedding space. Usually, the kernel directly computes the similarity without needing to know or to explicitly compute the representation [103, 304]. Furthermore, the kernel function must have two properties to be valid. The first property is that it must be symmetric and the second one is that it must be positive semidefinite [241, 249].

Thus, a **kernel matrix** $\mathbf{K}$ can be defined as a matrix of size $n \times n$ which computes the kernel function $k(i, j)$ between $n \times n$ pairs of objects indexed by i, j . This matrix $\mathbf{K}$, as the kernel function, needs to be symmetric positive semidefinite to be a **valid kernel**. These properties are ensured when all the eigenvalues of $\mathbf{K}$ are nonnegative or when $\mathbf{K}$ is a Gram (inner product) matrix (i.e. $\mathbf{K} = \mathbf{X}\mathbf{X}^T$ where $\mathbf{X}$ is the **data matrix** containing the $\mathbf{x}_i^T$ on its rows) [213] or, also if the inequality $\mathbf{x}^T \mathbf{K} \mathbf{x} \geq 0$ is respected for all $\mathbf{x} \in \mathbb{R}^n$ [193].

Furthermore, from the spectral decomposition theorem [191, 210] applicable to every symmetric positive semidefinite matrix, the kernel matrix $\mathbf{K}$ can also be defined as

$$\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}(\mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}})^T = \mathbf{X}\mathbf{X}^T \quad (2.23)$$

where $\mathbf{U}$ is an orthogonal matrix, i.e. $\mathbf{U}^T \mathbf{U} = \mathbf{I}$, containing the eigenvectors of $\mathbf{K}$, $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_n]$, and $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\mathbf{K}$ sorted in decreasing order. From this equation, the data matrix $\mathbf{X}$ can also be redefined as $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}$ which defines the embedding [103].

2.4.2 Distances from kernels and vice versa

In the literature, two methods are mainly used to transform a distance to a kernel. These methods are namely the Gaussian kernel and the multidimensional scaling.

Firstly, based on the theory of **Gaussian kernel** [241], a kernel matrix $\mathbf{K}$ can be obtained starting from an Euclidean distance as

$$\mathbf{K} = \exp \left[-\frac{\Delta^{(2)}}{2\sigma^2} \right] \quad (2.24)$$

where $\Delta^{(2)}$ is the squared distances matrix and σ is an appropriate ‘width’ parameter required to be tuned [103, 304]. A good estimator of the parameter σ^2 is the variance of the distances defined as

$$\sigma^2 = \frac{\sum_{i=1}^n \sum_{j=1}^n \Delta_{ij}^2}{n * (n - 1)} \quad (2.25)$$

Secondly, based on the theory of classical **multidimensional scaling** [35], a kernel matrix $\mathbf{K}$ can be obtained starting from an Euclidean distance as

$$\mathbf{K} = -\frac{1}{2} \mathbf{H} \Delta^{(2)} \mathbf{H} \quad (2.26)$$

where $\mathbf{H}$ is the symmetric centering matrix,

$$\mathbf{H} = \mathbf{I} - \frac{\mathbf{e}\mathbf{e}^T}{n} \quad (2.27)$$

The kernel matrix $\mathbf{K}$ defined by Eq. 2.26 is positive semidefinite if and only if the distance is embeddable in an Euclidean space. Furthermore, the distance is invariant when the origin of the coordinates changes, but that it is not the case for the inner product. In that case, the origin of the coordinates is fixed to the centroid of the node vector, through the left and right $\mathbf{H}$ multiplications, in order to obtain a unique kernel matrix [103].

On the other side, the kernel matrix $\mathbf{K}$ induces directly an Euclidean distance metric between two nodes i and j as [103, 104]

$$\Delta_{ij}^2 = k_{ii} + k_{jj} - 2k_{ij} \quad (2.28)$$

and the distance matrix containing the elements Δ_{ij}^2 is computed in matrix form as

$$\Delta^{(2)} = \mathbf{diag}(\mathbf{K})\mathbf{e}^T + \mathbf{e}(\mathbf{diag}(\mathbf{K}))^T - 2\mathbf{K} \quad (2.29)$$

The interested reader is advised to consult the Section 10.3 of [103] for more details on the development of these equations.

Now that the fundamental concepts of graph theory and some relevant matrices have been introduced, we present in the next chapter the framework on which this thesis is based, the bag-of-paths framework.

Bag-of-paths framework | 3

This chapter introduces a general framework called **bag-of-paths**. The bag-of-paths framework was recently introduced and mainly developed in [105, 156, 238] which largely inspired this chapter. In this thesis, three extensions of the bag-of-paths framework are proposed in Chapter 4, Chapter 5, and Chapter 6. We also discuss two applications of this framework through a graph-based semi-supervised classification task in Chapter 7 and through a fraud detection problem in Chapter 8.

The main intention behind the development of the bag-of-paths framework is to propose a new family of dissimilarity measures to deal with the flaws of the shortest path distance (see Subsection 2.3.2) and the commute time distance (see Subsection 2.3.3). While relevant in many applications, these distances cannot always be considered as good candidate distances in network data for multiple reasons [105].

Regarding the shortest path distance, it only relies on the shortest path between the nodes of the graph and thus does not integrate the **degree of connectivity** between them. As for the commute time distance, when the size of the graph increases, it converges to a useless value that only depends on the degrees of the nodes. This phenomenon is known as the random walker getting ‘lost in space’ problem [284]. Moreover, the commute time distance assumes a completely random movement in the network, which is unrealistic for many applications. To sum up, the shortest path distance does not integrate the whole structure of the graph by not taking into account the connectivity [105], whereas the commute time distance, due to its links with the random walks model, quickly loses the notion of proximity to the initial node when the graph becomes larger [284].

To avoid these drawbacks, the bag-of-paths framework defines a family of dissimilarity measures that captures the notion of relative accessibility between nodes, in other words, a combination of both proximity and amount of connectivity in the network. These quantities interpolate between the shortest path distance and the commute time distance, depending on a temperature parameter T . On one hand, when T is large, i.e. $T \rightarrow \infty$, the dissimilarity measure is reduced to the commute time distance. On the other hand, when T is small, i.e. $T \rightarrow 0^+$, the dissimilarity measure is reduced to the shortest path distance [103, 105].

In addition to the dissimilarity measures defined in the bag-of-paths fram-

ework, other new families of distances combining information of both proximity and amount of connectivity in the network have recently received attention [123, 176]. Some of these distances are based on combinatorial analysis arguments, such as the logarithmic forest distance [50] or more globally the walk distances [51, 52]. Other distances are built on electrical networks and extensions, such as the p -resistance distance [5, 128] and the R_p and R_{12} distances [208]. Still other distances are based on L1-L2 regularization network flow model [178, 185], as well as on stochastic calculus such as the Laplace transformed hitting time distance [125].

The remainder of this chapter is organized as follows. Section 3.1 summarizes the bag-of-paths model while Section 3.2 describes the bag-of-hitting-paths model. Then, Section 3.3 presents a closely related model that is the randomized shortest path model. Finally, Section 3.4 introduces three dissimilarity measures derived from these models.

3.1 Bag-of-paths model

Initially, the original idea of the bag-of-paths (BoP) model was introduced in [238] and was inspired by a stochastic transportation model [4, 25]. The model is based on the probability that a path $\wp$ randomly drawn in a **bag of paths** begins at node i and ends at node j [103, 105, 172, 190]. This bag of paths is assumed to be a bag containing objects following three properties. First, the objects contained in this bag are paths of arbitrary length. As we already mentioned in Subsection 2.3.2, a **path** $\wp$ is a sequence of transitions to adjacent nodes on G beginning at a start node $s(\wp) = i$ and finishing at an end node $e(\wp) = j$. A path can also be denoted as $\wp_{ij}$ to emphasize the start and end nodes. Moreover, the **length** of a path $\ell(\wp)$ is the number of jumps needed to follow that path. Secondly, the bag is constituted of an infinite, but countable, number of paths. Thirdly, each path is weighted according to its quality, that is, its total cost. The **total cost** of a path $\tilde{c}(\wp)$ is defined as the sum of the cost c_{ij} (see Subsection 2.2.2) over all edges along the path $\wp$ [103].

Following [105], to define the probability of drawing a path from the bag of paths, two other notions need to be introduced. The first one is the **set of paths** between a node i and a node j including cycle denoted by $\mathcal{P}_{ij} = \{\wp_{ij}\}$. According to the properties stated above, $\mathcal{P}_{ij}$ usually contains an infinite, but countable, number of paths $\wp_{ij}$. The second one is the **set of all paths** through the graph $\mathcal{P} = \bigcup_{i,j=1}^n \mathcal{P}_{ij}$.

Now, the probability of drawing a path $\wp \in \mathcal{P}$ in the bag of paths, which is the probability distribution of the set $\mathcal{P}$, can be defined as the probability distribution $P(\cdot)$ minimizing the total expected cost $\mathbb{E}[\tilde{c}(\wp)]$, favoring the *exploitation*, among all the distributions having a fixed **relative entropy** J_0 . The relative entropy is computed with respect to a reference distribution, here

the natural random walk of the graph (see Eq. 2.7) as in [105], allowing some random *exploration*.

The choice of this reference distribution naturally defines a probability distribution on the set of paths such that high-cost paths, or long paths, occur with a low probability while low-cost paths, or short paths, occur with a high probability [105].

In other words, it is the same as seeking for path probabilities, $P(\wp)$, $\wp \in \mathcal{P}$, minimizing the total expected cost subject to a constant relative entropy constraint:

$$\left\{ \begin{array}{ll} \underset{\{P(\wp)\}}{\text{minimize}} & \sum_{\wp \in \mathcal{P}} P(\wp) \tilde{c}(\wp) \\ \text{subject to} & \sum_{\wp \in \mathcal{P}} P(\wp) \log (P(\wp)/\tilde{P}(\wp)) = J_0 \\ & \sum_{\wp \in \mathcal{P}} P(\wp) = 1 \end{array} \right. \quad (3.1)$$

where $J_0 > 0$ is provided a priori by the user, according to the desired degree of randomness and $\tilde{P}(\wp)$ represents the probability of following the path $\wp$ when walking according to the natural random walk reference distribution, that is, using transition probabilities p_{ij} (see Eq. 2.7 and Subsection 2.2.4).

More precisely, the **likelihood of the path** $\tilde{\pi}(\wp)$, when the start and end nodes are known, can be defined as the product of the transition probabilities along the path $\wp$, $\tilde{\pi}(\wp) \triangleq \prod_{\tau=1}^t p_{k_{\tau-1}k_{\tau}}$ where the walker sequentially visits nodes $(k_0, k_1, \dots, k_{t-1}, k_t)$. Assuming that a uniform a priori probability is used to choose the start and end nodes, the **reference probability of the path** $\tilde{P}(\wp)$ can be defined as $\tilde{P}(\wp) \triangleq \tilde{\pi}(\wp) / \sum_{\wp' \in \mathcal{P}} \tilde{\pi}(\wp')$, which ensures that the reference probability is properly normalized [105].

The problem presented in Eq. 3.1 can be solved by minimizing the following Lagrange function integrating equality constraints

$$\mathcal{L} = \sum_{\wp \in \mathcal{P}} P(\wp) \tilde{c}(\wp) + \lambda \left[\sum_{\wp \in \mathcal{P}} P(\wp) \log \frac{P(\wp)}{\tilde{P}(\wp)} - J_0 \right] + \mu \left[\sum_{\wp \in \mathcal{P}} P(\wp) - 1 \right] \quad (3.2)$$

over the set of path probabilities $\{P(\wp)\}_{\wp \in \mathcal{P}}$ [105, 190]. Solving this minimization (see [103] for details) leads to a **Gibbs-Boltzmann probability distribution**,

$$P(\wp) = \frac{\tilde{P}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\sum_{\wp' \in \mathcal{P}} \tilde{P}(\wp') \exp[-\theta \tilde{c}(\wp')]} = \frac{\tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\sum_{\wp' \in \mathcal{P}} \tilde{\pi}(\wp') \exp[-\theta \tilde{c}(\wp')]} \quad (3.3)$$

where the Lagrange parameter λ plays the role of a temperature T and $\theta = 1/T$ is the inverse temperature directly related to the relative entropy J_0 .

Thus, as expected, short paths, or low cost paths, are favored in that they have a large probability of being sampled. The inverse temperature parameter θ allows to monitor the balance between *exploration* and *exploitation*. Indeed, as it can be observed from Eq. 3.3, when $\theta \rightarrow 0^+$, the path probabilities are reduced to the probabilities generated by a natural random walk on G (pure exploration). On the other hand, when $\theta \rightarrow \infty$, the probability distribution is biased, and only the lowest-cost paths are considered (pure exploitation).

Notice that, in the sequel, it is more convenient to provide the value of the parameter θ , with $\theta > 0$, instead of the relative entropy J_0 . Therefore, it is assumed that θ is the input to the algorithm, together with the adjacency and the cost matrices.

Finally, **the bag-of-paths probability** of drawing a path starting in node $s = i$ and ending in some other node $e = j$ can now be defined as

$$P(s = i, e = j) = \frac{\sum_{\varphi \in \mathcal{P}_{ij}} \tilde{P}(\varphi) \exp[-\theta \tilde{c}(\varphi)]}{\sum_{\varphi' \in \mathcal{P}} \tilde{P}(\varphi') \exp[-\theta \tilde{c}(\varphi')]} = \frac{\sum_{\varphi \in \mathcal{P}_{ij}} \tilde{\pi}(\varphi) \exp[-\theta \tilde{c}(\varphi)]}{\sum_{\varphi' \in \mathcal{P}} \tilde{\pi}(\varphi') \exp[-\theta \tilde{c}(\varphi')]} \quad (3.4)$$

where $\mathcal{P}_{ij}$ is the set of all path starting at node i and ending at node j .

As it is shown in [105], this quantity can easily be computed in analytic closed-form beginning with the introduction of a new transition probability matrix $\mathbf{W}$ defined as

$$\mathbf{W} \triangleq \mathbf{P} \circ \exp[-\theta \mathbf{C}] \quad (3.5)$$

where $\mathbf{C}$ is the cost matrix (see Subsection 2.2.2), $\mathbf{P}$ is the transition probability matrix (see Subsection 2.2.4), $\circ$ is the elementwise (Hadamard) matrix product and the exponential function is taken elementwise. Therefore, the elements w_{ij} of $\mathbf{W}$ are $w_{ij} = p_{ij} \exp[-\theta c_{ij}]$.

Thanks to this matrix $\mathbf{W}$, François et al. [105] showed that the sum in the numerator of Eq. 3.4 can be rewritten as

$$\sum_{\varphi \in \mathcal{P}_{ij}} \tilde{\pi}(\varphi) \exp[-\theta \tilde{c}(\varphi)] = \sum_{t=0}^{\infty} [\mathbf{W}^t]_{ij} = \left[\sum_{t=0}^{\infty} \mathbf{W}^t \right]_{ij} \quad (3.6)$$

where, by convention, zero-length paths, i.e. at time step 0, are allowed and associated with a unit probability and a zero cost. Computing the convergence of the power series of $\mathbf{W}$ leads to the definition of the matrix $\mathbf{Z}$ called, by analogy to Markov chains [149], the **fundamental matrix** which is

$$\mathbf{Z} \triangleq \sum_{t=0}^{\infty} \mathbf{W}^t = (\mathbf{I} - \mathbf{W})^{-1} \quad (3.7)$$

However, the convergence of this power series is only ensured if the matrix $\mathbf{W}$ has a spectral radius less than 1, i.e. $\rho(\mathbf{W}) < 1$. As the elements of the matrix $\mathbf{W}$ are nonnegative and G is strongly connected (see Subsection 2.1.2), a sufficient condition for $\rho(\mathbf{W}) < 1$ is that $\mathbf{W}$ is sub-stochastic (all its row sums are lesser than or equal to 1 and there is at least one row sum strictly lesser than 1) [193]. As it can be observed from Eq. 3.5, this condition is always achieved for $\theta > 0$ and at least one $c_{ij} > 0$ when $a_{ij} > 0$, which is assumed for now.

Hence, following Eqs. 3.6 and 3.7, the numerator of Eq. 3.4 can finally be redefined as

$$\sum_{\wp \in \mathcal{P}_{ij}} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)] = [(\mathbf{I} - \mathbf{W})^{-1}]_{ij} = z_{ij} \quad (3.8)$$

where z_{ij} is the element i, j of the fundamental matrix $\mathbf{Z}$ which is closely related to the potential of a Markov chain [60, 212]. Moreover, the elements of the matrix $\mathbf{Z}$ can be interpreted as the expected number of time that a random walker, starting at node i , visits node j (see for instance [86, 149]) before being killed during a killed random walk [105, 190, 238]. A killed random walk is a random walk where the walker has a non-zero probability of reaching a cemetery node, or being killed, when visiting a killing node [103].

In the same way, the denominator of Eq. 3.4 can be redefined as

$$\sum_{\wp \in \mathcal{P}} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)] = \sum_{i,j=1}^n \sum_{\wp \in \mathcal{P}_{ij}} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)] = \sum_{i,j=1}^n z_{ij} \triangleq \mathcal{Z} \quad (3.9)$$

where $\mathcal{Z}$ is the **partition function** of the regular bag-of-paths system.

Finally, the **bag-of-paths probability** of drawing a path starting in node $s = i$ and ending in some other node $e = j$, presented in Eq. 3.4, is

$$P(s = i, e = j) = \frac{z_{ij}}{\mathcal{Z}} = \frac{z_{ij}}{z_{\bullet\bullet}} \quad (3.10)$$

or, in matrix form,

$$\mathbf{\Pi} = \frac{\mathbf{Z}}{z_{\bullet\bullet}} \quad (3.11)$$

where $\mathbf{\Pi}$ is the **bag-of-paths probability matrix** containing the probabilities for each start-end pair of nodes. Notice that this matrix is not symmetric in general. Therefore, in the case of an undirected graph (see Subsection 2.1.1), a variant consists of computing the probability of drawing a path $i \rightsquigarrow j$ or $j \rightsquigarrow i$, i.e. regardless of the direction of the link. The result is a symmetric matrix $\mathbf{\Pi}^{\text{sym}}$ [105],

$$\mathbf{\Pi}^{\text{sym}} = \mathbf{\Pi} + \mathbf{\Pi}^T \quad (3.12)$$

where only the upper (or lower as the matrix is symmetric) triangular part is relevant.

3.2 Bag-of-hitting-paths model

The bag-of-hitting-paths model is a restricted version of the bag-of-paths model where the set of paths is limited to paths in which the end node does not appear more than once. In other words, a **hitting path** $\wp^h$, denoted by the superscript h , is a path where none of the intermediate nodes is allowed to be the end node, thus prohibiting looping on this end node. Similarly, as in the case of an absorbing Markov chain [86, 149, 212], this constraint on the path is technically enforced by making the end node absorbing.

Following the same reasoning as in the bag-of-paths model, the **set of hitting paths** starting from node i and ending at node j is denoted as $\mathcal{P}_{ij}^h = \{\wp_{ij}^h\}$ and the **set of all hitting paths** is defined as $\mathcal{P}^h = \bigcup_{i,j=1}^n \mathcal{P}_{ij}^h$. Moreover, the **bag-of-hitting-paths probability** of drawing a hitting path starting in i and ending in j , is derived from the Eq. 3.4 by adopting a Gibbs-Boltzmann distribution on $\mathcal{P}^h$ rather than on $\mathcal{P}$, which gives

$$P_h(s = i, e = j) = \frac{\sum_{\wp \in \mathcal{P}_{ij}^h} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\sum_{\wp' \in \mathcal{P}^h} \tilde{\pi}(\wp') \exp[-\theta \tilde{c}(\wp')]} = \frac{\sum_{\wp \in \mathcal{P}_{ij}^h} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\mathcal{Z}_h} \quad (3.13)$$

where $\mathcal{Z}_h$ is also called the **partition function** but this time of the hitting bag-of-paths system [105].

As explained in [105], each non-hitting path $\wp_{ij} \in \mathcal{P}_{ij}$ can be split uniquely into two sub-paths based on the end node j . The first path is the path before the end hitting node j , $\wp_{ij}^h \in \mathcal{P}_{ij}^h$, and the second path is the path after this node j , $\wp_{jj} \in \mathcal{P}_{jj}$. As the concatenation of these two sub-paths $\wp_{ij}^h \circ \wp_{jj} \in \mathcal{P}_{ij}$ is a valid path, they can be chosen independently. Therefore, for any path $\wp_{ij} = \wp_{ij}^h \circ \wp_{jj}$, the cost $\tilde{c}(\wp_{ij})$ and the likelihood $\tilde{\pi}(\wp_{ij})$ of the path $\wp_{ij}$ can be respectively defined as $\tilde{c}(\wp_{ij}) = \tilde{c}(\wp_{ij}^h) + \tilde{c}(\wp_{jj})$ and $\tilde{\pi}(\wp_{ij}) = \tilde{\pi}(\wp_{ij}^h) \tilde{\pi}(\wp_{jj})$. Now, considering this property, the Eq. 3.8 can be reformulated as

$$\begin{aligned} z_{ij} &= \sum_{\wp_{ij} \in \mathcal{P}_{ij}} \tilde{\pi}(\wp_{ij}) \exp[-\theta \tilde{c}(\wp)] \\ &= \sum_{\wp_{ij}^h \in \mathcal{P}_{ij}^h} \sum_{\wp_{jj} \in \mathcal{P}_{jj}} \tilde{\pi}(\wp_{ij}^h) \tilde{\pi}(\wp_{jj}) \exp[-\theta \tilde{c}(\wp_{ij}^h)] \exp[-\theta \tilde{c}(\wp_{jj})] \\ &= \left(\sum_{\wp_{ij}^h \in \mathcal{P}_{ij}^h} \tilde{\pi}(\wp_{ij}^h) \exp[-\theta \tilde{c}(\wp_{ij}^h)] \right) \left(\sum_{\wp_{jj} \in \mathcal{P}_{jj}} \tilde{\pi}(\wp_{jj}) \exp[-\theta \tilde{c}(\wp_{jj})] \right) \\ &= z_{ij}^h z_{jj} \end{aligned} \quad (3.14)$$

Therefore, the elements z_{ij}^h of **the fundamental matrix of the bag-of-hitting-paths** $\mathbf{Z}_h$ are defined as

$$z_{ij}^h = [\mathbf{Z}_h]_{ij} = \frac{z_{ij}}{z_{jj}} = \sum_{\wp \in \mathcal{P}_{ij}^h} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)] \quad (3.15)$$

where the diagonal elements z_{ii}^h are equal to 1. Moreover, the elements z_{ij}^h can be interpreted as the probability of surviving during the killed random walk from i to j with transition probabilities w_{ij} and where the node j is made absorbing. In other words, it is the probability of reaching absorbing node j without being killed during the walk [105].

Now, from the Eq. 3.15, the partition function $\mathcal{Z}_h$ can be defined as

$$\mathcal{Z}_h \triangleq \sum_{i,j=1}^n z_{ij}^h = \sum_{i,j=1}^n \frac{z_{ij}}{z_{jj}} \quad (3.16)$$

Finally, **the bag-of-hitting-paths probability** of drawing a path starting in node $s = i$ and ending in some other node $e = j$, presented in Eq. 3.13, is

$$P_h(s = i, e = j) = \frac{z_{ij}/z_{jj}}{\sum_{i',j'=1}^n z_{i'j'}^h} = \frac{z_{ij}^h}{z_{\bullet\bullet}^h} = \frac{z_{ij}^h}{\mathcal{Z}_h} \quad (3.17)$$

or, in matrix form,

$$\mathbf{\Pi}_h = \frac{\mathbf{Z}(\text{Diag}(\mathbf{Z}))^{-1}}{\mathbf{e}^T \mathbf{Z}(\text{Diag}(\mathbf{Z}))^{-1} \mathbf{e}} = \frac{\mathbf{Z}_h}{\mathbf{e}^T \mathbf{Z}_h \mathbf{e}} = \frac{\mathbf{Z}_h}{\mathcal{Z}_h} \quad (3.18)$$

where $\mathbf{\Pi}_h$ is the **bag-of-hitting-paths probability matrix** containing the probabilities for each start-end pair of nodes. As in the previous model (see Eq. 3.12), a symmetric version can be derived as

$$\mathbf{\Pi}_h^{\text{sym}} = \mathbf{\Pi}_h + \mathbf{\Pi}_h^T \quad (3.19)$$

3.3 Randomized shortest path model

The randomized shortest path (RSP) model [238] is a model coming from transportation sciences [4], which was further developed by exploiting basic concepts from statistical physics [105, 156, 238, 300]. This model also has some links with the works of [271, 272] in reinforcement learning and [78] in defining random walks on a graph with maximal entropy or minimal free energy. We can also point out that the RSP model was defined from a flow approach in [23, 121] (see Appendix A for more details).

Moreover, the RSP model is closely related to the bag-of-hitting-paths model presented in the previous section. The main difference between these

two models is that the RSP model is defined on the basis of the set of all hitting paths $\mathcal{P}_{st}^h$ from a source node s to an absorbing target node t and not on the basis of the set of all hitting paths $\mathcal{P}^h$ as the bag-of-hitting-paths model.

The main idea behind the RSP model is the following. Let us consider the set of all hitting paths, or walks, $\wp \in \mathcal{P}_{st}^h$ from a node $s \in \mathcal{V}$ (source) to an absorbing, killing, node $t \in \mathcal{V}$ (target) on G . Then, we assign a probability distribution $P(\cdot)$ on the discrete set of paths $\mathcal{P}_{st}^h$ by minimizing the **free energy** [136, 156, 219, 232],

$$\begin{aligned} & \left| \begin{array}{ll} \text{minimize} & \phi(P) = \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right) \\ \text{subject to} & \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) = 1 \end{array} \right. \end{aligned} \quad (3.20)$$

where $\tilde{c}(\wp)$ is the total cost of path $\wp$ when visiting the nodes in the sequential order and $\tilde{\pi}(\wp)$ is the likelihood of the path $\wp$ (see Section 3.1 for further details). Note that the RSP model could also be derived the same way as the BoP (see Section 3.1) [103].

The objective function in Eq. 3.20 is a mixture of two dissimilarity terms, with the temperature $T > 0$ balancing the trade-off between them. The first term is the expected cost to reach target node from source node (favoring shorter paths – *exploitation*). The second term corresponds to the relative entropy, or Kullback-Leibler divergence, between the path probability distribution and the reference (random walk) paths probability distribution (introducing randomness and diversity [223] – *exploration*). For a low temperature T , low-cost paths are favored whereas when T is large, paths are chosen according to their likelihood in the reference random walk on G .

Similarly to the two previous sections, solving the minimization problem (see [238, 300] for details) presented in Eq. 3.20 leads to a Gibbs-Boltzmann distribution on the set of paths $\mathcal{P}_{st}^h$,

$$P^{\text{RSP}}(\wp) = \frac{\tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\sum_{\wp' \in \mathcal{P}_{st}^h} \tilde{\pi}(\wp') \exp[-\theta \tilde{c}(\wp')]} = \frac{\tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\mathcal{Z}_h^{(t)}} \quad (3.21)$$

where $\mathcal{Z}_h^{(t)}$ is also called the **partition function** of the system.

As already discussed in the previous sections, it turns out that the partition function can be computed in closed-form from the auxiliary matrix¹ $\mathbf{W}^{(t)}$ (see Eq. 3.5). First, the **fundamental matrix** $\mathbf{Z}_h^{(t)}$ of the RSP system is

¹The target node t is made killing and absorbing by setting the corresponding row of the reference transition matrix $\mathbf{P}$ (see Subsection 2.2.4) to zero, which implies that row t of $\mathbf{W}^{(t)}$ is also zero.

defined, in the same way as in the bag-of-paths model (see Eq. 3.7), as

$$\mathbf{Z}_h^{(t)} = \mathbf{I} + \mathbf{W}^{(t)} + \mathbf{W}^{(t)^2} + \dots = (\mathbf{I} - \mathbf{W}^{(t)})^{-1} \quad (3.22)$$

The equation sums up contributions of different paths lengths, starting from zero-length paths (identity matrix $\mathbf{I}$). Moreover, it can also be shown that Eq. 3.21 can be rewritten in terms of the elements of $\mathbf{W}^{(t)}$ as

$$p^{\text{RSP}}(\wp) = \frac{\tilde{w}^{(t)}(\wp)}{\sum_{\wp' \in \mathcal{P}_{st}^h} \tilde{w}^{(t)}(\wp')} \quad (3.23)$$

with $\tilde{w}^{(t)}(\wp)$ that is the product of the corresponding elements of the matrix $\mathbf{W}^{(t)}$ along the edges of the path $\wp$. More precisely, $\tilde{w}^{(t)}(\wp)$ is defined as

$$\begin{aligned} \tilde{w}^{(t)}(\wp) &= \prod_{\tau=1}^{\ell} w_{\wp(\tau-1)\wp(\tau)}^{(t)} \\ &= \prod_{\tau=1}^{\ell} p_{\wp(\tau-1)\wp(\tau)} \exp[-\theta c_{\wp(\tau-1)\wp(\tau)}] \\ &= \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)] \end{aligned} \quad (3.24)$$

where $\wp(\tau)$ is the index of the node at position τ of the path $\wp$. The partition function is then provided by

$$\mathcal{Z}_h^{(t)} = \sum_{\wp \in \mathcal{P}_{st}^h} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)] = \sum_{\wp \in \mathcal{P}_{st}^h} \tilde{w}^{(t)}(\wp) = [\mathbf{Z}_h^{(t)}]_{st} = z_{st}^{h(t)} \quad (3.25)$$

More generally [106], the following forward $z_{si}^{h(t)}$ and backward $z_{jt}^{h(t)}$ variables can also be defined,

$$z_{si}^{h(t)} = \sum_{\wp \in \mathcal{P}_{si}^h} \tilde{w}^{(t)}(\wp) \quad \text{and} \quad z_{jt}^{h(t)} = \sum_{\wp \in \mathcal{P}_{jt}^h} \tilde{w}^{(t)}(\wp) \quad (3.26)$$

where $\mathcal{P}_{si}^h$ is the set of hitting paths starting in node s and ending in node i and $\mathcal{P}_{jt}^h$ is the set of hitting paths starting in node j and ending in node t . Interestingly, the backward variables can be interpreted as probabilities of surviving during a killed random walk with transition matrix $\mathbf{W}^{(t)}$, that is, reaching hitting node t without being killed during the walk [105].

Moreover, the forward $\mathbf{z}_s^{h'(t)}$ and backward $\mathbf{z}_t^{h'(t)}$ variables can also be defined in matrix form as

$$\mathbf{z}_s^{h'(t)} = \mathbf{Z}_h^{(t)\top} \mathbf{e}_s \quad \text{and} \quad \mathbf{z}_t^{h'(t)} = \mathbf{Z}_h^{(t)} \mathbf{e}_t \quad (3.27)$$

where $\mathbf{e}_s$ and $\mathbf{e}_t$ are respectively the s th and the t th columns of the fundamental matrix $\mathbf{I}$.

3.4 Derived dissimilarity measures

This section aims to present three dissimilarities based on the bag-of-hitting-paths model which are the surprisal distance, the free energy distance and the randomized shortest path dissimilarity. These three dissimilarities are used in a graph-based semi-supervised classification task in Chapter 7.

3.4.1 Surprisal distance

The **surprisal distance** $\Delta_{ij}^{\text{SURP}}$ is a distance built on the symmetrization of the associated surprisal measure $-\log(P_h(s = i, e = j))$, quantifying the ‘surprise’ generated by the event $(s = i) \wedge (e = j)$, derived from the bag-of-hitting-paths model. This distance is defined as [105]

$$\begin{aligned} \Delta_{ij}^{\text{SURP}} &\triangleq \begin{cases} -\frac{\log(P_h(s = i, e = j)) + \log(P_h(s = j, e = i))}{2} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \\ &= \begin{cases} -\frac{\log\left(\frac{z_{ij}^h}{\mathcal{Z}_h}\right) + \log\left(\frac{z_{ji}^h}{\mathcal{Z}_h}\right)}{2} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \end{aligned} \quad (3.28)$$

where $P_h(s = i, e = j)$, $P_h(s = j, e = i)$ are computed according to Eq. 3.17 and $\mathcal{Z}_h$ is the partition function introduced in Eq. 3.16. Obviously, $\Delta_{ij}^{\text{SURP}} \geq 0$, $\Delta_{ij}^{\text{SURP}}$ is symmetric and $\Delta_{ij}^{\text{SURP}} = 0$ for all $i = j$. Furthermore, as demonstrated in Appendix C of [105], this quantity also satisfies the triangle inequality (see Eq. 2.15) making it a valid distance measure respecting all the conditions seen in Subsection 2.3.1. In short, François et al. [105] showed that the following inequality holds

$$\begin{aligned} P_h(s = i, e = k) &\geq \mathcal{Z}_h P_h(s = i, e = j) P_h(s = j, e = k) \\ \Rightarrow \frac{z_{ik}^h}{\mathcal{Z}_h} &\geq \mathcal{Z}_h \frac{z_{ij}^h}{\mathcal{Z}_h} \frac{z_{jk}^h}{\mathcal{Z}_h} \\ \Rightarrow z_{ik}^h &\geq z_{ij}^h z_{jk}^h \end{aligned} \quad (3.29)$$

Now, from Eq. 3.16, we know that $\mathcal{Z}_h \geq 1$ because $\mathcal{Z}_h$ is the sum of the elements of the matrix $\mathbf{Z}_h$ which contains 1’s on its diagonal and positive values everywhere else². Therefore, it can be left unconsidered from the inequality which gives

$$P_h(s = i, e = k) \geq P_h(s = i, e = j) P_h(s = j, e = k) \quad (3.30)$$

²Note that $\mathcal{Z}_h = 1$ when the graph is composed of a single node with a self-loop.

Finally, by applying $-\log$ on both sides of the inequality, the first line of Eq. 3.29 becomes

$$-\log(P_h(s = i, e = k)) \leq -\log(P_h(s = i, e = j)) - \log(P_h(s = j, e = k)) \quad (3.31)$$

for $i \neq j \neq k \neq i$. This inequality proves that the surprisal measure, $-\log(P_h(s = i, e = j))$, obeys the triangle inequality. Therefore, the surprisal distance obtained by symmetrization of this measures also benefits from this property.

Moreover, the **surprisal distance matrix** Δ^{SURP} , containing the surprisal distance between all pairs of nodes, is defined as

$$\Delta^{\text{SURP}} = -\frac{\log(\mathbf{\Pi}_h) + \log(\mathbf{\Pi}_h^T)}{2} \quad (3.32)$$

where the diagonal elements are set to zero by subtracting the $\text{Diag}(\Delta^{\text{SURP}})$ to Δ^{SURP} and $\mathbf{\Pi}_h$ is the bag-of-hitting-paths probability matrix from Eq. 3.18.

3.4.2 Free energy distance

The **free energy, potential, distance** [105, 156] is closely related to both previously introduced surprisal distance and a family of distributed consensus algorithms developed in [265]. This distance is built around the quantity $-\frac{1}{\theta} \log z_{ij}^h$ which is the potential of node i with respect to node j [105]. From this quantity, the **directed free energy distance** between two nodes can be defined as

$$\phi(i, j) \triangleq -\frac{1}{\theta} \log(z_{ij}^h) = -\frac{1}{\theta} \log\left(\frac{z_{ij}}{z_{jj}}\right) \quad (3.33)$$

where z_{ij} is the element i, j of the fundamental matrix $\mathbf{Z}$ (see Eq. 3.8). Furthermore, applying $-\frac{1}{\theta} \log$ of both sides of the inequality presented in Eq. 3.29 gives

$$-\frac{1}{\theta} \log(z_{ik}^h) \leq -\frac{1}{\theta} \log(z_{ij}^h) - \frac{1}{\theta} \log(z_{jk}^h) \quad (3.34)$$

which immediately implies that the quantity $\phi(i, j)$ obeys the triangle inequality (see Eq. 2.15). This result suggests to define the free energy distance Δ_{ij}^{FE} as

$$\Delta_{ij}^{\text{FE}} \triangleq \begin{cases} \frac{\phi(i, j) + \phi(j, i)}{2} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \quad (3.35)$$

which ensures that Δ_{ij}^{FE} has all the properties of a distance measure (see Subsection 2.3.1 for details). Notice that Δ_{ij}^{FE} can also be expressed based on the previously introduced surprisal distance $\Delta_{ij}^{\text{SURP}}$ as $\Delta_{ij}^{\text{FE}} = (\Delta_{ij}^{\text{SURP}} - \log \mathbf{Z}_h) / \theta$. Therefore, the two distances are equivalent up to the addition of a constant and a rescaling.

Moreover, the **free energy distance matrix** Δ^{FE} , containing the free energy distance between all pairs of nodes, is defined as

$$\Delta^{\text{FE}} = \frac{\Phi + \Phi^T}{2} \quad \text{with } \Phi = -\frac{1}{\theta} \log \mathbf{Z}_h \quad (3.36)$$

where the diagonal elements are set to zero by subtracting the $\text{Diag}(\Delta^{\text{FE}})$ to Δ^{FE} and $\mathbf{Z}_h$ is the fundamental matrix of the bag-of-hitting-paths from Eq. 3.15.

The free energy distance Δ_{ij}^{FE} has some interesting properties. One of these being that this distance is graph-geodetic meaning that $\Delta_{ik}^{\text{FE}} = \Delta_{ij}^{\text{FE}} + \Delta_{jk}^{\text{FE}}$ if and only if every path from i to k passes through j [50]. Another property is the fact that, for an undirected graph G (see Subsection 2.1.1), if the parameter θ converges to ∞ , the distance Δ_{ij}^{FE} recovers the shortest path distance. Conversely, for an undirected graph G , if θ converges to 0^+ , the distance Δ_{ij}^{FE} recovers half the commute cost distance. Notice that the commute cost distance between two nodes is proportional to the commute time distance between two nodes (see Subsection 2.3.3) [47, 156]. All the evidence and details of these properties are available in [105].

Interestingly, the directed free energy distance $\phi(s, t)$ can also be obtained from the randomized shortest path model. For that, the probability distribution $P(\cdot)$ should be replaced by the optimal distribution $P^{\text{RSP}}(\varphi)$ provided by Eq. 3.21 in the objective function (Eq. 3.20). We therefore obtain

$$\begin{aligned} \phi(s, t) &= \sum_{\varphi \in \mathcal{P}_{st}^h} P^{\text{RSP}}(\varphi) \tilde{c}(\varphi) + T \sum_{\varphi \in \mathcal{P}_{st}^h} P^{\text{RSP}}(\varphi) \log \left(\frac{P^*(\varphi)}{\tilde{\pi}(\varphi)} \right) \\ &= \sum_{\varphi \in \mathcal{P}_{st}^h} P^{\text{RSP}}(\varphi) \tilde{c}(\varphi) + T \sum_{\varphi \in \mathcal{P}_{st}^h} P^{\text{RSP}}(\varphi) \left(-\frac{1}{T} \tilde{c}(\varphi) - \log \mathcal{Z}_h^{(t)} \right) \\ &= -T \log \mathcal{Z}_h^{(t)} \end{aligned} \quad (3.37)$$

3.4.3 Randomized shortest path dissimilarity

The **randomized shortest path dissimilarity** [156, 238, 300] is a dissimilarity measure which, as the free energy distance, ranges from the shortest path distance to the commute cost distance. Initially, this dissimilarity is defined in the randomized shortest path model but can also be defined inside the bag-of-paths model.

As the two previous distances, the randomized shortest path dissimilarity is also built on a quantity. This quantity is the **randomized shortest past cost** $\langle \tilde{c} \rangle_{ij}$ between a node i and a node j which is defined as the expected cost

over all hitting paths connecting i and j [103],

$$\begin{aligned} \langle \tilde{c} \rangle_{ij} = \mathbb{E}[\tilde{c}] &= \sum_{\wp \in \mathcal{P}_{ij}^h} P^{\text{RSP}}(\wp) \tilde{c}(\wp) = \frac{\sum_{\wp \in \mathcal{P}_{ij}^h} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)] \tilde{c}(\wp)}{\sum_{\wp' \in \mathcal{P}_{ij}^h} \tilde{\pi}(\wp') \exp[-\theta \tilde{c}(\wp')]} \\ &= \frac{\mathbf{e}_i^T \mathbf{Z}(\mathbf{C} \circ \mathbf{W}) \mathbf{Z} \mathbf{e}_j}{z_{ij}} - \frac{\mathbf{e}_j^T \mathbf{Z}(\mathbf{C} \circ \mathbf{W}) \mathbf{Z} \mathbf{e}_j}{z_{jj}} \end{aligned} \quad (3.38)$$

where $P^{\text{RSP}}(\wp)$ is the probability distribution of Eq. 3.21, $\mathbf{Z}$ is the fundamental matrix of Eq. 3.7, $\mathbf{W}$ is the transition probability matrix of Eq. 3.5, $\mathbf{C}$ is the cost matrix (see Subsection 2.2.2), $\mathbf{e}_i$ is the i th column of the fundamental matrix $\mathbf{I}$ and $\circ$ is the elementwise (Hadamard) matrix product.

Now, the randomized shortest path dissimilarity Δ_{ij}^{RSP} [156, 238, 300] between node i and node j is obtained by taking the average of the expected cost to go from i to j and going back to j ,

$$\Delta_{ij}^{\text{RSP}} \triangleq \begin{cases} \frac{\langle \tilde{c} \rangle_{ij} + \langle \tilde{c} \rangle_{ji}}{2} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \quad (3.39)$$

Unfortunately, this dissimilarity measure is not a valid distance measure as it does not verify the triangle inequality (see Eq. 2.15). However, as the free energy distance, the randomized shortest path dissimilarity has the nice property of converging (up to a scaling factor) to the shortest path distance when $\theta \rightarrow \infty$ and to half the commute cost distance when $\theta \rightarrow 0^+$ [103].

Furthermore, by defining a matrix $\mathbf{S} = (\mathbf{Z}(\mathbf{C} \circ \mathbf{W})\mathbf{Z}) \div \mathbf{Z}$, where $\div$ is the elementwise (Hadamard) matrix division, the expected cost $\langle \tilde{c} \rangle_{ij}$ can be rewritten as $\langle \tilde{c} \rangle_{ij} = s_{ij} - s_{jj} = [\mathbf{S} - \mathbf{e}(\text{diag}(\mathbf{S}))^T]_{ij}$. Finally, based on this matrix $\mathbf{S}$ and Eq. 3.39, the **randomized shortest path dissimilarity matrix** Δ^{RSP} [156, 300] can be computed in closed-form as

$$\Delta^{\text{RSP}} = \frac{\mathbf{S} + \mathbf{S}^T - \mathbf{e}(\text{diag}(\mathbf{S}))^T - \text{diag}(\mathbf{S})\mathbf{e}^T}{2} \quad (3.40)$$

This concludes our presentation of the bag-of-paths framework. In the upcoming five chapters, we introduce three extensions of the bag-of-paths framework and discuss two applications.

Randomized shortest path with capacity constraints | 4

In this chapter, we propose an extension of the **randomized shortest path model** (RSP) (see Section 3.3) by considering the introduction of **capacity constraints** into the model. Capacity constraints on edge flows are very common in practice [3], and the applicability of the RSP model would certainly be increased if such constraints could be integrated.

Capacity constraints appear frequently in network flow problems, and there is a vast literature on the subject, especially in the operations research field (see, e.g., [3, 26, 85, 99, 111, 139, 163, 295] and references therein). They arise, for instance, in the standard *minimum cost flow* and *maximum flow* problems that respectively aim to find the source-target flow with minimal cost subject to some capacity constraints and to maximize the source-target flow subject to some capacity constraints. As discussed in [3], these tasks often appear in real-world problems such as scheduling of tasks [159], image segmentation [302], 3D reconstruction [177], route aggregated video streams from cloud data centers [262], transportation of goods [141] (see Subsection 5.5.1 for more details about transportation problems) and many more. In short, capacity constraints are mainly present to avoid congestion, spreading the traffic, or simply because the flow is restricted.

In this context, various algorithms to solve the standard **minimum cost flow** and **maximum flow** problems have been proposed: cycle-canceling [158], successive shortest paths [90], primal-dual [98], augmenting path [99], and push-relabel [109], to name a few (see also the above citations and references therein). The difference with our work is that, in the RSP, (1) the model is expressed in terms of full paths, and (2) a Kullback-Leibler regularization term is introduced, inducing randomized routing policies that can be computed by standard matrix operations.

In the transportation networks field that inspired the RSP [4, 82], capacity constraints were also considered in various transportation models, such as the standard traffic assignment problem [126], the deterministic static equilibrium model [94], the stochastic user equilibrium model [25], or the random utility theory [237] (see also the references in these papers).

Among them, the closest to our work is the stochastic user equilibrium model. However, to the best of our knowledge, we are not aware of such

models integrating capacity constraints based on an RSP-type full paths formalism. Therefore, because of their practical usefulness, we believe it would be a valuable contribution to introduce such constraints within the model, which is developed in Section 4.2. Indeed, we exploit the fact that, as the RSP model can be derived using maximum entropy arguments [67, 136], linear inequality constraints can be handled by Lagrangian duality (see, e.g., [137]).

The main motivation of this chapter is therefore to extend the RSP model by introducing edge flow capacity constraints in order to expand the applicability of this model to a wider variety of problems such as maximum flow and minimum cost flow problems. In short, the main contributions of this chapter are:

- ▶ The introduction of a new algorithm constraining the **flow capacity on edges** in the randomized shortest path model;
- ▶ The introduction of a new algorithm constraining the **net flow capacity on edges** in the randomized shortest path model;
- ▶ The application of the flow capacity algorithm to solve maximum flow problems.

The content of this chapter is structured as follows. Section 4.1 introduces some necessary notation and quantities of interest needed to derive the algorithms. Then, Section 4.2 develops new algorithms for constraining the flow capacity on edges, while Section 4.3 deals with net flow capacity constraints. Illustrative examples and scalability experiment are presented in Section 4.4. Finally, Section 4.5 is the conclusion.

4.1 Basic notions

As already discussed, the main contributions of this chapter are based on the RSP model introduced in Section 3.3. This section aims to recall some necessary notation and introduce some quantities of interest obtained from the RSP model which are useful to derive the algorithms presented in Section 4.2 and Section 4.3.

4.1.1 Background and notation

First, notice that column vectors are written in bold lowercase and matrices are in bold uppercase. Moreover, in this chapter, we consider a weighted directed,¹ strongly connected graph or network without self-loops (see Section

¹When the graph is undirected, we consider that it is made of two reciprocal, directed, edges with the same affinities and costs.

2.1), $G = (\mathcal{V}, \mathcal{E})$, with a set $\mathcal{V}$ of n nodes and a set $\mathcal{E}$ of m directed edges (see Section 2.1 for more details).

From the adjacency matrix $\mathbf{A}$ (see Subsection 2.2.1) associated with this graph, the standard **reference random walk** on the graph is usually defined as the **transition probabilities matrix** $\mathbf{P} = (p_{ij})$ (see Eq. 2.6 and Subsection 2.2.4 for details). In addition, we also assume that there is a nonnegative cost $c_{ij} = 1/a_{ij}$ associated with each edge (i, j) of the graph with $\mathbf{C} = (c_{ij})$ being the **cost matrix** (see Subsection 2.2.2).

Finally, as a reminder of Section 3.1, a **path** or **walk** φ is a finite sequence of hops to adjacent nodes on G (including cycles), initiated from a source node s and stopping at some ending target node t with $s \neq t$. A **hitting path** is a path where the last node t does not appear as an intermediate node. In other words, a hitting path to node t stops when it reaches t for the first time. In practice, we consider hitting paths to the fixed target node t by setting this target node as absorbing (or ‘killing’). Computationally, this is achieved by putting the corresponding row t of the transition matrix to zero. The node along path φ at position τ on the path is denoted by $\varphi(\tau)$. The **total cost** of a path, $\tilde{c}(\varphi)$, is simply the sum of the edge costs c_{ij} along φ while its **length** $\ell(\varphi)$ is the number of steps, or hops, needed to follow that path.

4.1.2 Quantities of interest

The first quantity of interest that we are going to investigate is the expected **directed flow** in edge (i, j) , $\bar{n}_{ij}$, that is the expected number of passages through (i, j) when going from a source node s to a target node t . Intuitively, $\bar{n}_{ij}$ can be defined as

$$\bar{n}_{ij} \triangleq \sum_{\varphi \in \mathcal{P}_{st}^h} \mathbf{P}(\varphi) \eta((i, j) \in \varphi) \quad (4.1)$$

where $\eta((i, j) \in \varphi)$ counts the number of times edge (i, j) appears on path φ . As developed in [155, 238], the directed flow $\bar{n}_{ij}$ can be obtained from Eq. 3.37 and Eq. 3.22 for a given inverse temperature $\theta = 1/T$ by

$$\begin{aligned} \bar{n}_{ij} &= \frac{\sum_{\varphi \in \mathcal{P}_{st}^h} \tilde{\pi}(\varphi) \exp[-\theta \tilde{c}(\varphi)] \eta((i, j) \in \varphi)}{\sum_{\varphi' \in \mathcal{P}_{st}^h} \tilde{\pi}(\varphi') \exp[-\theta \tilde{c}(\varphi')]} \\ &= \frac{\sum_{\varphi \in \mathcal{P}_{st}^h} \tilde{\pi}(\varphi) \exp[-\theta \tilde{c}(\varphi)] \frac{\partial \tilde{c}(\varphi)}{\partial c_{ij}}}{\sum_{\varphi' \in \mathcal{P}_{st}^h} \tilde{\pi}(\varphi') \exp[-\theta \tilde{c}(\varphi')]} = -\frac{1}{\theta} \frac{\partial \log \mathcal{Z}_h^{(t)}}{\partial c_{ij}} \end{aligned}$$

$$\begin{aligned}
&= -\frac{1}{\theta} \frac{\partial \log(z_{st}^{h(t)}/z_{tt}^{h(t)})}{\partial c_{ij}} = \frac{1}{\theta} \left(\frac{1}{z_{st}^{h(t)}} \frac{\partial z_{st}^{h(t)}}{\partial c_{ij}} - \frac{1}{z_{tt}^{h(t)}} \frac{\partial z_{tt}^{h(t)}}{\partial c_{ij}} \right) \\
&= -\frac{1}{\theta} \left(\frac{-\theta w_{ij}^{(t)} z_{si}^{h(t)} z_{jt}^{h(t)}}{z_{st}^{h(t)}} + \frac{\theta w_{ij}^{(t)} z_{ti}^{h(t)} z_{jt}^{h(t)}}{z_{tt}^{h(t)}} \right) \\
&= \left(\frac{z_{si}^{h(t)}}{z_{st}^{h(t)}} - \frac{z_{ti}^{h(t)}}{z_{tt}^{h(t)}} \right) w_{ij}^{(t)} z_{jt}^{h(t)} = \frac{z_{si}^{h(t)} w_{ij}^{(t)} z_{jt}^{h(t)}}{z_{st}^{h(t)}} \quad (4.2)
\end{aligned}$$

where $w_{ij}^{(t)}$ is the element i, j of the auxiliary matrix (see Eq. 3.5 and Section 3.3), $z_{st}^{h(t)}$ is the element s, t of the fundamental matrix $\mathbf{Z}_h^{(t)}$ (see Eq. 3.22), and $\partial z_{st}^{h(t)}/\partial c_{ij} = -\theta w_{ij}^{(t)} z_{si}^{h(t)} z_{jt}^{h(t)}$ according to Eq. 11 of [155].

Now, as only the row s and the column t of the fundamental matrix $\mathbf{Z}_h^{(t)}$ are needed, two systems of linear equations can be solved instead of the matrix inversion in Eq. 3.22. Note also that it can be readily shown from Eq. 4.2 that flows are conserved on intermediate nodes $\mathcal{V} \setminus \{s, t\}$; indeed, $\sum_{i=1}^n (\bar{n}_{ij} - \bar{n}_{ji}) = 0$ (input flow to j is equal to output flow from j). Let us further define the matrix containing the expected number of passages through edges (i, j) by $\bar{\mathbf{N}} = (\bar{n}_{ij})$.

From the last Eq. 4.2, the second quantity of interest which is the expected **number of visits** to a node j can also be computed as

$$\begin{aligned}
\bar{n}_j &= \sum_{i \in \text{Pred}(j)} \bar{n}_{ij} + \delta_{sj} = \sum_{i=1}^n \bar{n}_{ij} + \delta_{sj} \\
&= \sum_{i=1}^n \left(\frac{z_{si}^{h(t)} w_{ij}^{(t)} z_{jt}^{h(t)}}{z_{st}^{h(t)}} \right) + \delta_{sj} = \sum_{i=1}^n (z_{si}^{h(t)} w_{ij}^{(t)}) \frac{z_{jt}^{h(t)}}{z_{st}^{h(t)}} + \delta_{sj} \\
&= \frac{z_{sj}^{h(t)} z_{jt}^{h(t)}}{z_{st}^{h(t)}} - \frac{\delta_{sj} z_{jt}^{h(t)}}{z_{st}^{h(t)}} + \delta_{sj} = \frac{z_{sj}^{h(t)} z_{jt}^{h(t)}}{z_{st}^{h(t)}} \quad (4.3)
\end{aligned}$$

because a unit flow of $+1$ is injected in node s . Note that $\text{Pred}(j)$ is the set of predecessor nodes of node j and that we used $\sum_{i=1}^n z_{si}^{h(t)} w_{ij}^{(t)} = z_{sj}^{h(t)} - \delta_{sj}$, coming from $\mathbf{Z}_h^{(t)}(\mathbf{I} - \mathbf{W}^{(t)}) = \mathbf{I}$ (see Eq. 3.22), with δ_{sj} being a Kronecker delta that is 1 when $s = j$ and 0 otherwise.

From these quantities of interest, the **optimal transition probabilities** of following any edge $(i, j) \in \mathcal{E}$ (the policy) induced by the set of paths $\mathcal{P}_{st}^h$ and their probability mass (Eq. 3.21) can be defined as [238]

$$p_{ij}^* = \frac{\bar{n}_{ij}}{\sum_{j'=1}^n \bar{n}_{ij'}} = \frac{\bar{n}_{ij}}{\bar{n}_i} = \frac{z_{jt}^{h(t)}}{z_{it}^{h(t)}} w_{ij}^{(t)} \quad \text{for all } i \neq t \quad (4.4)$$

and $p_{tj}^* = 0$ for the target node and all j . These transition probabilities define a **biased random walk** (an absorbing Markov chain) on G – the random walker is ‘attracted’ by the target node t . The lower the temperature, the larger the attraction. Interestingly, these transition probabilities do not depend on the source node, and they correspond to the optimal randomized strategy, or policy, thereby minimizing free energy (Eq. 3.20) for reaching the target node. Notice further that $\bar{n}_{ij} = \bar{n}_i p_{ij}^*$.

Furthermore, the **expected cost** until absorption by target node t at temperature T can easily be computed in closed-form from the RSP formalism based on the direct flow in edge (i, j) , $\bar{n}_{ij}$. This expected cost spread in the network has been used as a dissimilarity measure between nodes [156, 238, 300] and was called the directed **RSP dissimilarity** (see Subsection 3.4.3). More formally, as already shown in Eq. 3.38, the expected cost spread in the network [106] is given by

$$\langle \tilde{c} \rangle = \sum_{\varphi \in \mathcal{P}_{st}^h} P(\varphi) \tilde{c}(\varphi) \quad (4.5)$$

Let us now express the cost along path φ as $\tilde{c}(\varphi) = \sum_{(i,j) \in \mathcal{E}} \eta((i,j) \in \varphi) c_{ij}$ where we saw that $\eta((i,j) \in \varphi)$ is the number of times edge (i,j) appears on path φ and $\mathcal{E}$ is the set of edges. Injecting this last result in Eq. 4.5 provides

$$\langle \tilde{c} \rangle = \sum_{(i,j) \in \mathcal{E}} \underbrace{\left(\sum_{\varphi \in \mathcal{P}_{st}^h} P(\varphi) \eta((i,j) \in \varphi) \right)}_{\text{expected number of passages } \bar{n}_{ij}} c_{ij} = \sum_{(i,j) \in \mathcal{E}} \bar{n}_{ij} c_{ij} \quad (4.6)$$

or, in matrix form,

$$\langle \tilde{c} \rangle = \mathbf{e}^T (\bar{\mathbf{N}} \circ \mathbf{C}) \mathbf{e} \quad (4.7)$$

where $\circ$ is the elementwise matrix product and $\bar{\mathbf{N}}$ is the matrix of expected number of passages through edges defined in Eq. 4.2. This quantity is just the cumulative sum of the expected number of passages through each edge times the cost of following the edge.

4.2 Dealing with edge flow capacity constraints

In this section, an algorithm computing the optimal policy (the equivalent of Eq. 4.4) under flow capacity constraints on edges is derived. It is based on the fact that linear inequality constraints can easily be integrated into maximum entropy problems by considering Lagrangian duality (see, e.g., [137] and references therein).

For convenience, we now assume a weighted, undirected, connected graph G (see Section 2.1) with a single source node (node s) and a single target

node (node $t \neq s$). An input flow is injected into node s and absorbed by node t , but the model can be generalized for multiple sources and destinations (see Chapter 5). As before, it is assumed that the target node t is killing and absorbing, meaning that the transition probabilities $p_{tj} = 0$ for all nodes j , including node $j = t$.

The idea now is to constrain the flow visiting some edges belonging to a set of constrained edges $\mathcal{C}$. The expected number of passages through these edges (see Eq. 4.2) is therefore forced not to exceed some predefined values (upper bound),

$$\bar{n}_{ij} \leq \sigma_{ij} \quad \text{for edges } (i, j) \in \mathcal{C} \quad (4.8)$$

which ensures that the flows on edges in $\mathcal{C}$ are limited by the capacity $\sigma_{ij} > 0$ and thus must remain in the interval $[0, \sigma_{ij}]$. In this section, we consider that, although the graph G is undirected, the capacity constraints are directed and thus active in only one direction of an edge. Therefore, each undirected edge $i \leftrightarrow j$ of G possibly leads to two directed edges (i, j) and (j, i) in $\mathcal{C}$, with different capacity constraints. Thus the set of constrained edges $\mathcal{C}$ contains directed edges, limiting the directional flow through them.

Moreover, we assume that the constraints are feasible². The solution minimizes the free energy objective function (Eq. 3.20) while satisfying these inequality constraints.

4.2.1 Lagrange function in case of capacity constraints

From nonlinear optimization theory [27, 68, 119, 194], the Lagrange function (following Eq. 3.20) is

$$\begin{aligned} \mathcal{L}(\mathbf{P}, \boldsymbol{\lambda}) &= \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right) \\ &\quad + \mu \left(\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) - 1 \right) + \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} (\bar{n}_{ij} - \sigma_{ij}) \\ &= \underbrace{\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right)}_{\text{free energy, } \phi(\mathbf{P})} \\ &\quad + \mu \left(\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) - 1 \right) \end{aligned}$$

²If not, the duality gap in our algorithm does not converge to zero.

$$+ \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \underbrace{\left(\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \eta((i,j) \in \wp) - \sigma_{ij} \right)}_{\bar{n}_{ij} \text{ (see Eq. 4.2)}} \quad (4.9)$$

where there is a Lagrange parameter μ associated with the sum-to-one constraint and a Lagrange parameter λ_{ij} associated with each constrained edge. The Lagrange parameters $\{\lambda_{ij}\}$, $(i,j) \in \mathcal{C}$, are all non-negative in the case of inequality constraints [119] and are stacked into the parameter vector $\boldsymbol{\lambda}$.

Note that, by inspecting Eq. 4.9 and from the convexity of the Kullback-Leibler divergence, the primal objective function to be minimized with respect to the discrete probabilities (which is similar to Eq. 3.20) is convex, the equality constraints are all linear, and the inequality constraints form a convex set. Therefore, the duality gap between the primal and dual problems is zero, which is exploited to solve the problem (see, e.g., [27, 68, 119, 194]).

Now, the Lagrange function in Eq. 4.9 can be rearranged as

$$\begin{aligned} \mathcal{L}(\mathbf{P}, \boldsymbol{\lambda}) &= \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \underbrace{\left(\tilde{c}(\wp) + \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \eta((i,j) \in \wp) \right)}_{\text{augmented cost } \tilde{c}'(\wp) \text{ cumulated on path } \wp} \\ &\quad + T \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right) \\ &\quad + \mu \left(\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) - 1 \right) - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij} \\ &= \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \sum_{(i,j) \in \mathcal{C}} \eta((i,j) \in \wp) \underbrace{(c_{ij} + \delta((i,j) \in \mathcal{C}) \lambda_{ij})}_{\text{augmented costs } c'_{ij}} \\ &\quad + T \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right) \\ &\quad + \mu \left(\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) - 1 \right) - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij} \\ &= \underbrace{\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \tilde{c}'(\wp) + T \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right)}_{\text{free energy based on augmented costs, } \phi'(\mathbf{P})} \\ &\quad + \mu \left(\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) - 1 \right) - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij} \end{aligned} \quad (4.10)$$

where the symbol $\delta((i,j) \in \mathcal{C})$ is defined as 1 when edge $(i,j) \in \mathcal{C}$ and

0 otherwise. Note that we used $\tilde{c}(\wp) = \sum_{(i,j) \in \mathcal{E}} \eta((i,j) \in \wp) c_{ij}$ (Eq. 4.6) computing the total cost along path $\wp$.

During this derivation, we observe that the costs c_{ij} can be redefined into **augmented costs** that integrate the additional ‘virtual’ costs (the Lagrange parameters) needed to satisfy the constraints,

$$c'_{ij} = \begin{cases} c_{ij} + \lambda_{ij} & \text{when edge } (i,j) \in \mathcal{C} \\ c_{ij} & \text{otherwise} \end{cases} \quad (4.11)$$

where $\mathbf{C}' = (c'_{ij})$ is the matrix containing these augmented costs. Thus, the Lagrange parameters have an interpretation similar to the dual variables in linear programming; they represent the extra cost to pay, associated with each edge in order to satisfy the constraints [27, 68, 119, 194, 279]. This is also common in many network flow problems [3, 85].

Let $\phi'(\mathbf{P})$ be the free energy obtained in Eq. 4.10 from these augmented costs (Eq. 4.11). We now turn to the problem of finding the Lagrange parameters λ_{ij} by exploiting Lagrangian duality.

4.2.2 Exploiting Lagrangian duality

In this subsection, we take advantage of the fact that, in the formulation of the problem (close to maximum entropy arguments), the Lagrange dual function and its gradient are relatively easy to compute; see, for instance, [137] for similar arguments in the context of supervised classification.

Indeed, as the objective function is strictly convex, the equality constraints are linear and the support set for the path probabilities is convex, it is known that, provided that the problem is feasible³, there is only one global minimum and the duality gap is zero [27, 36, 68, 119, 194]. The optimum is a saddle point of the Lagrange function, and a common optimization procedure (sometimes called the Arrow-Hurwicz-Uzawa algorithm [11, 194]) consists of sequentially (i) solving the primal (finding the optimal probability distribution) while considering the Lagrange parameters as fixed, and then (ii) maximizing the dual (which is concave) with respect to the Lagrange parameters, until convergence.

In our context, this provides the following steps [119] which are iterated,

³Recall that we assume that the problem is feasible; see the discussion at the end of Subsection 4.2.3.

$$\begin{cases}
\mathcal{L}(\boldsymbol{\lambda}) = \mathcal{L}(\mathbf{P}^*(\boldsymbol{\lambda}), \boldsymbol{\lambda}) = \underset{\{\mathbf{P}(\varphi)\}_{\varphi \in \mathcal{P}_{st}^h}}{\text{minimize}} \mathcal{L}(\mathbf{P}, \boldsymbol{\lambda}) & \text{(compute the dual function)} \\
\boldsymbol{\lambda}^* = \arg \max_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) & \text{(maximize the dual function)} \\
\boldsymbol{\lambda} = \boldsymbol{\lambda}^* & \text{(update } \boldsymbol{\lambda})
\end{cases} \quad (4.12)$$

This is the procedure that is followed whereby the dual function is maximized through a simple gradient ascent procedure.

4.2.2.1 Computing the dual function

To compute the dual function $\mathcal{L}(\mathbf{P}^*(\boldsymbol{\lambda}), \boldsymbol{\lambda})$ in Eq. 4.12, we first have to find the optimal probability distribution $\mathbf{P}^*$ in terms of the Lagrange parameters. We thus have to compute the minimum of Eq. 4.10 for a constant $\boldsymbol{\lambda}$. But this Lagrange function (Eq. 4.10) is identical to the Lagrange function associated with the standard RSP optimization problem (Eq. 3.20), except that the costs c_{ij} are replaced by the augmented costs c'_{ij} , and that the introduction of the final additional term does not depend on the probability distribution. Therefore, the probability distribution $\mathbf{P}(\cdot)$ minimizing Eq. 4.10 is a Gibbs-Boltzmann distribution of the form of Eq. 3.21, but it now depends on the augmented costs instead of the original costs.

Next, we replace the probability distribution $\mathbf{P}(\cdot)$ in $\mathcal{L}(\mathbf{P}, \boldsymbol{\lambda})$ by the optimal Gibbs-Boltzmann distribution $\mathbf{P}^*(\cdot)$, which depends on the augmented costs and thus also on the Lagrange parameters. From the result of Eq. 3.37, the obtained dual function⁴ (Eq. 4.12) is

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathcal{L}(\mathbf{P}^*(\boldsymbol{\lambda}), \boldsymbol{\lambda}) = -T \log \mathcal{Z}_h'^{(t)} - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij} \quad (4.13)$$

In this equation, $\mathcal{Z}_h'^{(t)}$ is the partition function (see Eq. 3.25), computed from the augmented costs $\mathbf{C}'$, which depend on $\boldsymbol{\lambda}$. We now need to maximize this dual function with respect to these Lagrange parameters.

4.2.2.2 Maximizing the dual function

The maximization of the dual function can be done, by, for example, using the simple method developed by Rockafellar (see [234], Eqs. 10 and 12). But let us first compute the gradient of the dual function (Eq. 4.13) with respect to the non-negative λ_{ij} defined on edges $(i, j) \in \mathcal{C}$. From $-T \partial \log \mathcal{Z}_h'^{(t)} / \partial c_{ij} = \bar{n}_{ij}$

⁴We leave out the sum-to-one constraint term which does not depend on $\boldsymbol{\lambda}$.

(Eq. 4.2) and $\partial c'_{ij}/\partial \lambda_{ij} = \delta((i, j) \in \mathcal{C})$ (Eq. 4.11), this gradient is

$$\frac{\partial \mathcal{L}(\boldsymbol{\lambda})}{\partial \lambda_{ij}} = \partial \left(-T \log \mathcal{Z}'_h(t) - \sum_{(k,l) \in \mathcal{C}} \lambda_{kl} \sigma_{kl} \right) / \partial \lambda_{ij} = \bar{n}_{ij} - \sigma_{ij} \quad (4.14)$$

where $\bar{n}_{ij}$ is computed from the augmented costs. It can be observed that we simply recover the expressions for the capacity constraints; this is actually a standard result when dealing with maximum entropy problems (see, e.g., [67, 137, 143]).

To compute the Lagrange parameters, we thus follow [234] who proposed the following (gradient-based) updating rule,⁵

$$\lambda_{ij} \leftarrow \max(\lambda_{ij} + \alpha(\bar{n}_{ij} - \sigma_{ij}), 0) \text{ for all } (i, j) \in \mathcal{C} \quad (4.15)$$

which is guaranteed to converge in the concave case, as long as α is positive, not too large, and the problem is feasible [234]. Note that the gradient ascent step parameter α must be tuned and that it is very sensitive to the value of the inverse temperature parameter θ .

This expression states that, if the flow in an edge (i, j) is too large (that is, $\bar{n}_{ij} - \sigma_{ij} > 0$), the augmented cost should increase in order to reduce this flow. On the contrary, if the flow is below the capacity threshold, the augmented cost should decrease until eventually reaching the normal cost value c_{ij} (when $\lambda_{ij} = 0$). Notice also that, because costs are multiplied by θ in the model (see Eq. 3.22), this model becomes insensitive to cost variations when θ is small (close to 0, an almost random walk behavior).

Of course, we could use other, more sophisticated, and more efficient, optimization techniques (see, e.g., [27, 36, 119, 211]), but this simple procedure worked satisfactorily for our tests on small- to medium-sized graphs. The parameter α has to be tuned manually for each different dataset, but this was not an issue. However, in its present form, the algorithm does not scale to larger graphs.

4.2.3 Resulting algorithm and discussion

The resulting algorithm is presented in Algorithm 1⁶. It computes the optimal policy (Eq. 4.4), minimizing the objective function (Eq. 3.20) while satisfying the inequality constraints (Eq. 4.8). This optimal policy guides the random walker to the target state with a trade-off between exploitation and exploration that is monitored by the inverse temperature parameter $\theta = 1/T$. The different steps of the procedure are as follows:

⁵Note that [234] uses 2α for the error correction term (instead of α here).

⁶In this pseudocode, $\mathbf{e}_i$ is a column (basis) vector of 0s except on row i where it contains a 1.

1. Initialize the Lagrange parameters to 0 and the augmented costs to the original edge costs $\mathbf{C}$.
2. Iterate the following steps until convergence:
 - The required elements of the fundamental matrix are recomputed from the current augmented costs (Eq. 3.22) by solving two systems of linear equations.
 - The expected number of passages through each edge (edge flows) is computed (Eq. 4.2).
 - The Lagrange parameters and the augmented costs are updated (Eqs. 4.15 and 4.11).
3. Compute and return the optimal policy (transition probabilities) according to Eq. 4.4.

Because two systems of n linear equations need to be solved at each iteration, its complexity is of the order $2k \cdot \mathcal{O}(n^3)$, depending on the number of iterations k needed for convergence. This number of iterations is unknown in advance but could become large depending on the problem and the gradient step. However, for sparse graphs, the complexity could be reduced by taking advantage of special numerical methods to solve sparse systems of linear equations [256].

Note that we also considered introducing lower bound constraints on the expected flows in the RSP model instead of upper bound constraints, as presented in Section 4.2. More precisely, the flow rate is constrained to be greater than or equal to a threshold value. However, the resulting virtual costs, Lagrange parameters, associated to this optimization problem can become negative in order to augment the flow in the edge. These negative costs lead to the appearance of negative cycles that cause numerical instabilities.

One solution to avoid these numerical instabilities is to prohibit negative costs by adding a constraint. Unfortunately, this solution makes the problem unfeasible in some situations. Therefore, we decided to leave the study of lower-bounded capacities for future work.

As a last remark, note that it was assumed that the problem is feasible, which can be difficult to check in practice. Indeed, the model injects a unit flow into the network so that the maximum flow through the network must be at least equal to one. This means that we cannot blindly assign capacities because, in case the problem is not feasible, the algorithm does not converge. One way to check the overall capacity of the network is to run a standard max-flow algorithm [3, 26, 85, 99, 111, 139, 163, 295].

Algorithm 1 Randomized shortest path with capacity constraints.**Input:**

- A weighted, undirected, connected graph G (see Section 2.1) containing n nodes. Node s is the source node and node t the target node.
- The $n \times n$ reference transition probabilities matrix $\mathbf{P}$ (see Subsection 2.2.4) associated with G .
- The $n \times n$ symmetric cost matrix $\mathbf{C}$ (see Subsection 2.2.2) associated with G , defining non-negative costs of transitions.
- The set of constrained edges $\mathcal{C}$ (see the text for details).
- The set of non-negative capacities on flows, $\{\sigma_{ij}\}$, defined on the set of constrained edges, $(i, j) \in \mathcal{C}$.
- The inverse temperature parameter $\theta > 0$.
- The gradient ascent step $\alpha > 0$.

Output:

- The $n \times n$ randomized policy provided by the transition matrix $\mathbf{P}^*$, defining a biased random walk on G satisfying the capacity constraints.
1. $\boldsymbol{\lambda} \leftarrow \mathbf{0}$ $\triangleright$ initialize the $|\mathcal{C}| \times 1$ Lagrange parameters vector
 2. $\mathbf{C}' \leftarrow \mathbf{C}$ $\triangleright$ initialize the augmented costs matrix
 3. Set row t of matrix $\mathbf{P}$ to $\mathbf{0}^T$ $\triangleright$ target node t is made absorbing and killing
 4. **repeat** $\triangleright$ main iteration loop
 5. $\mathbf{W}'^{(t)} \leftarrow \mathbf{P} \circ \exp[-\theta \mathbf{C}']$ $\triangleright$ update $\mathbf{W}'^{(t)}$ matrix (see Eq. 3.5)
 6. Solve $(\mathbf{I} - \mathbf{W}'^{(t)})\mathbf{z}_t^{\text{h}'(t)} = \mathbf{e}_t$ $\triangleright$ backward variables $\mathbf{z}_t^{\text{h}'(t)}$ (column t of the fundamental matrix $\mathbf{Z}_h^{\text{h}'(t)}$ with elements $z_{it}^{\text{h}'(t)}$ (see Eq. 3.27)
 7. Solve $(\mathbf{I} - \mathbf{W}'^{(t)})^T \mathbf{z}_s^{\text{h}'(t)} = \mathbf{e}_s$ $\triangleright$ forward variables $\mathbf{z}_s^{\text{h}'(t)}$ (row s of the fundamental matrix $\mathbf{Z}_h^{\text{h}'(t)}$ viewed as a column vector) with elements $z_{si}^{\text{h}'(t)}$ (see Eq. 3.27)
 8. $\bar{\mathbf{N}} \leftarrow \frac{\text{Diag}(\mathbf{z}_s^{\text{h}'(t)}) \mathbf{W}'^{(t)} \text{Diag}(\mathbf{z}_t^{\text{h}'(t)})}{z_{st}^{\text{h}'(t)}}$ $\triangleright$ compute the expected number of passages in each edge (see Eq. 4.2)
 9. **for all** $(i, j) \in \mathcal{C}$ **do** $\triangleright$ gradient ascent: update all quantities associated with constrained edges
 10. $\lambda_{ij} \leftarrow \max(\lambda_{ij} + \alpha(\bar{n}_{ij} - \sigma_{ij}), 0)$ $\triangleright$ update Lagrange parameters (see Eq. 4.15)
 11. $c'_{ij} \leftarrow c_{ij} + \lambda_{ij}$ $\triangleright$ update augmented costs (see Eq. 4.11)
 12. **end for**
 13. **until** convergence
 14. $\mathbf{P}^* \leftarrow (\text{Diag}(\mathbf{z}_t^{\text{h}'(t)}))^{-1} \mathbf{W}'^{(t)} \text{Diag}(\mathbf{z}_t^{\text{h}'(t)})$ $\triangleright$ compute optimal policy (see Eq. 4.4)
 15. **return** $\mathbf{P}^*$

4.3 Dealing with net flow capacity constraints

Let us now consider the case where the capacities $\sigma_{ij} > 0$ with $(i, j) \in \mathcal{C}$ are defined on *net flows* instead of raw flows. As before, it is assumed in this section that the original graph is weighted undirected and connected (see Section 2.1), and that the adjacency matrix as well as the cost matrix are symmetric (see Subsections 2.2.1 and 2.2.2).

In some situations, for instance electrical networks [32, 86], only net flows matter. Intuitively, it means that the edge flows in opposite directions $i \rightarrow j$

and $j \rightarrow i$ compensate each other so that only the positive net flow, defined as $|\bar{n}_{ij} - \bar{n}_{ji}|$, is taken into account, where edge flows are given by Eq. 4.2. In many situations, net flows look intuitively more natural because of the flow compensation mechanism common to electricity. Net flows have already been used in the RSP model to define node betweenness measures [155], generalizing two previous models based on electrical currents [40, 202].

As for electrical currents [32, 86], the non-negative net flow in each edge (i, j) , denoted here as $\bar{j}_{ij}$, is given in the RSP formalism by

$$\bar{j}_{ij} = \max((\bar{n}_{ij} - \bar{n}_{ji}), 0) \quad (4.16)$$

or, in matrix form,

$$\bar{\mathbf{J}} = \max((\bar{\mathbf{N}} - \bar{\mathbf{N}}^T), \mathbf{0}) \quad (4.17)$$

where the maximum is taken elementwise. This means that, for each edge, the net flow is defined (that is, positive) in only one direction⁷, and is equal to zero in the other direction.

Moreover, similarly to Eq. 4.3, the net expected number of visits to a node j can be computed as

$$\bar{j}_j = \sum_{i \in \text{Pred}(j)} \bar{j}_{ij} + \delta_{sj} = \sum_{i=1}^n \bar{j}_{ij} + \delta_{sj} \quad (4.18)$$

because a unit flow of +1 is injected in node s .

In this situation, the constraints operate on the net flows defined in Eq. 4.16 instead of the raw flows,

$$\begin{aligned} \bar{j}_{ij} &= \max((\bar{n}_{ij} - \bar{n}_{ji}), 0) \leq \sigma_{ij}, \\ \text{or equivalently both } &\begin{cases} \bar{n}_{ij} - \bar{n}_{ji} \leq \sigma_{ij} \\ \bar{n}_{ji} - \bar{n}_{ij} \leq \sigma_{ij} \end{cases} \quad \text{for each edge } (i, j) \in \mathcal{C} \end{aligned} \quad (4.19)$$

In the net flows setting, we further assume that σ_{ji} is always defined when there exists a capacity constraint σ_{ij} and that $\sigma_{ji} = \sigma_{ij}$ (symmetry). In Eq. 4.19, only one among the two flow differences is positive so that the constraint only operates in this direction of the flow: the second constraint is automatically satisfied. This also means that only one of the two constraints can become active.

To summarize, if the set of constraint nodes $\mathcal{C}$ contains edge (i, j) , it also necessarily contains its reciprocal (j, i) (they come as a pair) with the same capacity value, $\sigma_{ji} = \sigma_{ij}$. We now turn to the definition of the Lagrange function and the derivation of the algorithm.

⁷Another common convention is to consider $j_{ij} = (\bar{n}_{ij} - \bar{n}_{ji})$, and thus $\bar{j}_{ji} = -\bar{j}_{ij}$.

4.3.1 Lagrange function in case of net flow capacity constraints

We now proceed as in Subsection 4.2.1, after rearranging the terms as in Eq. 4.10 and, once again using Eq. 4.1, the Lagrange function becomes

$$\begin{aligned}
\mathcal{L}(\mathbf{P}, \boldsymbol{\lambda}) &= \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right) \\
&\quad + \mu \left(\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) - 1 \right) + \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} (\bar{n}_{ij} - \bar{n}_{ji} - \sigma_{ij}) \\
&= \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \left(\tilde{c}(\wp) + \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} (\eta((i,j) \in \wp) - \eta((j,i) \in \wp)) \right) \\
&\quad + T \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right) \\
&\quad + \mu \left(\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) - 1 \right) - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij} \tag{4.20}
\end{aligned}$$

Now, from the symmetry of edges (edges are present in pairs; for each $(i,j) \in \mathcal{C}$: $(j,i) \in \mathcal{C}$ and $\sigma_{ij} = \sigma_{ji}$), we deduce $\sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \eta((j,i) \in \wp) = \sum_{(j,i) \in \mathcal{C}} \lambda_{ij} \eta((j,i) \in \wp) = \sum_{(i,j) \in \mathcal{C}} \lambda_{ji} \eta((i,j) \in \wp)$. Injecting this result into Eq. 4.20 and proceeding in the same way as in Eq. 4.10 provides

$$\begin{aligned}
\mathcal{L}(\mathbf{P}, \boldsymbol{\lambda}) &= \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \underbrace{\left(\tilde{c}(\wp) + \sum_{(i,j) \in \mathcal{C}} (\lambda_{ij} - \lambda_{ji}) \eta((i,j) \in \wp) \right)}_{\text{augmented cost } \tilde{c}'(\wp) \text{ cumulated on path } \wp} \\
&\quad + T \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right) \\
&\quad + \mu \left(\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) - 1 \right) - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij} \\
&= \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \sum_{(i,j) \in \mathcal{E}} \eta((i,j) \in \wp) \underbrace{(c_{ij} + \delta((i,j) \in \mathcal{C}) (\lambda_{ij} - \lambda_{ji}))}_{\text{augmented costs } c'_{ij}} \\
&\quad + T \sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right) \\
&\quad + \mu \left(\sum_{\wp \in \mathcal{P}_{st}^h} P(\wp) - 1 \right) - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij}
\end{aligned}$$

$$\begin{aligned}
&= \underbrace{\sum_{\varphi \in \mathcal{P}_{st}^h} P(\varphi) \mathcal{C}'(\varphi) + T \sum_{\varphi \in \mathcal{P}_{st}^h} P(\varphi) \log \left(\frac{P(\varphi)}{\tilde{\pi}(\varphi)} \right)}_{\text{free energy based on augmented costs, } \phi'(P)} \\
&+ \mu \left(\sum_{\varphi \in \mathcal{P}_{st}^h} P(\varphi) - 1 \right) - \sum_{(i,j) \in \mathcal{C}} \lambda_{ij} \sigma_{ij}
\end{aligned} \tag{4.21}$$

which has exactly the same form as in the raw flow case (see Eq. 4.10) with the exception that the definition of the augmented costs differs in the two expressions. Indeed, as before, the costs c_{ij} can be redefined into *augmented* costs,

$$c'_{ij} = \begin{cases} c_{ij} + \lambda_{ij} - \lambda_{ji} & \text{when edge } (i, j) \in \mathcal{C} \\ c_{ij} & \text{otherwise} \end{cases} \tag{4.22}$$

We must stress the requirement that the constraints in Eq. 4.19 be symmetric and come by pair. In addition, as discussed before, only one constraint can become active among the two directions (i, j) and (j, i) , which implies that one of the two Lagrange multipliers $\{\lambda_{ij}, \lambda_{ji}\}$ must be equal to 0: the λ_{ij} for which $(\bar{n}_{ij} - \bar{n}_{ji}) < 0$.

Moreover, following the same reasoning as in the previous section (see Eqs. 4.13 and 4.15), it can immediately be observed that the dual function has the same form as before and is provided by Eq. 4.13.

4.3.2 Resulting algorithm and discussion

In the net flow context, proceeding in the same way as in the previous subsection (see Subsection 4.2.2.2), the gradient of the dual Lagrange function (Eq. 4.13) for augmented costs provided by Eq. 4.22 is

$$\begin{aligned}
d\mathcal{L}(\boldsymbol{\lambda})/d\lambda_{ij} &= (\partial\mathcal{L}(\boldsymbol{\lambda})/\partial c'_{ij})(\partial c'_{ij}/\partial \lambda_{ij}) + (\partial\mathcal{L}(\boldsymbol{\lambda})/\partial c'_{ji})(\partial c'_{ji}/\partial \lambda_{ij}) \\
&+ \partial\mathcal{L}(\boldsymbol{\lambda})/\partial \lambda_{ij} \\
&= \bar{n}_{ij} - \bar{n}_{ji} - \sigma_{ij}
\end{aligned} \tag{4.23}$$

From this last result, we can derive the update of the Lagrange parameters λ_{ij} with $(i, j) \in \mathcal{C}$,

$$\lambda_{ij} \leftarrow \begin{cases} \max(\lambda_{ij} + \alpha(\bar{n}_{ij} - \bar{n}_{ji} - \sigma_{ij}), 0) & \text{when } (\bar{n}_{ij} - \bar{n}_{ji}) \geq 0 \\ 0 & \text{when } (\bar{n}_{ij} - \bar{n}_{ji}) < 0 \end{cases} \tag{4.24}$$

As before, the gradient ascent step parameter α must be tuned and it is very sensitive to the value of the inverse temperature parameter θ . Algorithm 1 is easy to adapt in order to consider net flow capacity constraints (with $\sigma_{ij} = \sigma_{ji}$); only lines 10 and 11 must be modified according to Eqs. 4.22 and 4.24.

Notice that still another procedure could be derived when considering the edges as undirected and unique; that is, there are unique affinity and cost values associated with each edge $i \leftrightarrow j$. In that situation, the Lagrange function is defined in terms of m costs (instead of $2m$ in this work) and augmented costs cannot differ along the edge direction. A similar algorithm, updating only one Lagrange parameter per edge, could be derived for this case.

In practice, we however observed that the net flow constraint algorithm is slower to converge and more sensitive to the gradient step than simple flow constraints; in other words, the problem seems harder to solve. This is especially the case for small values of θ , where the process behaves more like a random walker. Indeed, in that situation, the addition of capacity constraints seems to be partly in conflict with the objective of walking randomly and moving back and forth. One way to handle this issue [82] would be first to define a **directed acyclic graph** (DAG, deduced, e.g., from the electric potential on nodes when imposing a +1 potential at the source node and a 0 potential at the target node, followed by a topological sort). This will enable a RSP problem with simple capacity constraints to be solved on this DAG. This has the additional advantage that it should be more efficient (we avoid cycles and can use dynamic programming techniques to compute the RSP solution) and scale to larger graphs. This extension is left for future work.

4.4 Experiments

In this section, we present two illustrative examples of the use of capacity constraints on the edge flows of a graph as well as a brief study of the scalability of Algorithm 1 on max flow problems.

4.4.1 First illustrative example

The first example illustrates the net expected number of visits to nodes $\bar{j}$ (see Eq. 4.18) over the RSP paths distribution between one source node s and one target node t on a 20×20 grid, in different situations obtained after running Algorithm 1 with net flow capacity constraints (see Section 4.2). Nodes were linked to their neighbors⁸ with a unit affinity and a unit cost. In the first situation illustrated in Fig. 4.1a, we did not set any capacity constraint, and the net expected number of visits is represented in Figs. 4.1b to 4.1d for several θ values. As expected, walkers followed a trajectory close to the diagonal of the grid, representing the shortest paths between the source node and the target node. Furthermore, we can observe that the net expected number of visits to nodes is more dispersed around the diagonal when θ is small and more concentrated when θ is large. This behavior was also expected because

⁸Only horizontal and vertical neighbors are considered (no diagonal edge).

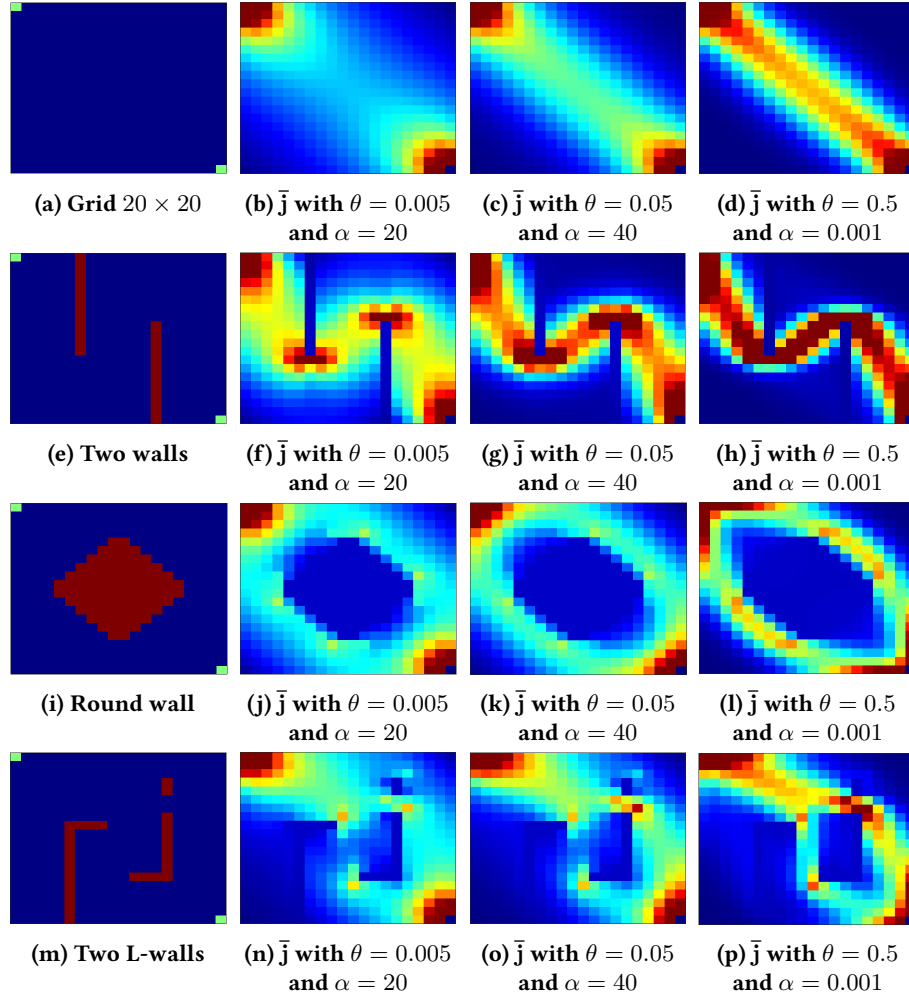


Figure 4.1: Net expected number of visits to nodes $\bar{j}$ (see Eq. 4.18) obtained with the capacity-constrained RSP Algorithm 1 on a 2D grid with several obstacles between two nodes. The source node s (upper left corner) and the target node t (lower right corner) are highlighted in green. All the edges around the obstacles are constrained to low capacities. Red represents a higher $\bar{j}$ and blue a lower one. The trajectory of $\bar{j}$ always follows the shortest path between the two nodes, taking the obstacles into account. The $\bar{j}$ is more dispersed when θ is small and more concentrated when θ is large.

the RSP model favors the least-cost paths when $\theta \rightarrow \infty$ and graph exploration when $\theta \rightarrow 0^+$ (see Section 3.3 for details).

In the other situations, we placed obstacles (porous walls) by constraining the capacities of all the edges linking the nodes represented in red in Figs. 4.1e, 4.1i, and 4.1m to 0.01. As can be observed in Figs. 4.1f to 4.1h, 4.1j to 4.1l, and 4.1n to 4.1p, in these situations, the net expected number of visits to nodes no longer concentrates around the diagonal, but instead closely follows the obstacles. This trajectory reflects the least-cost paths between the source node and the target node, avoiding the low-capacity obstacles. As before, the expected net number of node visits becomes increasingly dense around obstacles as θ increases.

4.4.2 Second illustrative example

Our second illustrative example was taken from [227] and makes a link between the RSP with net flow capacity constraints and the maximum flow problem. It aimed to show that Algorithm 1 could be used to compute the maximum flow (which takes a value of 12 in this example) between the source node s and the target node t in the undirected graph G presented in Fig. 4.2. Each edge of this graph has a unit cost and affinity, as well as capacities, which are shown on the illustration. Note that, in practice, because the RSP model assumes a unit input flow, all capacities are scaled⁹ so that the value of the computed maximum flow after scaling lies between 0 and 1. The maximum flow of G is then obtained by the reverse transformation.

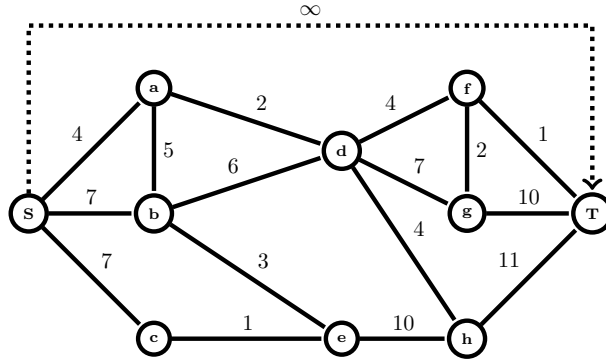


Figure 4.2: A small undirected graph composed of eight nodes [227]. The values on the edges represent capacities; moreover, all edge costs are set to 1, except the added shortcut edge (dashed line) whose cost is 10. The maximum possible flow between the source node s and the target node t is 12, and is equal to the min-cut between nodes $\{a, b, c\}$ and $\{d, e\}$.

⁹We divide capacities by a graph cut value (an upper bound on the min-cut), such as the cut between nodes $\{f, g, h\}$ and t : 22 in our example.

To find the max-flow, we added a directed edge from s to t (dashed line) with infinite capacity and an edge cost of 10 to the original graph. This new edge introduced a ‘shortcut’ allowing the passage of all the overflow of the graph, but with a higher cost. Theoretically, our algorithm avoids going through this shortcut as much as possible for a sufficiently large θ value, because it has a high cost compared to the other trajectories in the graph. Therefore, it should try to maximize the flow that travels through the original graph before using this shortcut edge.

Fig. 4.4 shows the evolution of the flow between the nodes $\{f, g, h\}$ and t (the flow through the original graph), provided by Algorithm 1, and thus satisfying the capacity constraints in terms of the θ parameter value. As observed in Figs. 4.3 and 4.4, this flow through the original graph reaches almost exactly the maximum flow value of 12 for all values of θ larger than 1. However, when θ is low (close to zero), the walks become increasingly random, and no longer consider costs (see Eq. 3.20). For that reason, part of the total flow goes through the shortcut, even if this is not optimal in terms of cost. This explains the reduction of the flow through the original graph when θ is close to zero.

4.4.3 Scalability experiment

In this subsection, we study the computational efficiency of Algorithm 1. More specifically, we compare the computation time of Algorithm 1 to those of three standard operations research algorithms and also to those of the linear solver CVX [117] on maximum flow problems. The three investigated operations research algorithms are the Edmonds-Karp adaptation of the Ford-Fulkerson augmenting path algorithm (AP) [89, 99], the Goldberg-Tarjan push-relabel algorithm (PR) [109], and the Boykov-Kolmogorov search trees algorithm (ST) [37].

We run all the algorithms 50 times on directed graphs of increasing size ranging from 10 to 250 nodes¹⁰. For each run, we generate a different graph and we associate with each edge a capacity randomly chosen between 1 and 25 and a unit cost. Regarding the capacity-constrained RSP (cRSP), we add a directed edge between the source and the target nodes with infinite capacity and a cost equivalent to the graph size. Moreover, we set the values of the parameters θ and α to respectively 1 and 0.5. We report the evolution of the mean computation time over the 50 runs for each algorithm in Fig. 4.5. All results were obtained with Matlab (version R2020a) running on Intel Core i7-8750H CPU@4.10GHz and 32 GB of RAM.

From Fig. 4.5, we can observe that the cRSP, the AP, the PR, and the ST

¹⁰Note that we had to limit the size of the graphs to 250 nodes as the free version of the linear solver CVX fails to solve larger problems in our setup.

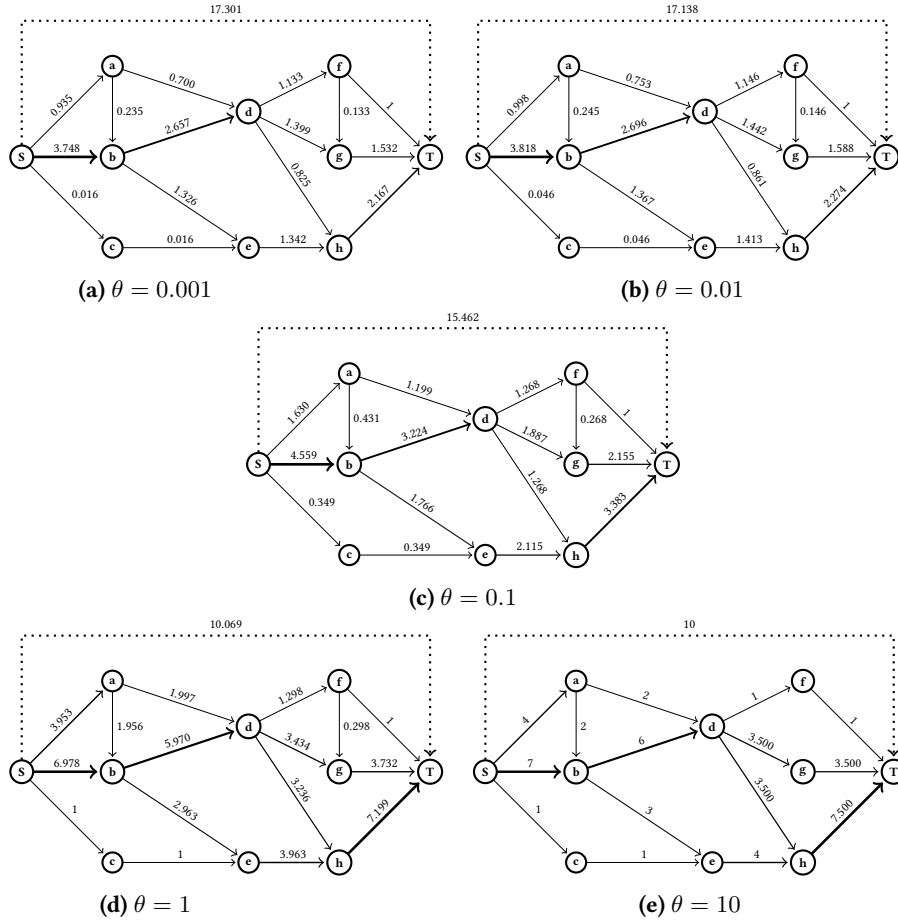


Figure 4.3: Representation of the flow from s to t depending on the value of θ , for the capacity-constrained RSP and the graph appearing in Fig. 4.2. The thickness of the arrows is scaled with respect to the largest net flow present in the graph. The flow between s and t increases with θ and almost reaches its maximum value of 12 when $\theta \geq 1$.

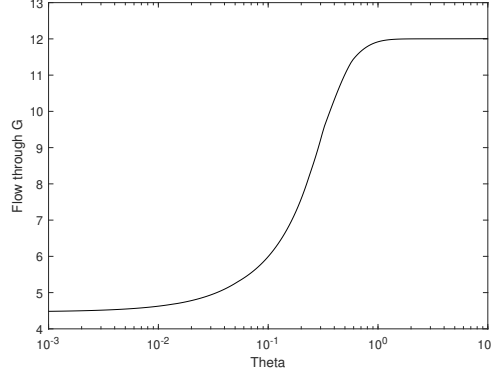


Figure 4.4: Evolution of the flow between nodes $\{f, g, h\}$ and t satisfying the capacity constraints (total flow in the original graph G , without the shortcut edge), provided by Algorithm 1, in terms of the θ parameter, with a gradient step $\alpha = 1/\theta$. The intersection between the curve and the y -axis when $\theta \rightarrow 0^+$ is 4.699. Note that the x -axis is scaled logarithmically. The flow increases with θ and reaches its maximum value of 12 when $\theta > 1$.

perform much faster than the linear solver CVX. Cuturi already observed that solving optimal transport problems with entropy regularization can be significantly more efficient than using standard linear programming methods in some situations [69, 221]. We can also see that the cRSP is slower than the three operations research algorithms.

This result was expected as the cRSP has a time complexity superior to those of the AP, the PR, and the ST. As a reminder, the cRSP complexity is of the order $2k \cdot \mathcal{O}(n^3)$, where k is the number of iterations needed for convergence. While the complexity of the AP, the PR, and the ST are respectively $\mathcal{O}(nm^2)$, $\mathcal{O}(n^2m)$, and $\mathcal{O}(n^2m|C|)$, where n is the number of nodes, m is the number of edges, and $|C|$ is the cost of the minimum cut.

However, we have to stress that the capacity-constrained RSP has not been specifically designed to solve max flow problems as opposed to the investigated operational research algorithms. Also, as a reminder, the optimization technique we use to compute the Lagrange parameters (see Section 4.2) is quite basic and could be modified to make Algorithm 1 more efficient.

4.5 Conclusion

In this chapter, we developed an extension of the randomized shortest path (RSP) model. This extension deals with capacity constraints on edges for both raw and net flows within the RSP model. Two algorithms solving the constrained problem for raw and net flows are developed by exploiting the La-

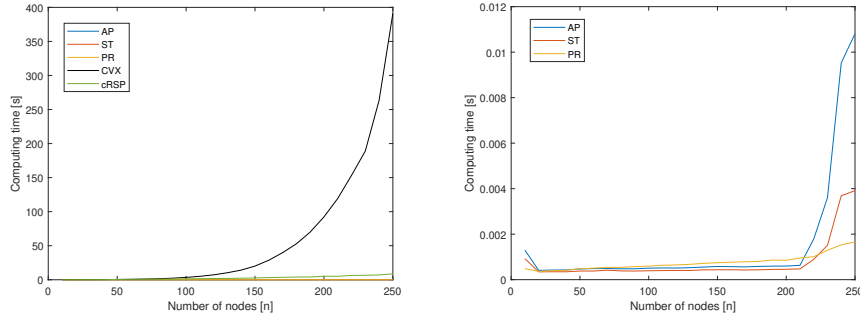


Figure 4.5: Evolution of the mean computation times of all methods (left) and of the three operations research algorithms (right) on directed graphs of increasing size n . Note that the scale of the two graphs is not the same. The linear solver CVX is slower than all other methods. All three operations research algorithms are faster than the cRSP, especially the PR.

grange duality. Moreover, these algorithms also take advantage of a feature of the RSP model, namely that most quantities of interest can be computed in closed-form using standard matrix operations.

Therefore, this contribution allows to apply the RSP model to a whole series of problems containing capacity constraints. As mentioned in the introduction, these problems are quite common in real life and are mostly studied in the operations research field, where capacity constraints appear, for instance, in the minimum cost flow and maximum flow problems [3, 139].

Moreover, the scalability experiment of maximum flow problems showed that our algorithm (Algorithm 1) performs much faster than the standard linear solver CVX [117] to solve such problems. However, our algorithm is slower than the three studied standard operations research algorithms. Therefore, reducing the complexity of our algorithm would make it more competitive with existing standard solutions which do not integrate randomness.

Regarding future works, we are interested in applying the proposed algorithms to operations research problems. Indeed, it would certainly be interesting to compare the RSP solution (with entropy regularization) to more standard algorithms to solve minimum cost flow problems and minimum cost flow with capacity constraints problems [3, 163]. A first small comparison is proposed in Subsection 5.5.3 in the context of transportation problems.

Another interesting future work would be to apply our algorithms to solve larger maximum flow problems. However, their complexity of $\mathcal{O}(n^3)$ does not allow us to use them on larger graphs that contain millions of nodes. One solution to this complexity problem could be to investigate more sophisticated optimization techniques, going beyond the simple gradient technique used to

solve the capacity-constrained RSP problem in Section 4.2. An interesting approach would be to adapt the sequential minimal optimization technique proposed by Platt [224] to our problem, as it has already been done for other maximum entropy problems in [137].

Still another interesting idea would be to try to reformulate the optimization problem in terms of expected net costs instead of expected costs in Eq. 4.20.

We also plan to explore a route that could improve the scalability of the proposed algorithms on sparse graphs. Following [82], the idea would be to extract a directed acyclic graph (DAG) from the original s - t graph by, for instance, performing a breath-first traversal or computing the electric current flow between the source node s and the target node t . It should therefore be possible to efficiently derive the optimal policy (and, consequently, the directed flows) on this DAG by using the Bellman-Ford-like expression that computes the free energy directed distance (see Eq. 8.6 and [105]). This idea has recently been used in [280] to model the behavior of animals in large landscapes. It would also be interesting to explore the introduction of capacity constraints on a DAG, as already discussed in Subsection 4.3.2.

Relative entropy-regularized optimal transport on a graph

5

This chapter investigates a flow-constrained randomized shortest path [156, 238, 300] formulation to the relative entropy-regularized, or randomized, **optimal transport problem** on a graph with multiple inputs and outputs having fixed marginal probabilities (or margins, providing fixed unit input and output flows), studied in [120, 123]. The previous works of [120, 123] extended the relative entropy-regularized optimal transport problem (see the recent work [69], but also [91, 144, 221, 297]) to a graph structure.

More globally, the optimal transport on a graph has been studied extensively in the operations research field, where it is known as the **minimum cost flow** in a network or the **minimum cost circulation** problem (see [3, 139, 295] and introduction of Chapter 4 for more details). Intuitively, it aims to carry goods from some input nodes to output nodes with least expected cost while maintaining a prescribed level of relative entropy of the path probabilities connecting inputs to outputs. In this problem, the input flows (proportion of goods carried from input nodes) and output flows (proportion of goods carried from output nodes) are constrained to be equal to some predefined values (marginal probabilities), which are not present in the standard **randomized shortest path** (RSP, [156, 238, 300]) and **bag-of-paths** (BoP, [105, 190]) models (see Chapter 3).

The introduced algorithm solving this problem provides an optimal randomized policy balancing exploitation and exploration through a simple iterative algorithm inspired by Chapter 4. This resulting policy takes the form of a time-homogeneous biased Markov chain guiding the agent from input nodes to output nodes with minimal expected cost for a given entropy level, while satisfying the flow constraints on the input and output nodes.

Similarly to the standard randomized shortest path and bag-of-paths models¹, the model is monitored by a parameter θ in such a way that, when

¹As already mentioned in Section 3.3, the main difference between the BoP and the RSP

θ goes to infinity, it approximates the optimal, lowest-cost, solution to the transportation problem. Conversely, when θ is close to zero, the solution approaches a random walk behavior (provided a priori by the user) in terms of relative entropy (also called Kullback-Leibler divergence). Thus, when varying θ , the model interpolates between an optimal (exploitation) and a random (exploration) behavior.

The main idea, in comparison with the previous works [120, 123], is the following. The relative entropy-regularized optimal transport on a graph problem studied in [123] is based on a BoP formulation where the set of all paths between the source and target nodes is considered. On the contrary, in this chapter, the problem is rephrased within a RSP formalism [156, 238] only considering the set of paths from one single source supernode and one target supernode, both added to the original network. The introduced model is therefore called the **margin-constrained bag-of-paths** model along the chapter, in order to remain consistent with [123]. Moreover, to highlight the links between these two formalisms, the margin-constrained bag-of-paths surprisal distance measure between nodes originally derived in [123] is re-derived within the RSP formalism.

Furthermore, the main advantage of this reformulation into a source-target RSP problem is that it allows us to easily define capacity constraints on edge flows, as presented in Chapter 4. As developed in the introduction of this previous chapter, flow capacity constraints are relevant in many graph-based applications such as traffic or passenger volume constraints on transportation networks [3]. Therefore, the work presented in Chapter 4 allows us to reformulate the margin-constrained BoP on a graph problem into a capacity constrained RSP problem which has its own merits compared to the previous works [120, 123]. This new formulation is, however, slower than the dedicated algorithm developed in [123] so it should only be used when capacity constraints are indeed present.

The main motivation of this chapter is therefore to extend the RSP model by introducing input and output flow constraints in order to broaden the scope of this model to a wider class of minimum cost flow problems requiring both flow and capacity constraints on edges. In short, the main contributions of this chapter are:

- The introduction of a new algorithm to solve the relative entropy-regularized optimal transport (in terms of minimum expected cost flow) on a graph problem by considering **edge flow constraints**;

formalism is that, for the BoP, all possible paths in the network are considered [105, 190], whereas only source-target paths connecting two nodes of interest are considered in the RSP [156, 238, 300]. The RSP therefore avoids the need to define prior distributions on source and on target nodes because there is only one single source and target.

- ▶ The introduction of a new algorithm to solve the relative entropy-regularized optimal transport on a graph problem by considering both **flow** and **capacity constraints** on edges;
- ▶ The derivation of the margin-constrained bag-of-paths surprisal distance between nodes of a graph [123] inside the RSP formalism;
- ▶ The application of these two algorithms to solve transportation problems;

The next section contains the related work. Section 5.2.1 presents the notation and some preliminaries. After, Section 5.3 develops the new RSP-based algorithm relying on flow constraints. Then, the RSP-based algorithm combining flow and capacity constraints is presented in Section 5.4. Section 5.5 introduces how these algorithms can be used to solve transportation problems. Finally, Section 5.6 presents the conclusions of the chapter.

5.1 Related work

As discussed in the introduction, the optimal transport on a graph has been studied extensively in the operations research field, where it is known as the minimum cost flow in a network or the minimum cost circulation problem (see, e.g., [3, 139, 295]). Generally, it admits a linear programming formulation, as it will be shown later in this chapter (see Eq. 5.38), but several alternatives, including more efficient, algorithms have been proposed in the literature.

The entropy-regularized extension of this problem on a bipartite network has already been studied in the field of spatial geography and economics [296, 297, 91] where it is known as the gravity model; see also [231] for a recent account. It corresponds to a regularized optimal matching problem on this bipartite graph with fixed marginals on the nodes. This formulation was revisited and extended in machine learning by Cuturi [69], who showed that using Sinkhorn’s matrix scaling algorithm is much more efficient than standard linear programming solvers. Cuturi also observed that the derived entropy-regularized distance improves upon classic optimal transport distances on supervised classification tasks. A large amount of subsequent related work was published, providing more in-depth analysis and extensions; see for instance [221].

The entropy-regularized optimal transport on a general directed graph, which is the main focus of this work, was studied in [55, 56], where the authors develop optimal transportation policies along bounded paths of finite length. Interestingly, they show that this problem is closely related to the discrete counterpart of the Schrödinger bridge problem (see, e.g., [218, 54]).

The optimal randomized policy, which is time-dependent in general, is derived. In parallel to this work, [120, 123] investigated transportation models on a graph considering all possible paths, leading to a Markovian time-homogeneous policy.

In addition, and very recently, [10] proposed an algorithm solving the same entropy-regularized optimal transport problem on a graph while, in addition, considering capacity constraints on the edges and storage capacities on the nodes. The developed procedure uses Dykstra's iterative projection algorithm to solve the problem. Notice that this last work is therefore closely related to the work presented in this chapter, the differences being that (1) we do not consider storage capacities on the nodes, (2) our algorithm repeatedly uses the RSP model solving a system of linear equations, to compute the optimal policy, and (3) the obtained policy is of the form of a time-homogeneous Markov chain, instead of edge flows.

5.2 Notation, problem statement, and preliminaries

This section aims first to present and recall some necessary notation already presented in Chapter 2 in Subsection 5.2.1. Then, the problem studied in this chapter is described in Subsection 5.2.2. Finally, the notion of extended graph is introduced in Subsection 5.2.3.

5.2.1 Notation

Assume a directed, strongly connected and weighted (see Section 2.1), graph $G = \{\mathcal{V}, \mathcal{E}\}$ containing $(n - 2)$ nodes², in which we have to carry goods from a predefined set of input nodes $\mathcal{In}$ to a set of output nodes $\mathcal{Out}$. The user specifies the proportions of non-negative, continuous, **flow** σ_i^{in} coming from each input $i \in \mathcal{In}$ as well as the proportion of flow delivered to each output σ_j^{out} , $j \in \mathcal{Out}$. For all other nodes $i \notin \mathcal{In}$, we set $\sigma_i^{\text{in}} = 0$. Symmetrically, for all nodes $j \notin \mathcal{Out}$, $\sigma_j^{\text{out}} = 0$. In general, input nodes are almost always different from output nodes, $\mathcal{In} \cap \mathcal{Out} = \emptyset$, but this assumption is not needed in the model.

Let us further assume

$$\begin{cases} \sum_{i \in \mathcal{In}} \sigma_i^{\text{in}} = 1 & \text{with all } \sigma_i^{\text{in}} \geq 0 \\ \sum_{j \in \mathcal{Out}} \sigma_j^{\text{out}} = 1 & \text{with all } \sigma_j^{\text{out}} \geq 0 \end{cases} \quad (5.1)$$

²See later for the justification; an extended graph with two additional nodes, and thus n nodes in total, is defined later in Subsection 5.2.3.

In other words, a unit flow is considered. If we have to transport a non-unitary flow, we simply find the solution for a unit flow and then multiply the quantities by the total flow. The $n \times 1$ column vector containing the input flow of each node is denoted by σ_{in} whereas the vector containing the output flows is σ_{out} .

Moreover, it is assumed that a non-negative **local cost** c_{ij} (see Subsection 2.2.2) is associated with each edge (i, j) , reflecting the penalty of following this edge (it can be a distance, a cost, a travel time, an energy consumption, etc). The **cost matrix** $\mathbf{C}$ contains the individual costs c_{ij} as elements. When there is no edge from node i to node j we consider the cost c_{ij} to be infinite. The graph G is also associated with an **adjacency matrix** $\mathbf{A}$ (see Subsection 2.2.1) containing local affinities, or weights, $\{a_{ij}\}$ between nodes. When there is no direct link between two nodes i, j , $a_{ij} = 0$. Matrices $\mathbf{C}$ and $\mathbf{A}$ are given and depend on the problem at hand. The **transition probability matrix** $\mathbf{P}$ associated with the natural random walk on this graph G is defined in the usual way (see Eq. 2.6 and Subsection 2.2.4 for details).

5.2.2 Problem statement

Given this context, the problem is to find the best policy, which takes the form of path probabilities, to carry the input flow from the source nodes to the target nodes, minimizing the expected cost along the paths connecting sources to targets, while

1. keeping a given level of exploration quantified by the Kullback-Leibler divergence between path probabilities and complete random paths provided by the natural random walk (see Eq. 2.7), and
2. satisfying the flow constraints stating that input flows and output flows are fixed to σ_i^{in} for each node $i \in \mathcal{In}$ and σ_j^{out} for each node $j \in \mathcal{Out}$.

This problem is solved in Section 5.3. Before that, let us introduce an extended graph G_{ext} by adding two supernodes to G .

5.2.3 Definition of an extended, single-source and target, graph

For convenience, we now transform the original graph G into a new, equivalent, single-source single-target graph $G_{\text{ext}} = \{\mathcal{V}_{\text{ext}}, \mathcal{E}_{\text{ext}}\}$ with n nodes in a standard way [3, 139]³. Two nodes – one **source node** (a supernode indexed as node 1) and one **target node** (a supernode indexed as node n , where n is the total number of nodes in this new extended graph) are added to the graph. All the other nodes in $\mathcal{V}$ (nodes 2 to $(n - 1)$) and their connections $\mathcal{E}$ remain

³This kind of trick is common in network flow algorithms.

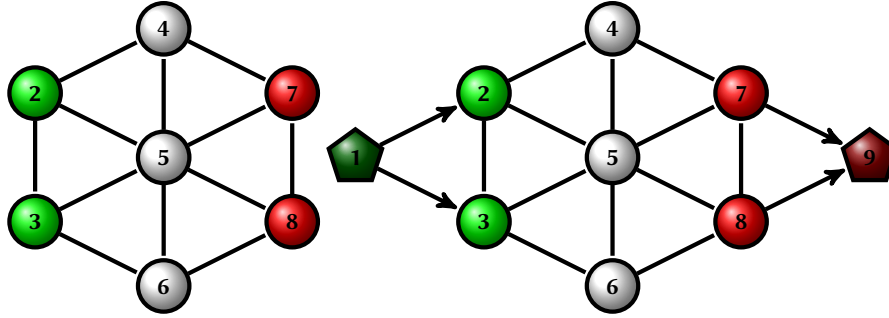


Figure 5.1: On the left, a small undirected graph G with two input nodes $\mathcal{I}n = \{2, 3\}$ (in green) and two output nodes $\mathcal{O}ut = \{7, 8\}$ (in red). On the right, the extended graph G_{ext} of this small undirected graph G with one source supernode 1 (in dark green) connected to all the input nodes and one target supernode $n = 9$ (in dark red) connected to all the output nodes. Therefore, on this extended graph, the source supernode is node 1 and the target supernode is node n (the total number of nodes in G_{ext}).

the same as in G (G is a subgraph of G_{ext}); therefore, $\mathcal{V}_{\text{ext}} = \mathcal{V} \cup \{1, n\}$. All nodes with label $i \in \mathcal{V}$ keep the same index numbering in $\mathcal{V}_{\text{ext}}$. Moreover, the new source node (node 1) is only connected by directed edges to the input nodes with a zero cost, and the output nodes are only connected to the new target node (node n) by a directed edge, also with zero cost (no penalty when following these edges). In order to be equivalent to the original graph G , the source node generates a unit flow while the target node is made killing and absorbing (a cemetery or sink node), and thus absorbs this unit flow. This new graph is called the **extended graph**. A toy example of this concept is presented in Fig. 5.1.

5.2.3.1 Adjacency matrix of the extended graph

We might now ask ourselves which weights (values of corresponding elements of the adjacency matrix) should be associated with the edges of node 1 and node n . First of all, as explained at the beginning of this subsection, node 1 is only connected by directed edges to the input nodes and the output nodes are only connected to node n by a directed edge. Therefore, the weights associated with the directed edges between the input nodes and node 1 and between node n and the output nodes are zero because these edges do not exist.

Before defining the weights associated with the existing edges incident to node 1 and node n , let us first recall two fundamental Markov chain quantities defined on G_{ext} already presented in Subsection 4.1.2. These quantities are the expected number of visits to node $i \in \mathcal{V}$, denoted as $\bar{n}_i$, and the expected

number of passages (also called the flow) through edge (i, j) , $\bar{n}_{ij}$. Moreover, note that by conservation of flows, we have $\bar{n}_j = \sum_{i \in \mathcal{P}red(j)} \bar{n}_{ij}$ and $\bar{n}_{ij} = \bar{n}_i p_{ij}^{\text{ext}}$ for any $j \in \mathcal{V}$.

Now, regarding the directed edges coming out of node 1, a natural requirement would be that the weights of these edges are proportional to the input flows, in such a way that the corresponding transition probabilities are exactly equal to these input flows σ_{in} . Indeed, because node 1 generates a unit flow, $\bar{n}_1 = 1$, it is necessary that the transition probabilities $p_{1i}^{\text{ext}} = \sigma_i^{\text{in}}$ to ensure that $\bar{n}_{1i} = \bar{n}_1 p_{1i}^{\text{ext}} = \sigma_i^{\text{in}}$ by conservation of flows for any $i \in \mathcal{In}$.

For the directed edges incident to node n , the choice of the weights is slightly more difficult. These weights, denoted as $\mathbf{w}$, can either be provided explicitly by the user or computed in such a way that the corresponding transition probabilities guarantee, by conservation of flows, that $\bar{n}_{jn} = \bar{n}_j p_{jn}^{\text{ext}} = \sigma_j^{\text{out}}$ for any $j \in \mathcal{Out}$. In other words, the $\mathbf{w}$ should be chosen in order to have σ_{out} as output distribution of the flows. However, compared to the previous situation, the flow of each node $j \in \mathcal{Out}$, $\bar{n}_j$, is not equal to 1, and thus needs to be computed, which slightly complicates the determination of $\mathbf{w}$. The choice of $\mathbf{w}$ is further developed in the two following subsections.

Therefore, the adjacency matrix of the extended graph, $\mathbf{A}_{\text{ext}}$, can be written as

$$\mathbf{A}_{\text{ext}} = \begin{matrix} & \begin{matrix} 1 & \{2, \dots, (n-1)\} = \mathcal{V} & n \end{matrix} \\ \begin{matrix} 1 \\ \{2, \dots, (n-1)\} = \mathcal{V} \\ n \end{matrix} & \begin{bmatrix} 0 & \sigma_{\text{in}}^T & 0 \\ \mathbf{0} & \mathbf{A} & \mathbf{w} \\ 0 & \mathbf{0}^T & 0 \end{bmatrix} \end{matrix} \quad (5.2)$$

where $\mathbf{0}$ is the null column vector full of 0s. Similarly, the cost matrix becomes

$$\mathbf{C}_{\text{ext}} = \begin{matrix} & \begin{matrix} 1 & \{2, \dots, (n-1)\} = \mathcal{V} & n \end{matrix} \\ \begin{matrix} 1 \\ \{2, \dots, (n-1)\} = \mathcal{V} \\ n \end{matrix} & \begin{bmatrix} 0 & \mathbf{0}^T & 0 \\ \mathbf{0} & \mathbf{C} & \mathbf{0} \\ 0 & \mathbf{0}^T & 0 \end{bmatrix} \end{matrix} \quad (5.3)$$

5.2.3.2 Transition matrix of the natural random walk on the extended graph

If the weights $\mathbf{w}$ are set by the user, the transition matrix $\mathbf{P}_{\text{ext}} = (p_{ij}^{\text{ext}})$ of the natural random walk on this extended graph G_{ext} can easily be computed by Eq. 2.7 from $\mathbf{A}_{\text{ext}}$ instead of $\mathbf{A}$. In that case, our model computes a policy (transition probabilities followed to carry the goods) interpolating between the optimal expected lowest-cost policy and the one closest (in terms of Kullback-Leibler divergence) to the natural random walk. But it is, in general, when $\theta \rightarrow 0$, *not exactly* to this natural random walk transition matrix

because of the flow constraints in Eq. 5.1 which are *in general not satisfied* for edges entering node n in the extended graph G_{ext} .

It is, however, possible to find sets of values of the weights $\mathbf{w}$ such that the flow in each of these edges is exactly equal to σ_j^{out} when considering a natural random walk on the extended graph [120]. If we do not have a good reason to choose the weights $\mathbf{w}$, it seems reasonable to compute the consistent transition probabilities because it removes the arbitrariness associated with the choice of these weights.

Let us deduce the form of the transition matrix $\mathbf{P}_{\text{ext}}$ of the natural random walk on the extended graph as well as the value of $\mathbf{w}$ in Eq. 5.2. Clearly, its first row (source node 1) is $[0 \ \sigma_{\text{in}}^T \ 0]$ and the last row (target node n) is full of 0s (killing, absorbing, node). The row sums corresponding to the other nodes $i \in \mathcal{V}$ (belonging to the original graph G) of the new, extended, $n \times n$ matrix $\mathbf{A}_{\text{ext}} = (a_{ij}^{\text{ext}})$ (Eq. 5.2) are equal to⁴ $[\mathbf{A}_{\text{ext}}]_{i\bullet} = a_{i\bullet}^{\text{ext}} = a_{i\bullet} + w_i$ with $a_{i\bullet} = \sum_{j \in \mathcal{V}} a_{ij}$. Thus, for $i, j \in \mathcal{V}$ (in other terms, $i, j \notin \{1, n\}$), $p_{ij}^{\text{ext}} = a_{ij}/(a_{i\bullet} + w_i)$ and, for $j = n$ (last column), $p_{in}^{\text{ext}} = w_i/(a_{i\bullet} + w_i)$.

We now introduce the $(n-2) \times 1$ vector α containing the elements corresponding to nodes $i \in \mathcal{V}$ on the last column n of $\mathbf{P}_{\text{ext}}$ (see Eq. 5.6),

$$\alpha_i \triangleq p_{in}^{\text{ext}} = \frac{w_i}{a_{i\bullet} + w_i} \text{ for } i \in \mathcal{V} \quad (5.4)$$

hence containing elements i, n of the $n \times n$ transition matrix of the extended graph. This quantity corresponds to the probability of being killed in node i during the next step of the random walk (transiting to the absorbing/cemetery node n). Moreover, it is easy to verify that $1 - \alpha_i = a_{i\bullet}/(a_{i\bullet} + w_i)$, which is the probability of surviving (continuing the walk). Therefore, we readily obtain for $i, j \in \mathcal{V}$,

$$p_{ij}^{\text{ext}} = \frac{a_{ij}}{a_{i\bullet} + w_i} = \frac{a_{i\bullet}}{(a_{i\bullet} + w_i)} \times \frac{a_{ij}}{a_{i\bullet}} = (1 - \alpha_i)p_{ij} \quad (5.5)$$

where the p_{ij} are the transition probabilities of the natural random walk on the original graph G (see Eq. 2.7). Therefore, the **transition matrix** representing the killed random walk on the extended graph is

$$\mathbf{P}_{\text{ext}} = \begin{matrix} & \begin{matrix} 1 & \{2, \dots, (n-1)\}=\mathcal{V} & n \end{matrix} \\ \begin{matrix} 1 \\ \{2, \dots, (n-1)\}=\mathcal{V} \\ n \end{matrix} & \begin{bmatrix} 0 & \sigma_{\text{in}}^T & 0 \\ \mathbf{0} & (\mathbf{I} - \mathbf{Diag}(\alpha))\mathbf{P} & \alpha \\ 0 & \mathbf{0}^T & 0 \end{bmatrix} \end{matrix} \quad (5.6)$$

where $\mathbf{P}$ is the natural random walk on the original graph (see Eq. 2.7 and Subsection 2.2.4) and $\mathbf{Diag}(\alpha)$ is a diagonal matrix with α on its diagonal

⁴Recall that all nodes with label $i \in \mathcal{V}$ keep the same index in $\mathcal{V}_{\text{ext}}$.

and the elements of the vector α still have to be determined – this problem is solved in the next subsection. Interestingly, for nodes not belonging to $\mathcal{I}n$, these transition probabilities remain exactly the same as before, as long as the random walker is not absorbed.

Once α is computed, the elements of the weight vector $\mathbf{w}$ appearing in the adjacency matrix of Eq. 5.2 can be obtained from Eq. 5.4,

$$w_i = \frac{\alpha_i a_{i\bullet}}{1 - \alpha_i} \quad (5.7)$$

or, in matrix form,

$$\mathbf{w} = (\alpha \circ (\mathbf{A}\mathbf{e})) \div (\mathbf{e} - \alpha) \quad (5.8)$$

where $\circ$ and $\div$ are respectively the elementwise product and division and $\mathbf{e}$ is a column vector full of 1s.

5.2.3.3 Computing the values of α

In this subsection, the vector α is computed by following a reasoning similar to the one appearing in [120, 123].

The elements α_i with $i \in \mathcal{V}$ is set such that the flows in the output edges (i, n) , $\bar{n}_{in}$, are equal to the prescribed output flows, $\bar{n}_{in} = \bar{n}_i p_{in}^{\text{ext}} = \bar{n}_i \alpha_i = \sigma_i^{\text{out}}$. Thus, $\alpha_i = \sigma_i^{\text{out}} / \bar{n}_i$ and we therefore have to find the $\bar{n}_i$.

To do so, let us start from the previous identity, valid for each $j \neq \{1, n\}$ (and thus $j \in \mathcal{V}$) on the extended graph,

$$\begin{aligned} \bar{n}_j &= \sum_{i \in \text{Pred}(j)} \bar{n}_{ij} = \sum_{i \in \text{Pred}(j)} \bar{n}_i p_{ij}^{\text{ext}} \\ &= \sum_{i=1}^n \bar{n}_i p_{ij}^{\text{ext}} = \sum_{i=2}^n \bar{n}_i p_{ij}^{\text{ext}} + \underbrace{\bar{n}_1 p_{1j}^{\text{ext}}}_{\sigma_j^{\text{in}}} \\ &= \sum_{i=2}^n \bar{n}_i p_{ij}^{\text{ext}} + \sigma_j^{\text{in}} = \sum_{i=2}^{n-1} \bar{n}_i p_{ij}^{\text{ext}} + \underbrace{\bar{n}_n p_{nj}^{\text{ext}}}_0 + \sigma_j^{\text{in}} \\ &= \sum_{i \in \mathcal{V}} \bar{n}_i p_{ij}^{\text{ext}} + \sigma_j^{\text{in}} \quad \text{for each } j \in \mathcal{V} \end{aligned} \quad (5.9)$$

But, we know from Eq. 5.6 that $p_{ij}^{\text{ext}} = (1 - \alpha_i) p_{ij}$ when $i, j \notin \{1, n\}$, which is injected in Eq. 5.9. Recalling also that $\bar{n}_i \alpha_i = \sigma_i^{\text{out}}$, we then obtain

$$\begin{aligned} \bar{n}_j &= \sum_{i \in \mathcal{V}} \bar{n}_i p_{ij}^{\text{ext}} + \sigma_j^{\text{in}} = \sum_{i \in \mathcal{V}} \bar{n}_i (p_{ij} - \alpha_i p_{ij}) + \sigma_j^{\text{in}} \\ &= \sum_{i \in \mathcal{V}} \bar{n}_i p_{ij} - \sum_{i \in \mathcal{V}} \underbrace{\bar{n}_i \alpha_i}_{\sigma_i^{\text{out}}} p_{ij} + \sigma_j^{\text{in}} \end{aligned}$$

$$= \sum_{i \in \mathcal{V}} \bar{n}_i p_{ij} - \sum_{i \in \mathcal{V}} p_{ij} \sigma_i^{\text{out}} + \sigma_j^{\text{in}} \quad \text{for each } j \in \mathcal{V} \quad (5.10)$$

or, in matrix form, where $\bar{\mathbf{n}}$ is the $(n-2) \times 1$ column vector containing the $\bar{n}_i$,

$$\bar{\mathbf{n}} = \mathbf{P}^T \bar{\mathbf{n}} - \mathbf{P}^T \boldsymbol{\sigma}_{\text{out}} + \boldsymbol{\sigma}_{\text{in}} \quad (5.11)$$

We thus have to solve

$$(\mathbf{I} - \mathbf{P}^T) \bar{\mathbf{n}} = \boldsymbol{\sigma}_{\text{in}} - \mathbf{P}^T \boldsymbol{\sigma}_{\text{out}} \quad (5.12)$$

and then compute $\boldsymbol{\alpha}$ from

$$\boldsymbol{\alpha} = \boldsymbol{\sigma}_{\text{out}} \div \bar{\mathbf{n}} \quad (5.13)$$

where $\div$ denotes the elementwise division.

As $\mathbf{P}^T$ is rank-deficient, the solution to Eq. 5.12 is the sum of a particular solution plus any vector in the null space of $(\mathbf{I} - \mathbf{P}^T)$ [242, 243, 245]. As the null space of $(\mathbf{I} - \mathbf{P}^T)$ is spanned by $\boldsymbol{\pi}$, the equilibrium distribution of the Markov chain on G with transition matrix $\mathbf{P}$ (see Algorithm 1 presented in [123] page 99 for details), we obtain

$$\bar{\mathbf{n}} = (\mathbf{I} - \mathbf{P}^T)^+(\boldsymbol{\sigma}_{\text{in}} - \mathbf{P}^T \boldsymbol{\sigma}_{\text{out}}) + \mu \boldsymbol{\pi} \quad (5.14)$$

where $+$ denotes the Moore-Penrose pseudoinverse and μ is the non-negative ‘persistence parameter’ [120]. To ensure that $\bar{\mathbf{n}} \geq \boldsymbol{\sigma}_{\text{out}}$ (so that there is a feasible solution), the parameter μ has to satisfy the following inequality [120],

$$\mu \geq \max_i \left\{ \frac{\sigma_i^{\text{out}} - \mathbf{e}_i^T (\mathbf{I} - \mathbf{P}^T)^+(\boldsymbol{\sigma}_{\text{in}} - \mathbf{P}^T \boldsymbol{\sigma}_{\text{out}})}{\pi_i} \right\} \quad (5.15)$$

Any value of μ satisfying this constraint provides valid flows.

Therefore the transition matrix of the extended graph is given by Eq. 5.6 where $\boldsymbol{\alpha}$ is provided by the previous Eqs. 5.13-5.14 and is pre-defined to satisfy Eq. 5.15. Then, the value of $\mathbf{w}$ can be obtained from $\boldsymbol{\alpha}$ by using Eq. 5.8.

5.3 Optimal transport on a graph in a constrained RSP model

Our formulation of the problem is based on the RSP model introduced in Section 3.3. The objective now is to compute the randomized shortest path solution on G_{ext} satisfying the source/target flow constraints stated in Subsection 5.2.2, and reformulated here for convenience,

$$\begin{cases} \bar{n}_{1i} = \sigma_i^{\text{in}} & \text{for each node } i \in \mathcal{I}n \\ \bar{n}_{jn} = \sigma_j^{\text{out}} & \text{for each node } j \in \mathcal{O}ut \end{cases} \quad (5.16)$$

where $\bar{n}_{ij}$ is the flow (expected number of passages) in edge (i, j) (see Eq. 4.2). For all other nodes $i \notin \mathcal{In}$ and $j \notin \mathcal{Out}$, we set $\bar{n}_{1i} = 0$ and $\bar{n}_{jn} = 0$.

As already mentioned in the introduction, the margin-constrained BoP problem on a graph has recently been studied in [123]. This chapter is based on an entropy regularization at the flow level while the approach of [123] adopts a bag-of-paths approach (see Section 3.1). Considering now multiple input nodes and output nodes, we want to solve the problem

$$\left| \begin{array}{ll} \text{minimize}_{\{P(\varphi)\}_{\varphi \in \mathcal{P}}} & \phi(P) = \sum_{\varphi \in \mathcal{P}} P(\varphi) \tilde{c}(\varphi) + T \sum_{\varphi \in \mathcal{P}} P(\varphi) \log \left(\frac{P(\varphi)}{\tilde{\pi}(\varphi)} \right) \\ \text{subject to} & \sum_{j \in \mathcal{V}} \sum_{\varphi_{ij} \in \mathcal{P}_{ij}} P(\varphi_{ij}) = \sigma_i^{\text{in}} \quad \forall i \in \mathcal{In} \\ & \sum_{i \in \mathcal{V}} \sum_{\varphi_{ij} \in \mathcal{P}_{ij}} P(\varphi_{ij}) = \sigma_j^{\text{out}} \quad \forall j \in \mathcal{Out} \end{array} \right. \quad (5.17)$$

where the set of considered paths is $\mathcal{P} = \cup_{i \in \mathcal{In}} \cup_{j \in \mathcal{Out}} \mathcal{P}_{ij}$ and $\tilde{c}(\varphi)$ is the total cost of a path (see Section 3.1 for details about these concepts).

It was shown in [123] that the solution can be obtained by iterative proportional fitting (also called matrix balancing or biproportional scaling), as for the standard, relaxed, optimal transport problem with entropy regularization (see for instance [69, 91, 144, 221, 297]).

In the present chapter, we adopt a different point of view (in comparison with [123]) for solving this problem by designing a new algorithm based on the constraints imposed on flows in the graph G_{ext} (see Eq. 5.16), inspired by Chapter 4. It is important to note that this formulation of the problem can readily integrate additional capacity constraints as well; see Chapter 4 and Section 5.4 for details. This makes the present formulation interesting for practical problems: it extends the scope of the optimal transport on a graph procedure introduced in [120, 123] to problems with capacity constraints.

The algorithm solving the relative entropy-regularized optimal transport on a graph problem is derived from results obtained in Chapter 4 by exploiting its Lagrange formulation and Lagrangian duality. It provides the optimal **randomized policy** taking the form of the transition matrix of a **biased random walk** on G_{ext} , as in the case of the standard RSP problem (see Eq. 4.4). It corresponds to a Markov chain on the extended graph biasing the random walk towards the output nodes while satisfying the input and output flow constraints of Eq. 5.16.

5.3.1 Lagrange function

The equality constraints (Eq. 5.16) can be expressed in the following Lagrange function defined on the extended graph G_{ext} with $\mathcal{P}_{1n}^h$ being the set of all

possible hitting paths from source node 1 to target node n ,

$$\begin{aligned}
\mathcal{L}(\mathbf{P}, \boldsymbol{\lambda}) = & \underbrace{\sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) \tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) \log \left(\frac{\mathbf{P}(\wp)}{\tilde{\pi}(\wp)} \right)}_{\text{free energy, } \phi(\mathbf{P})} \\
& + \mu \left(\sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) - 1 \right) \\
& + \sum_{i \in \mathcal{I}n} \lambda_i^{\text{in}} (\bar{n}_{1i} - \sigma_i^{\text{in}}) + \sum_{j \in \mathcal{O}ut} \lambda_j^{\text{out}} (\bar{n}_{jn} - \sigma_j^{\text{out}}) \quad (5.18)
\end{aligned}$$

Given that $\bar{n}_{ij} = \sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) \eta((i, j) \in \wp)$ with $\eta((i, j) \in \wp)$ being the number of times edge (i, j) appears on path $\wp$ (see Subsection 4.1.2), the Lagrange function (Eq. 5.18) defined on the extended graph G_{ext} becomes

$$\begin{aligned}
\mathcal{L}(\mathbf{P}, \boldsymbol{\lambda}) = & \underbrace{\sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) \tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) \log \left(\frac{\mathbf{P}(\wp)}{\tilde{\pi}(\wp)} \right)}_{\text{free energy, } \phi(\mathbf{P})} \\
& + \mu \left(\sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) - 1 \right) \\
& + \sum_{i \in \mathcal{I}n} \lambda_i^{\text{in}} \left[\sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) \eta((1, i) \in \wp) - \sigma_i^{\text{in}} \right] \\
& + \sum_{j \in \mathcal{O}ut} \lambda_j^{\text{out}} \left[\sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) \eta((j, n) \in \wp) - \sigma_j^{\text{out}} \right] \quad (5.19)
\end{aligned}$$

Note that the objective function to be minimized is convex and the equality constraints are all linear. The Lagrange function can be rearranged as

$$\begin{aligned}
\mathcal{L}(\mathbf{P}, \boldsymbol{\lambda}) = & \sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) \left[\underbrace{\tilde{c}(\wp) + \sum_{i \in \mathcal{I}n} \lambda_i^{\text{in}} \eta((1, i) \in \wp) + \sum_{j \in \mathcal{O}ut} \lambda_j^{\text{out}} \eta((j, n) \in \wp)}_{\text{augmented cost } \tilde{c}'(\wp) \text{ cumulated on path } \wp} \right] \\
& + T \sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) \log \left(\frac{\mathbf{P}(\wp)}{\tilde{\pi}(\wp)} \right) + \mu \left(\sum_{\wp \in \mathcal{P}_{1n}^h} \mathbf{P}(\wp) - 1 \right) \\
& - \sum_{i \in \mathcal{I}n} \lambda_i^{\text{in}} \sigma_i^{\text{in}} - \sum_{j \in \mathcal{O}ut} \lambda_j^{\text{out}} \sigma_j^{\text{out}}
\end{aligned}$$

$$\begin{aligned}
&= \underbrace{\sum_{\wp \in \mathcal{P}_{1n}^h} P(\wp) \tilde{c}'(\wp) + T \sum_{\wp \in \mathcal{P}_{1n}^h} P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right)}_{\text{free energy } \phi'(P) \text{ based on augmented costs}} \\
&\quad + \underbrace{\mu \left(\sum_{\wp \in \mathcal{P}_{1n}^h} P(\wp) - 1 \right) - \sum_{i \in \mathcal{In}} \lambda_i^{\text{in}} \sigma_i^{\text{in}} - \sum_{j \in \mathcal{Out}} \lambda_j^{\text{out}} \sigma_j^{\text{out}}}_{\text{does not depend on } P(\cdot)} \quad (5.20)
\end{aligned}$$

where $\tilde{c}'(\wp) = \sum_{\tau=1}^t c'_{s(\tau-1)s(\tau)}$ is the total cumulated augmented cost along path $\wp$ with the original costs c_{ij} being replaced by the *augmented costs* c'_{ij} on G_{ext} ,

$$c'_{ij} = \begin{cases} c_{ij}^{\text{ext}} + \lambda_j^{\text{in}} & \text{when } i = 1 \text{ and } j \in \mathcal{In} \\ c_{ij}^{\text{ext}} + \lambda_i^{\text{out}} & \text{when } i \in \mathcal{Out} \text{ and } j = n \\ c_{ij}^{\text{ext}} & \text{otherwise} \end{cases} \quad (5.21)$$

as the initial costs are equal to zero for edges starting in node 1 and ending in $\mathcal{In}$. The same holds for edges starting in $\mathcal{Out}$ and ending in node n . $\mathbf{C}'$ is the matrix containing these augmented costs. Besides, $\phi'(P)$ is the free energy (see Eq. 3.37 and Section 3.3) depending on these augmented costs.

Thus, in Eq. 5.20, everything happens as if the costs had been redefined by taking the Lagrange parameters into account. These Lagrange parameters can therefore be interpreted as additional costs necessary to satisfy the equality constraints. We now have to find the Lagrange parameters λ by using Lagrangian duality.

5.3.2 Exploiting Lagrangian duality

Following the same reasoning as in Subsection 4.2.2, we exploit the fact that, in this formulation of the problem, the Lagrange dual function and its gradient are easy to compute⁵. Moreover, because the objective function is convex and all the equality constraints are linear, there is only one global minimum and the duality gap is zero [27, 68, 119]. The optimum is a saddle point of the Lagrange function, and a common optimization procedure (sometimes called the Arrow-Hurwicz-Uzawa algorithm [11, 194]) consists of sequentially (i) solving the primal (finding the optimal probability distribution) while considering the Lagrange parameters as fixed, and then (ii) maximizing the dual (which is concave) with respect to the Lagrange parameters, until convergence.

⁵This is actually a standard result related to maximum entropy problems (see for instance [137]).

In our context, this provides the following steps, which are iterated until convergence,

$$\begin{cases} \mathcal{L}(\mathbf{P}^*, \boldsymbol{\lambda}) = \min_{\{\mathbf{P}(\varphi)\}_{\varphi \in \mathcal{P}_{1n}^h}} \mathcal{L}(\mathbf{P}, \boldsymbol{\lambda}) & \text{(compute the dual function)} \\ \mathcal{L}(\mathbf{P}^*, \boldsymbol{\lambda}^*) = \max_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{P}^*, \boldsymbol{\lambda}) & \text{(maximize the dual function)} \end{cases} \quad (5.22)$$

where we set $\mathcal{L}(\mathbf{P}, \boldsymbol{\lambda}) = \mathcal{L}(\mathbf{P}^*, \boldsymbol{\lambda}^*)$ at the end of each iteration. In short, the dual function is computed analytically and then maximized through a block coordinate ascent in terms of the Lagrange parameters $\boldsymbol{\lambda}$.

5.3.2.1 Computing the dual function

Let us first compute $\mathbf{P}^* = \arg \min_{\{\mathbf{P}(\varphi)\}_{\varphi \in \mathcal{P}_{1n}^h}} \mathcal{L}(\mathbf{P}, \boldsymbol{\lambda})$ satisfying the sum-to-one constraint. From Eq. 5.20, as for the RSP (see Section 3.3), the primal can be solved easily and exactly; it provides a Gibbs-Boltzmann distribution (see Eq. 3.21), but this time in terms of the *augmented costs*, $\tilde{c}'(\varphi)$, introduced in Eq. 5.21. It simply computes the new probability distribution $\mathbf{P}^*(\cdot)$.

Once $\mathbf{P}^*$ has been computed, as shown in Section 3.3, Eq. 3.37, $\phi(\mathbf{P}^*) = -T \log \mathcal{Z}_h'^{(n)}$ on G_{ext} , then the corresponding dual function (Eq. 5.20) can be rewritten in terms of the partition function (Eq. 3.25) on G_{ext} as

$$\begin{aligned} \mathcal{L}(\mathbf{P}^*, \boldsymbol{\lambda}) &= \phi'(\mathbf{P}^*) - \sum_{i \in \mathcal{I}n} \lambda_i^{\text{in}} \sigma_i^{\text{in}} - \sum_{j \in \mathcal{O}ut} \lambda_j^{\text{out}} \sigma_j^{\text{out}} \\ &= -T \log \mathcal{Z}_h'^{(n)} - \sum_{i \in \mathcal{I}n} \lambda_i^{\text{in}} \sigma_i^{\text{in}} - \sum_{j \in \mathcal{O}ut} \lambda_j^{\text{out}} \sigma_j^{\text{out}} \end{aligned} \quad (5.23)$$

where $\mathcal{Z}_h'^{(n)} = \sum_{\varphi \in \mathcal{P}_{1n}^h} \tilde{\pi}(\varphi) \exp[-\theta \tilde{c}'(\varphi)]$ is the partition function (Eq. 3.25) computed from the augmented costs $\mathbf{C}'$ (Eq. 5.21), which depends on the Lagrange multipliers $\{\boldsymbol{\lambda}_{\text{in}}, \boldsymbol{\lambda}_{\text{out}}\}$.

5.3.2.2 Maximizing the dual function

To compute the maximum of the dual function, we use a block coordinate ascent procedure [27, 166] optimizing sequentially with respect to the $\{\lambda_k^{\text{in}}\}_{k \in \mathcal{I}n}$ and then with respect to the $\{\lambda_k^{\text{out}}\}_{k \in \mathcal{O}ut}$. As the dual function is concave [119], the block coordinate ascent procedure converges to the global maximum.

Computing Lagrange parameters associated with input nodes. Let us start with the $\{\lambda_k^{\text{in}}\}_{k \in \mathcal{I}n}$. Recalling that $\partial_{c_{ij}}(-T \log \mathcal{Z}) = \tilde{n}_{ij}$ (see Eq. 4.2),

the gradient of the dual function with respect to λ_k^{in} can be computed for each node $k \in \mathcal{I}n$,

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{P}^*, \boldsymbol{\lambda})}{\partial \lambda_k^{\text{in}}} &= \frac{\partial(-T \log \mathcal{Z}_h'^{(n)})}{\partial \lambda_k^{\text{in}}} - \sigma_k^{\text{in}} = \frac{\partial(-T \log \mathcal{Z}_h'^{(n)})}{\partial c'_{1k}} \frac{\partial c'_{1k}}{\partial \lambda_k^{\text{in}}} - \sigma_k^{\text{in}} \\ &= \bar{n}_{1k} - \sigma_k^{\text{in}} \end{aligned} \quad (5.24)$$

where we used the definition of the augmented costs in Eq. 5.21. Expressing the fact that the gradient must cancel itself at the optimum simply reduces to the equality constraint of Eq. 5.16, which is common in maximum-entropy problems [143, 144] and was already observed in Subsection 4.2.2.2,

$$\bar{n}_{1k} = \sigma_k^{\text{in}} \quad \text{for each } k \in \mathcal{I}n \quad (5.25)$$

Now, from Eq. 4.2, $\bar{n}_{1k} = z_{11}^{\text{h}'(n)} p_{1k}^{\text{ext}} \exp[-\theta c'_{1k}] z_{kn}^{\text{h}'(n)} / z_{1n}^{\text{h}'(n)}$ and, as node 1 has no incoming edge, $z_{11}^{\text{h}'(n)} = 1$ (see Eq. 3.22). Moreover, again from Eq. 3.22, $(\mathbf{I} - \mathbf{W}^{(n)}) \mathbf{Z}_h'^{(n)} = \mathbf{Z}_h'^{(n)} - \mathbf{W}^{(n)} \mathbf{Z}_h'^{(n)} = \mathbf{I}$, and taking element 1, n provides

$$\begin{aligned} z_{1n}^{\text{h}'(n)} &= \sum_{k \in \mathcal{Succ}(1)} w_{1k}'^{(n)} z_{kn}^{\text{h}'(n)} \\ &= \sum_{k \in \mathcal{Succ}(1)} p_{1k}^{\text{ext}} \exp[-\theta c'_{1k}] z_{kn}^{\text{h}'(n)} \\ &= \sum_{k \in \mathcal{I}n} p_{1k}^{\text{ext}} \exp[-\theta \lambda_k^{\text{in}}] z_{kn}^{\text{h}'(n)} \end{aligned} \quad (5.26)$$

where $\mathcal{Succ}(1)$ is the set of successor nodes of node 1, which corresponds to the set of input nodes $\mathcal{I}n$. Thus, from the previous discussion and Eq. 4.2,

$$\bar{n}_{1k} = \frac{z_{11}^{\text{h}'(n)} p_{1k}^{\text{ext}} \exp[-\theta c'_{1k}] z_{kn}^{\text{h}'(n)}}{z_{1n}^{\text{h}'(n)}} = \frac{p_{1k}^{\text{ext}} \exp[-\theta \lambda_k^{\text{in}}] z_{kn}^{\text{h}'(n)}}{\sum_{k' \in \mathcal{I}n} p_{1k'}^{\text{ext}} \exp[-\theta \lambda_{k'}^{\text{in}}] z_{k'n}^{\text{h}'(n)}} \quad (5.27)$$

because $z_{11}^{\text{h}'(n)} = 1$. We observe that $\sum_{k=1}^n \bar{n}_{1k} = 1$, meaning that the input flow is 1, as it should be.

Note from Eq. 3.26 that, as edge $(1, k)$ with $(k \neq 1)$ does not appear on any path in $\mathcal{P}_{kn}^h$ by construction of the extended graph, the backward variables $z_{kn}^{\text{h}'(n)}$ with $k > 1$ do not depend on the λ_k^{in} . Thus, from Eq. 5.25, to update the λ_k^{in} , we have to solve

$$\frac{p_{1k}^{\text{ext}} \exp[-\theta \lambda_k^{\text{in}}] z_{kn}^{\text{h}'(n)}}{\sum_{k' \in \mathcal{I}n} p_{1k'}^{\text{ext}} \exp[-\theta \lambda_{k'}^{\text{in}}] z_{k'n}^{\text{h}'(n)}} = \sigma_k^{\text{in}} \quad \text{for all } k \in \mathcal{I}n \quad (5.28)$$

These equations are invariant when adding a constant to the Lagrange parameters. Therefore, we impose the convenient condition $\sum_{k \in \mathcal{I}n} \lambda_k^{\text{in}} \sigma_k^{\text{in}} = 0$.

Therefore, to compute the Lagrange parameters associated with input nodes, the following system of logistic function equations needs to be solved

$$\left\{ \begin{array}{l} \frac{p_{1k}^{\text{ext}} \exp[-\theta \lambda_k^{\text{in}} z_{kn}^{\text{h}'(n)}]}{\sum_{k' \in \mathcal{I}n} p_{1k'}^{\text{ext}} \exp[-\theta \lambda_{k'}^{\text{in}} z_{k'n}^{\text{h}'(n)}]} = \sigma_k^{\text{in}} \quad \text{for all } k \in \mathcal{I}n \\ \text{subject to } \sum_{k \in \mathcal{I}n} \lambda_k^{\text{in}} \sigma_k^{\text{in}} = 0 \end{array} \right. \quad (5.29)$$

It was shown in the Appendix B of [171] that the closed-form solution to this system can be found by taking the ratio between λ_k^{in} and λ_l^{in} based on Eq. 5.29 and applying $-\frac{1}{\theta} \log$ on both sides, which gives

$$\lambda_k^{\text{in}} - \lambda_l^{\text{in}} = -\frac{1}{\theta} \left[\log \left(\frac{\sigma_k^{\text{in}}}{p_{1k}^{\text{ext}} z_{kn}^{\text{h}'(n)}} \right) - \log \left(\frac{\sigma_l^{\text{in}}}{p_{1l}^{\text{ext}} z_{ln}^{\text{h}'(n)}} \right) \right] \quad (5.30)$$

We now force the second equality constraint of Eq. 5.29 by multiplying both sides by σ_l^{in} and summing over l to obtain

$$\begin{aligned} \sum_{l \in \mathcal{I}n} \lambda_k^{\text{in}} \sigma_l^{\text{in}} - \sum_{l \in \mathcal{I}n} \lambda_l^{\text{in}} \sigma_l^{\text{in}} &= -\frac{1}{\theta} \left[\sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} \log \left(\frac{\sigma_k^{\text{in}}}{p_{1k}^{\text{ext}} z_{kn}^{\text{h}'(n)}} \right) \right. \\ &\quad \left. - \sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} \log \left(\frac{\sigma_l^{\text{in}}}{p_{1l}^{\text{ext}} z_{ln}^{\text{h}'(n)}} \right) \right] \end{aligned} \quad (5.31)$$

Given that $\sum_{l \in \mathcal{I}n} \lambda_l^{\text{in}} \sigma_l^{\text{in}} = 0$ and $\sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} = 1$, Eq. 5.31 becomes

$$\lambda_k^{\text{in}} = -\frac{1}{\theta} \left[\log \left(\frac{\sigma_k^{\text{in}}}{p_{1k}^{\text{ext}} z_{kn}^{\text{h}'(n)}} \right) - \sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} \log \left(\frac{\sigma_l^{\text{in}}}{p_{1l}^{\text{ext}} z_{ln}^{\text{h}'(n)}} \right) \right] \quad (5.32)$$

Moreover, we know that $p_{1l}^{\text{ext}} = \sigma_l^{\text{in}}$ for $l \in \mathcal{I}n$ (see Eq. 5.6), which provides

$$\begin{aligned} \lambda_k^{\text{in}} &= -\frac{1}{\theta} \left[\log \left(\frac{\sigma_k^{\text{in}}}{\sigma_k^{\text{in}} z_{kn}^{\text{h}'(n)}} \right) - \sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} \log \left(\frac{\sigma_l^{\text{in}}}{\sigma_l^{\text{in}} z_{ln}^{\text{h}'(n)}} \right) \right] \\ &= -\frac{1}{\theta} \left[\log \left(\frac{1}{z_{kn}^{\text{h}'(n)}} \right) - \sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} \log \left(\frac{1}{z_{ln}^{\text{h}'(n)}} \right) \right] \\ &= \frac{1}{\theta} \left(\log z_{kn}^{\text{h}'(n)} - \sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} \log z_{ln}^{\text{h}'(n)} \right) \end{aligned}$$

$$= \sum_{l \in \mathcal{In}} \sigma_l^{\text{in}} (\phi_l - \phi_k) \quad \text{for } k \in \mathcal{In} \quad (5.33)$$

where we used Eq. 3.37 and $\sum_{l \in \mathcal{In}} \sigma_l^{\text{in}} = 1$. It can be observed that the $\log z_{kn}^{\text{h}'(n)}$ is centered with respect to its weighted mean (with weights provided by σ_k^{in}).

Computing Lagrange parameters associated with output nodes. Symmetrically to Eq. 5.27, we have for the Lagrange parameters $\{\lambda_l^{\text{out}}\}_{l \in \mathcal{Out}}$ associated with the edges incident to the target node n ,

$$\bar{n}_{ln} = \frac{z_{1l}^{\text{h}'(n)} p_{ln}^{\text{ext}} \exp[-\theta c'_{ln}] z_{nn}^{\text{h}'(n)}}{z'_{1n}} = \frac{z_{1l}^{\text{h}'(n)} p_{ln}^{\text{ext}} \exp[-\theta \lambda_l^{\text{out}}]}{\sum_{l' \in \mathcal{Out}} z_{1l'}^{\text{h}'(n)} p_{l'n}^{\text{ext}} \exp[-\theta \lambda_{l'}^{\text{out}}]} \quad (5.34)$$

because $z_{nn}^{\text{h}'(n)} = 1$. As before, $\sum_{l=1}^n \bar{n}_{ln} = 1$ is verified. Setting $\bar{n}_{ln} = \sigma_l^{\text{out}}$ and proceeding in the same way as for the Lagrange parameters associated with input nodes provides, for the Lagrange parameters in $\mathcal{Out}$,

$$\lambda_l^{\text{out}} = \frac{1}{\theta} \left(\log z_{1l}^{\text{h}'(n)} - \log \left(\frac{\sigma_l^{\text{out}}}{p_{ln}^{\text{ext}}} \right) - \sum_{k \in \mathcal{Out}} \sigma_k^{\text{out}} \left[\log z_{1k}^{\text{h}'(n)} - \log \left(\frac{\sigma_k^{\text{out}}}{p_{kn}^{\text{ext}}} \right) \right] \right) \quad \text{for } l \in \mathcal{Out} \quad (5.35)$$

These Lagrange parameters, for both input (Eq. 5.33) and output nodes (Eq. 5.35), are recomputed in turn until convergence. They are then used in order to calculate the final augmented cost values and the corresponding fundamental matrix, allowing to obtain the quantities of interest, in particular, the optimal transition probabilities matrix (the policy).

5.3.3 Resulting algorithm

The resulting algorithm is presented in Algorithm 2. The different steps of the procedure are the following:

1. Compute the extended graph G_{ext} (its edge costs and transition probabilities matrices) from the original graph G as described in Section 5.2.3 and Eqs. 5.2, 5.3, and 5.6. We are now working on this extended graph.
2. Initialize the Lagrange parameters to 0 and the augmented costs to the edge costs of the extended graph $\mathbf{C}_{\text{ext}}$.

3. Iterate the following steps until convergence, first to update the quantities associated with the source nodes, and then to update the quantities associated with the target nodes:
 - The elements of the fundamental matrix (Eq. 3.22) are computed from the current augmented costs on G_{ext} .
 - The Lagrange parameters are updated (Eqs. 5.33 and 5.35).
 - The augmented costs are updated (Eq. 5.21).
4. Compute the optimal policy (transition probabilities) from the augmented costs (depending on the Lagrange parameters, see Eq. 5.21) on G_{ext} thanks to Eq. 4.4.

The time complexity of the algorithm is dominated by the two systems of linear equations that need to be solved at each iteration. Therefore, it is of order $2k \cdot \mathcal{O}(n^3)$ where n is the number of nodes and k is the number of required iterations. Note that Algorithm 2 is closely related to Algorithm 2 presented in [123] page 102. Let us now compare the computation time of these algorithms.

5.3.4 Computation time investigation

In this subsection, we evaluate the computational efficiency of Algorithm 2 (RSPOT) following the same experimental design as [123]. More specifically, we compare the computation time of Algorithm 2 to those of the two algorithms developed in [123], the margin-constrained bag-of-paths (cBoP) and margin-constrained bag-of-hitting-paths (cBoPH) algorithms, and also to those of two different linear solvers, Coin-or CLP [183] and CVX GLPK [278].

We run all the algorithms 30 times on lattices of increasing size ranging from 16 to 1000 nodes. For each run, we selected randomly one third of the nodes as source nodes and another third as target nodes. Moreover, the flow constraints associated with the source and target nodes are similar and uniform. Regarding the value of the parameter θ of the RSPOT, the cBoP, and the cBoPH algorithms, we have set its value to 10. We reported the evolution of the mean computation time over the 30 runs for each algorithm in Fig. 5.2. All results were obtained with Julia (version 1.7.0) running on Intel Core i7-8750H CPU@4.10GHz and 32 GB of RAM.

From Fig. 5.2, we can observe that the RSPOT, the cBoP, and the cBoPH perform much faster than the linear solvers CLP and CVX as the number of nodes increases. As already discussed in Section 5.1, Cuturi already observed that phenomenon with the Sinkhorn distance for entropy-regularized optimal transport problems [69, 221]. We can also see that the RSPOT is a bit slower than its counterpart in the bag-of-paths model, the cBoP, which has

Algorithm 2 Solving the relative entropy-regularized optimal transport on a graph problem with multiple sources and targets, called the margin-constrained bag-of-paths model.

Input:

- A weighted directed graph G_{ext} containing n nodes, derived from a strongly connected graph G as detailed in Section 5.2.3. Node 1 is the source supernode and node n the absorbing, target, supernode. The indegree of node 1 and the outdegree of node n are both equal to 0.
- The set of input nodes $\mathcal{In}$ (only connected to the source supernode 1) and output nodes $\mathcal{Out}$ (only connected to the target supernode n).
- The $n \times n$ transition matrix $\mathbf{P}_{\text{ext}}$ associated with G_{ext} (see Eq. 5.6).
- The $n \times n$ non-negative cost matrix $\mathbf{C}_{\text{ext}}$ associated with G_{ext} (see Eq. 5.3). These original costs are equal to zero for edges starting in node 1 and ending in $\mathcal{In}$ as well as edges starting in $\mathcal{Out}$ and ending in node n .
- The $n \times 1$ vectors of input flows, σ_{in} , and output flows, σ_{out} , both non-negative and summing to 1.
- The inverse temperature parameter $\theta > 0$.

Output:

- The $n \times n$ randomized policy defined by the transition matrix $\mathbf{P}^*$.
1. $\lambda_{\text{in}} \leftarrow \mathbf{0}; \lambda_{\text{out}} \leftarrow \mathbf{0}$ $\triangleright$ initialize $n \times 1$ Lagrange parameter vectors
 2. $\mathbf{C}' \leftarrow \mathbf{C}_{\text{ext}}$ $\triangleright$ initialize the augmented costs matrix
 3. **repeat** $\triangleright$ main iteration loop
 4. $\mathbf{W}'^{(n)} \leftarrow \mathbf{P}_{\text{ext}} \circ \exp[-\theta \mathbf{C}']$ $\triangleright$ compute $\mathbf{W}'^{(n)}$ matrix (see Eq. 3.5)
 5. Solve $(\mathbf{I} - \mathbf{W}'^{(n)}) \mathbf{z}_n^{\text{h}'(n)} = \mathbf{e}_n$ $\triangleright$ backward variables (column n of the fundamental matrix $\mathbf{Z}'^{(n)}_{\text{h}}$ defined in Eq. 3.27) with elements $z_{kn}^{\text{h}'(n)}$ (n is fixed)
 6. **for all** $k \in \mathcal{In}$ **do** $\triangleright$ compute Lagrange parameters associated with source nodes
 7. $\lambda_k^{\text{in}} \leftarrow \frac{1}{\theta} \log z_{kn}^{\text{h}'(n)}$ $\triangleright$ compute Lagrange parameters (see first part of Eq. 5.33)
 8. **end for**
 9. **for all** $k \in \mathcal{In}$ **do** $\triangleright$ update quantities associated with source nodes (see Eq. 5.21)
 10. $\lambda_k^{\text{in}} \leftarrow \lambda_k^{\text{in}} - \sum_{k' \in \mathcal{In}} \sigma_{k'}^{\text{in}} \lambda_{k'}^{\text{in}}$ $\triangleright$ normalize Lagrange parameters (see second part of Eq. 5.33)
 11. $c'_{1k} \leftarrow \lambda_k^{\text{in}}$ $\triangleright$ update augmented costs (recall that $c_{1k}^{\text{ext}} = 0$ for all $k \in \mathcal{In}$)
 12. **end for**
 13. $\mathbf{W}'^{(n)} \leftarrow \mathbf{P}_{\text{ext}} \circ \exp[-\theta \mathbf{C}']$ $\triangleright$ update $\mathbf{W}'^{(n)}$ matrix (see Eq. 3.5)
 14. Solve $(\mathbf{I} - \mathbf{W}'^{(n)})^T \mathbf{z}_1^{\text{h}'(n)} = \mathbf{e}_1$ $\triangleright$ forward variables (row 1 of the fundamental matrix $\mathbf{Z}'^{(n)}_{\text{h}}$ defined in Eq. 3.27) with elements $z_{1k}^{\text{h}'(n)}$ (1 is fixed)
 15. **for all** $l \in \mathcal{Out}$ **do** $\triangleright$ compute Lagrange parameters associated with target nodes
 16. $\lambda_l^{\text{out}} \leftarrow \frac{1}{\theta} \log z_{1l}^{\text{h}'(n)} - \frac{1}{\theta} \log \left(\frac{\sigma_l^{\text{out}}}{p_{ln}^{\text{ext}}} \right)$ $\triangleright$ compute Lagrange parameters (see first part of Eq. 5.35)
 17. **end for**
 18. **for all** $l \in \mathcal{Out}$ **do** $\triangleright$ update quantities associated with target nodes (see Eq. 5.21)
 19. $\lambda_l^{\text{out}} \leftarrow \lambda_l^{\text{out}} - \sum_{l' \in \mathcal{Out}} \sigma_{l'}^{\text{out}} \lambda_{l'}^{\text{out}}$ $\triangleright$ normalize Lagrange parameters (see second part of Eq. 5.35)
 20. $c'_{ln} \leftarrow \lambda_l^{\text{out}}$ $\triangleright$ update augmented costs (recall that $c_{ln}^{\text{ext}} = 0$ for all $l \in \mathcal{Out}$)
 21. **end for**
 22. **until** convergence of $\lambda_{\text{in}}, \lambda_{\text{out}}$
 23. $\mathbf{P}^* \leftarrow (\text{Diag}(\mathbf{z}_n^{\text{h}'(n)}))^{-1} \mathbf{W}'^{(n)} \text{Diag}(\mathbf{z}_n^{\text{h}'(n)})$ $\triangleright$ compute optimal policy (see Eq. 4.4)
 24. **return** $\mathbf{P}^*$
-

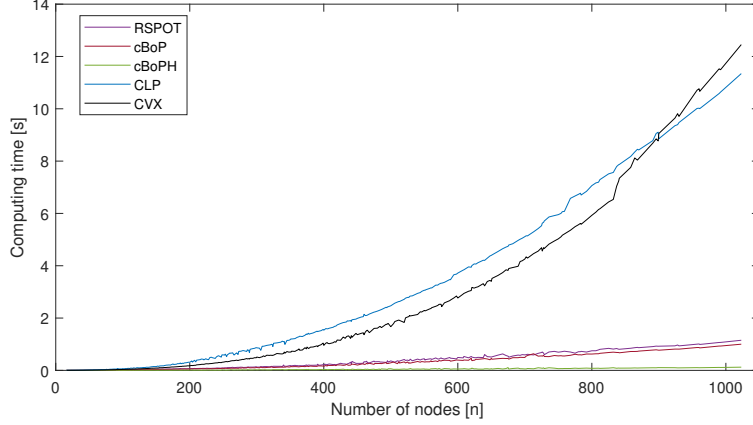


Figure 5.2: Evolution of the mean computation times of the RSPOT, the cBoP, the cBoPH, the Coin-or CLP, and the CVX GLPK on lattices of increasing size n , containing $\lfloor n/3 \rfloor$ sources and $\lfloor n/3 \rfloor$ targets with uniform weights. The linear solvers CLP and CVX are slower than all other methods.

already been pointed out in the introduction of this chapter. Furthermore, we can highlight that the cBoPH is the fastest on this experiment. Let us now introduce a dissimilarity measure based on the RSP optimal transport model.

5.3.5 A distance measure between nodes

In this subsection, we derive two important quantities from the margin-constrained bag-of-paths model. These quantities are the coupling matrix and the surprisal distance measure between nodes of the graph, defined from the coupling matrix. For more details regarding these two quantities, see [123].

5.3.5.1 Coupling matrix

Let us first recall the definition of the **coupling matrix** [282, 283], which was used in [123] to define a distance measure between the nodes of the graph (see Subsection 5.3.5.2).

The coupling matrix is denoted by $\mathbf{\Gamma} = (\gamma_{ij})$ and is defined as the joint probability of starting the walk in node $i \in \mathcal{In}$ (reaching input node $S = i$ from supernode 1 at time step 1) and ending the walk in node $j \in \mathcal{Out}$ (visiting output node $T = j$ at time step $\ell(\varphi) - 1$ and then immediately transiting to supernode n) when walking according to the optimal path probabilities defined in Eqs. 3.21 and 3.23 and using the augmented costs after convergence in order to satisfy the constraints,

$$\gamma_{ij} \triangleq \mathbb{P}^*(S = i, T = j) = \mathbb{P}^*(\varphi(1) = i, \varphi(\ell - 1) = j | \varphi(0) = 1, \varphi(\ell) = n)$$

$$\begin{aligned}
&= \frac{w'_{1i} \left(\sum_{\wp_{ij} \in \mathcal{P}_{ij}^h} \tilde{w}'^{(n)}(\wp_{ij}) \right) w'_{jn}}{\sum_{\wp' \in \mathcal{P}^h} \tilde{w}'^{(n)}(\wp')} = \frac{w'_{1i} z_{ij}^{h'} w'_{jn}}{\sum_{\wp' \in \mathcal{P}^h} \tilde{w}'^{(n)}(\wp')} \\
&= \frac{w'_{1i} z_{ij}^{h'} w'_{jn}}{\sum_{i' \in \mathcal{I}n} \sum_{j' \in \mathcal{O}ut} w'_{1i'} z_{i'j'}^{h'} w'_{j'n}} \quad \text{with } i \in \mathcal{I}n \text{ and } j \in \mathcal{O}ut \quad (5.36)
\end{aligned}$$

where $\mathcal{P}_{ij}^h$ is the set of hitting paths starting in input node $i \in \mathcal{I}n$ and ending in output node $j \in \mathcal{O}ut$, $\ell(\wp)$ is the length of path $\wp$, and $\wp(\tau)$ refers to the node at position τ on path $\wp$. It also holds that $\sum_{i \in \mathcal{I}n} \gamma_{ij} = \sigma_j^{\text{out}}$ and $\sum_{j \in \mathcal{O}ut} \gamma_{ij} = \sigma_i^{\text{in}}$. Notably, the elements $w'_{ij}^{(n)}$ are computed from the augmented costs after convergence of Algorithm 2. Further note that the rows and the columns associated with the two supernodes indexed as nodes 1 and n should be removed to obtain the coupling matrix associated with the original graph.

5.3.5.2 Margin-constrained bag-of-paths surprisal distance

We can now compute the **margin-constrained bag-of-paths surprisal distance** introduced in [123], itself inspired by [105], as

$$\Delta_{ij}^{\text{cBoP}} = \begin{cases} -\frac{1}{2}(\log(\gamma_{ij}) + \log(\gamma_{ji})) & \text{if } i \neq j, \\ 0 & \text{if } i = j \end{cases} \quad (5.37)$$

where γ_{ij} is the element i, j of the coupling matrix $\mathbf{\Gamma}$ defined in Eq. 5.36. This distance is a generalization of the surprisal distance [105, 156], presented in Subsection 3.4.1, where positive weights $\mathbf{v} = (v_i)$, summing to 1, can be attached to each node through σ_{in} and σ_{out} .

Intuitively, the distance (Eq. 5.37) quantifies the ‘surprise’ generated by the event $(S = i) \wedge (T = j)$, that is, sampling a path having input node $i \in \mathcal{I}n$ and output node $j \in \mathcal{O}ut$ from the bag of paths $\mathcal{P}_{1n}$ defined on G_{ext} , with probability distribution provided by Eq. 3.21 and using augmented costs in order to satisfy the constraints (Eq. 5.16).

In the experiments of this thesis, we consider that each node acts as both input and output, with $\sigma_{\text{in}} = \sigma_{\text{out}} = \mathbf{v}$. This choice is inspired from the well-known PageRank [41, 214] and the random walk with restart [113, 273] models in which, at each time step, the random walker has a chance of leaving the current node (which is then similar to a sink node) to restart in some nodes of the network (which are then similar to source nodes). This choice, although somewhat counter-intuitive⁶, provides quite competitive results on graph-based semi-supervised classification task as shown in Chapter 7.

⁶Because we inject and remove the same quantity of flow in each node, this setting is only relevant when the parameter θ is not too large. Indeed, when θ increases, the transportation

Note that the authors of [123] also propose a hitting path version of this distance which is not definable in our present model. This distance based on hitting paths is nevertheless also investigated in the experiments presented in Chapter 7.

5.4 Optimal transport on a graph from a constrained RSP model with capacity constraints

As already stated, the main advantage of rephrasing the entropy-regularized optimal transport on a graph problem within a source-target RSP formalism is that this new formulation can deal with both flow and capacity constraints. To this end, we need to combine the Algorithm 1 presented in Chapter 4 with the Algorithm 2 presented in Subsection 5.3.3.

From these algorithms, we can observe that the Lagrange parameters λ associated with the capacity constraint and the Lagrange parameters λ_{in} and λ_{out} associated with the input and output flow constraints never update the same elements of the augmented cost matrix C' . In fact, as explained in Section 4.2, the Lagrange parameters λ only update the set of capacity constrained edges $\mathcal{C}$ defined on the original graph G . Whereas, as described in the previous Section, the Lagrange parameters λ_{in} and λ_{out} , update respectively the edges starting in node 1 and ending in the set of input nodes $\mathcal{In}$ and the edges starting in the set of output nodes $\mathcal{Out}$ and ending in node n .

Therefore, to combine these algorithms, we can process as follows:

1. Compute the extended graph G_{ext} (its edge costs and transition probabilities matrices) from the original graph G as described in Section 5.2.3 and Eqs. 5.2, 5.3, and 5.6. We are now working on this extended graph.
2. Initialize the Lagrange parameters λ , λ_{in} and λ_{out} to 0 and the augmented costs to the edge costs of the extended graph C_{ext} .
3. Iterate the following steps until convergence of the λ associated with the capacity constraints:
 - Iterate the following steps until convergence, first to update the quantities associated with the source nodes, and then to update the quantities associated with the target nodes:
 - The elements of the fundamental matrix are computed from the current augmented costs Eq. 3.22 on G_{ext} .

is more and more optimal and the input-output flows tend to neutralize each other, resulting in a coupling matrix that converges to the identity matrix. This is, however, not the case for intermediate measures of θ for which the coupling matrix is able to capture the similarity between nodes (in terms of proximity and high connectivity).

- The Lagrange parameters λ_{in} and λ_{out} are updated (Eqs. 5.33 and 5.35).
 - The augmented costs are updated (Eq. 5.21).
 - The expected number of passages through each edge (edge flows) is computed (Eq. 4.2).
 - The Lagrange parameters λ and the augmented costs are updated (Eqs. 4.15 and 4.11).
4. Compute and return the optimal policy (transition probabilities) on G_{ext} according to Eq. 4.4.

The resulting algorithm is presented in Algorithm 3. As before, the time complexity of the algorithm is dominated by the two systems of linear equations that need to be solved at each iteration of the Algorithm 2 (second loop of Algorithm 3). Therefore, it is of order $2kl \cdot \mathcal{O}(n^3)$ where n is the number of nodes, k is the number of required iterations of the main loop and l is the number of required iterations of the second loop.

5.5 Case study: application to transportation problems

In this section, the algorithms introduced in Subsection 5.3.3 and Section 5.4 are used to solve some **transportation problems** (TPs) which are a specific case of the minimum cost flow problem introduced in Chapter 4. However, we have to stress that the goal of this experiment is to show the applicability of these algorithms on TP rather than demonstrate that our algorithms outperform state-of-the-art techniques of this domain.

The section is organized as follows. First, the theoretical background of TP is presented in Subsection 5.5.1. Then, Subsection 5.5.2 illustrates through an example how our algorithms can deal with TPs. Finally, Subsection 5.5.3 presents and discusses the results of the experimental comparisons of Algorithm 2 with state-of-the-art techniques on several TPs.

5.5.1 Theoretical background

In 1941, Hitchcock [130] introduced the concept of the linear programming (LP) transportation problem (TP). During the same period, in parallel to the work of Hitchcock, Koopmans [162] and Kantorovich [142] (himself inspired by the work of Monge in transportation theory [197]) proposed independent formulations of the same problem with different computational schemes [96]. Therefore, the TP is also known in the literature as Monge–Kantorovich or Hitchcock–Koopmans transportation problem [230, 283]. The interested reader can find additional information about the history of LP TP in [75, 107,

Algorithm 3 Solving the relative entropy-regularized optimal transport on a graph problem with multiple sources and targets and capacity constraints.

Input:

- A weighted directed graph G_{ext} containing n nodes, derived from a strongly connected graph G as detailed in Section 5.2.3. Node 1 is the source supernode and node n the absorbing, target, supernode.
- The set of input nodes $\mathcal{In}$ and output nodes $\mathcal{Out}$.
- The $n \times n$ transition matrix $\mathbf{P}_{\text{ext}}$ associated with G_{ext} (see Eq. 5.6).
- The $n \times n$ non-negative cost matrix $\mathbf{C}_{\text{ext}}$ associated with G_{ext} (see Eq. 5.3).
- The $n \times 1$ vectors of input flows, σ_{in} , and output flows, σ_{out} .
- The set of non-negative capacities on flows, $\{\sigma_{ij}\}$, defined on the set of constrained edges $\mathcal{C}$.
- The inverse temperature parameter $\theta > 0$.
- The gradient ascent step $\alpha > 0$.

Output:

- The $n \times n$ randomized policy defined by the transition matrix $\mathbf{P}^*$.

1. $\lambda_{\text{in}} \leftarrow \mathbf{0}; \lambda_{\text{out}} \leftarrow \mathbf{0}; \lambda \leftarrow \mathbf{0}$ $\triangleright$ initialize $n \times 1$ Lagrange parameter vectors
 2. $\mathbf{C}' \leftarrow \mathbf{C}_{\text{ext}}$ $\triangleright$ initialize the augmented costs matrix
 3. **repeat** $\triangleright$ main iteration loop
 4. **repeat** $\triangleright$ second iteration loop
 5. $\mathbf{W}'^{(n)} \leftarrow \mathbf{P}_{\text{ext}} \circ \exp[-\theta \mathbf{C}']$ $\triangleright$ compute $\mathbf{W}'^{(n)}$ matrix (see Eq. 3.5)
 6. Solve $(\mathbf{I} - \mathbf{W}'^{(n)})\mathbf{z}_n^{\text{h}'(n)} = \mathbf{e}_n$ $\triangleright$ backward variables (column n of the fundamental matrix $\mathbf{Z}_n^{\text{h}'(n)}$ defined in Eq. 3.27) with elements $z_{kn}^{\text{h}'(n)}$ (n is fixed)
 7. **for all** $k \in \mathcal{In}$ **do** $\triangleright$ compute Lagrange parameters associated with source nodes
 8. $\lambda_k^{\text{in}} \leftarrow \frac{1}{\theta} \log z_{kn}^{\text{h}'(n)}$ $\triangleright$ compute Lagrange parameters (see first part of Eq. 5.33)
 9. **end for**
 10. **for all** $k \in \mathcal{In}$ **do** $\triangleright$ update quantities associated with source nodes (see Eq. 5.21)
 11. $\lambda_k^{\text{in}} \leftarrow \lambda_k^{\text{in}} - \sum_{k' \in \mathcal{In}} \sigma_{k'}^{\text{in}} \lambda_{k'}^{\text{in}}$ $\triangleright$ normalize Lagrange parameters (see second part of Eq. 5.33)
 12. $c'_{1k} \leftarrow \lambda_k^{\text{in}}$ $\triangleright$ update augmented costs (recall that $c_{1k}^{\text{ext}} = 0$ for all $k \in \mathcal{In}$)
 13. **end for**
 14. $\mathbf{W}'^{(n)} \leftarrow \mathbf{P}_{\text{ext}} \circ \exp[-\theta \mathbf{C}']$ $\triangleright$ update $\mathbf{W}'^{(n)}$ matrix (see Eq. 3.5)
 15. Solve $(\mathbf{I} - \mathbf{W}'^{(n)})^T \mathbf{z}_1^{\text{h}'(n)} = \mathbf{e}_1$ $\triangleright$ forward variables (row 1 of the fundamental matrix $\mathbf{Z}_n^{\text{h}'(n)}$ defined in Eq. 3.27) with elements $z_{1k}^{\text{h}'(n)}$ (1 is fixed)
 16. **for all** $l \in \mathcal{Out}$ **do** $\triangleright$ compute Lagrange parameters associated with target nodes
 17. $\lambda_l^{\text{out}} \leftarrow \frac{1}{\theta} \log z_{1l}^{\text{h}'(n)} - \frac{1}{\theta} \log \left(\frac{\sigma_l^{\text{out}}}{p_{ln}^{\text{ext}}} \right)$ $\triangleright$ compute Lagrange parameters (see first part of Eq. 5.35)
 18. **end for**
 19. **for all** $l \in \mathcal{Out}$ **do** $\triangleright$ update quantities associated with target nodes (see Eq. 5.21)
 20. $\lambda_l^{\text{out}} \leftarrow \lambda_l^{\text{out}} - \sum_{l' \in \mathcal{Out}} \sigma_{l'}^{\text{out}} \lambda_{l'}^{\text{out}}$ $\triangleright$ normalize Lagrange parameters (see second part of Eq. 5.35)
 21. $c'_{ln} \leftarrow \lambda_l^{\text{out}}$ $\triangleright$ update augmented costs (recall that $c_{ln}^{\text{ext}} = 0$ for all $l \in \mathcal{Out}$)
 22. **end for**
 23. **until** convergence of $\lambda_{\text{in}}, \lambda_{\text{out}}$
 24. $\mathbf{N} \leftarrow \frac{\text{Diag}(\mathbf{z}_s^{\text{h}'(t)}) \mathbf{W}'^{(t)} \text{Diag}(\mathbf{z}_t^{\text{h}'(t)})}{z_{st}^{\text{h}'(t)}}$ $\triangleright$ compute the expected number of passages in each edge (see Eq. 4.2)
 25. **for all** $(i, j) \in \mathcal{C}$ **do** $\triangleright$ gradient ascent: update all quantities associated with constrained edges
 26. $\lambda_{ij} \leftarrow \max(\lambda_{ij} + \alpha(\bar{n}_{ij} - \sigma_{ij}), 0)$ $\triangleright$ update Lagrange parameters (see Eq. 4.15)
 27. $c'_{ij} \leftarrow c_{ij}^{\text{ext}} + \lambda_{ij}$ $\triangleright$ update augmented costs (see Eq. 4.11)
 28. **end for**
 29. **until** convergence of λ
 30. $\mathbf{P}^* \leftarrow (\text{Diag}(\mathbf{z}_n^{\text{h}'(n)}))^{-1} \mathbf{W}'^{(n)} \text{Diag}(\mathbf{z}_n^{\text{h}'(n)})$ $\triangleright$ compute optimal policy (see Eq. 4.4)
 31. **return** $\mathbf{P}^*$
-

	Destination 1	Destination 2	...	Destination l	Supply
Source 1	x_{11}	x_{12}	...	x_{1l}	s_1
Source 2	x_{21}	x_{22}	...	x_{2l}	s_2
...	...	...	...	...	...
Source k	x_{k1}	x_{k2}	...	x_{kl}	s_k
Demand	d_1	d_2	...	d_l	$\sum_{i=1}^k s_i = \sum_{j=1}^l d_j$

Table 5.1: Table representation of transportation problem.

230, 283]. Nowadays, TPs regroup a large number of sub-problems which are categorized by transportation mode such as military tanker fleet scheduling problem [96], air traffic flow management problem [127], truck loading problem [303] and so forth (see [83, 141] for details and the references in these papers).

According to [74, 130], a basic TP can be described as the shipping of a homogeneous product by a set $\mathcal{K}$ of k shipping sources with respective supply of $s_1, s_2, \dots, s_k$ to a set $\mathcal{L}$ of l shipping destinations with respective demand of $d_1, d_2, \dots, d_l$. Moreover, the transportation unit cost tc_{ij} from a source i to a destination j is known for all the combinations (i, j) of source-destination. The problem is to determine the optimal quantity of product x_{ij} to be shipped over all roads (i, j) minimizing the total transportation cost while satisfying the supply and the demand constraints. This TP can be represented by a transportation graph (see Fig. 5.3), a transportation table (see Table 5.1) or in mathematical optimization form as a linear programming problem:

$$\begin{array}{ll}
 \underset{x_{ij}}{\text{minimize}} & \sum_{i=1}^k \sum_{j=1}^l x_{ij} tc_{ij} \\
 \text{subject to} & \sum_{j=1}^l x_{ij} = s_i \quad \forall i \in \mathcal{K} \\
 & \sum_{i=1}^k x_{ij} = d_j \quad \forall j \in \mathcal{L} \\
 & x_{ij} \geq 0 \quad \forall i, j
 \end{array} \quad (5.38)$$

Note that the formulation of the problem presented in Eq. 5.38 assumes that the TP is balanced, which means that the total supply, $s_{\bullet} = \sum_{i=1}^k s_i$, and the total demand, $d_{\bullet} = \sum_{j=1}^l d_j$, are equal. This formulation is called **balanced transportation problem** (BTP).

However, the formulation can be easily adapted to take into account **unbalanced transportation problems** (UTP) [246], that is situations of excess supply, $s_{\bullet} > d_{\bullet}$, or excess demand, $d_{\bullet} > s_{\bullet}$. In the case of an excess supply, a dummy shipping destination $l + 1$ can be added to the set $\mathcal{L}$ in order to return to a balanced problem. The dummy shipping destination $l + 1$ has a demand of $d_{l+1} = s_{\bullet} - d_{\bullet}$ and a transportation unit cost tc_{il+1} from each shipping source i equal to zero. Similarly, in the case of an excess demand, a dummy shipping source $k + 1$ can be added to the set $\mathcal{K}$ to balance the prob-

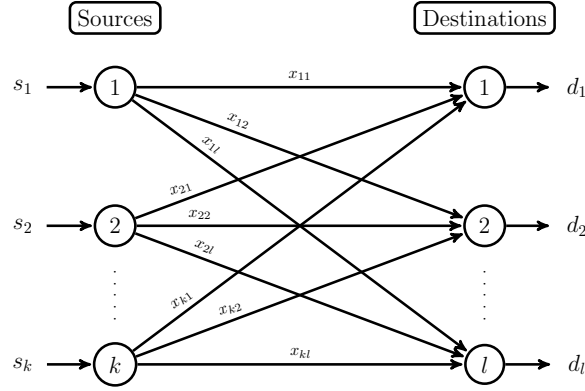


Figure 5.3: Network representation of transportation problem.

lem. The dummy shipping source $k + 1$ has a supply of $s_{k+1} = d_{\bullet} - s_{\bullet}$ and a transportation unit cost tc_{k+1j} to each shipping destination j equal to zero.

In 1951, Dantzig [74] developed an efficient method to solve TP by adapting the simplex method to this specific problem. The adaptation of Dantzig is called the primal simplex transportation method or the modified distribution method (MODI). Shortly after, Charnes and Cooper [49] proposed in 1953 an alternative solution which is the stepping stone method (SSM). These two methods are recognized as the foundation of the research in optimization of initial basic feasible solution (IBFS) in TP. Indeed, both the MODI and the SSM start from a sub-optimal IBFS and optimize them until reaching the minimal total transportation cost. Obviously, the time required to optimize the solution depends, among others, on the quality of the IBFS that is its proximity with the optimal solution. The literature on IBFS also emerged between the 1950s and the 1960s with methods such as the north-west corner method (NWC) [49], the Vogel's approximation method (VAM) [233], the row minimum method (RMM) [124], the column minimum method (CMM) [124], the least cost method (LCM) [75] etc. The development of these methods accompanied with illustrative examples is available in the following textbooks [129, 146, 264].

To sum up, a basic solution procedure for TP consists of the three following phases:

- **Phase 1:** Formulate mathematically the TP (see Eq. 5.38).
- **Phase 2:** Find an IBFS of the TP.
- **Phase 3:** Optimize the IBFS found in Phase 2.

Since the 1950s, several other methods have been developed to accelerate and optimize the search for an IBFS. Among these methods, we find modified

versions of the VAM [19, 115, 164, 251], standard deviation method [84], geometric mean method [44], multiple methods based on the total opportunity cost matrix [8, 153, 258] and many others [138, 145, 148, 192, 260]. Empirical comparisons of the most recent methods to find IBFS are available in [192, 258].

On the other hand, IBFS optimization research has also continued to grow with methods such as the tableau-dual method [12], the decomposition approach [294], the shortest path approach, the primal Hungarian method [199], the dualplex partitioning method [108], the dual matrix approach [222] etc.

5.5.2 Illustrative example

We now use a small example to illustrate how we can apply the algorithms introduced in Subsection 5.3.3 and Section 5.4 on TP.

Let us consider that a wholesaler is planning to allocate goods from three warehouses to three retailers. The manager of the company has prepared a transportation table (see Table 5.2) which provides transportation costs from each warehouse (shipping source) to each retailer (shipping destination).

	Retailer 1	Retailer 2	Retailer 3	Supply
Warehouse 1	6	8	10	150
Warehouse 2	7	11	11	175
Warehouse 3	4	5	12	275
Demand	200	100	300	600

Table 5.2: Table representation of a small example of transportation problem.

The first step to apply our algorithms to this example is to transform the transportation table into a undirected transportation graph. Then, we extend this transportation graph, as explained in Subsection 5.2.3, by adding one source node 1 connected to all the warehouses and one target node 8 connected to all the retailers. Finally, the flow constraints σ_{in} and σ_{out} should be defined. The input flow σ_i^{in} of each warehouse is fixed at their supply divided by the total supply ($\sigma_{\text{in}} = [0.25; 0.29; 0.46]$). Similarly, the output flow σ_j^{out} of each retailer is fixed at their demand divided by the total demand ($\sigma_{\text{out}} = [1/3; 1/6; 0.5]$).

The transportation solutions provided by the Algorithm 2 for several θ values are represented in Fig. 5.4. From Figs. 5.4f to 5.4j, we can observe that the optimal total transportation cost of 4525 is almost reached for all θ values larger than 1. However, from Figs. 5.4a to 5.4e, we can see that the total transportation cost moves further and further away from the optimal solution as θ approaches zero. As already mentioned in Section 3.3, when θ is

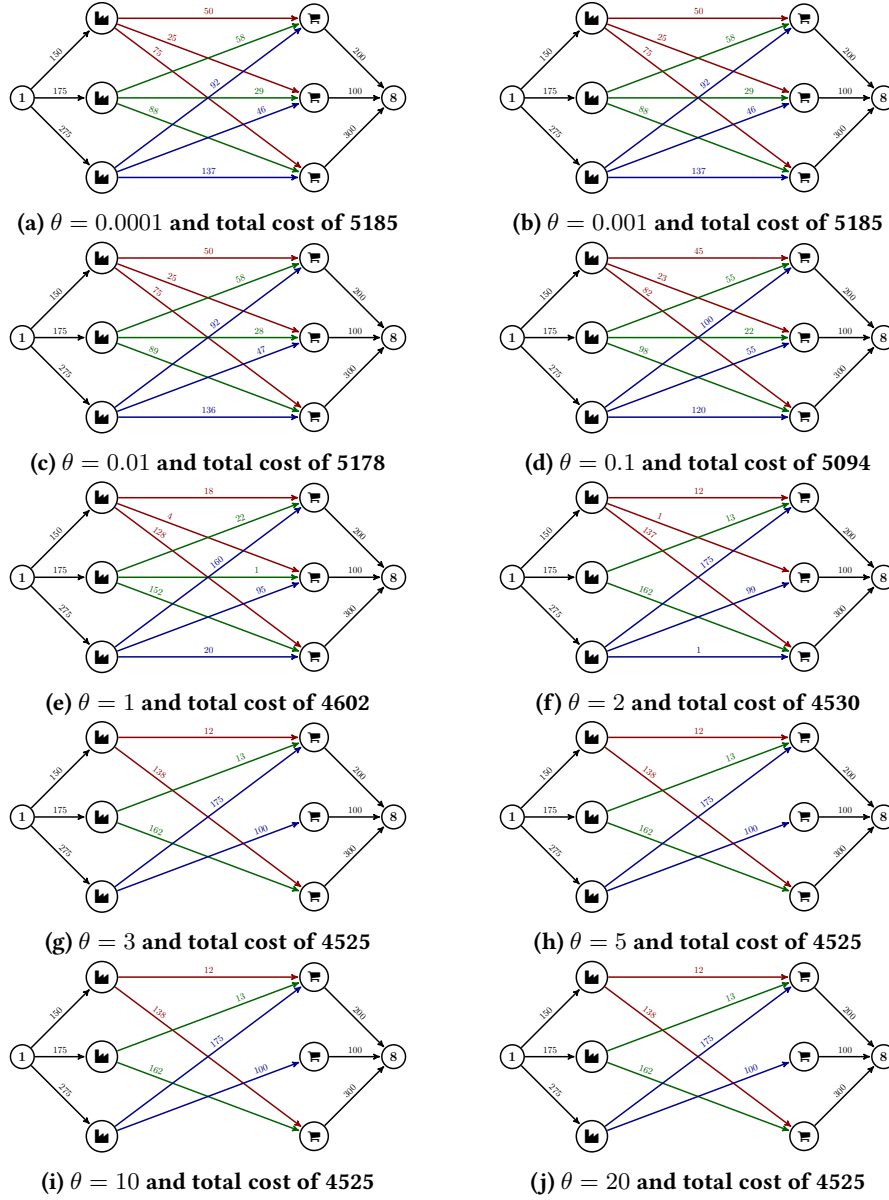


Figure 5.4: Representation of the solutions of the transportation problem presented in Table 5.2 provided by the Algorithm 2 for several θ values. The optimal total transportation cost of 4525 is reached for all $\theta > 2$.

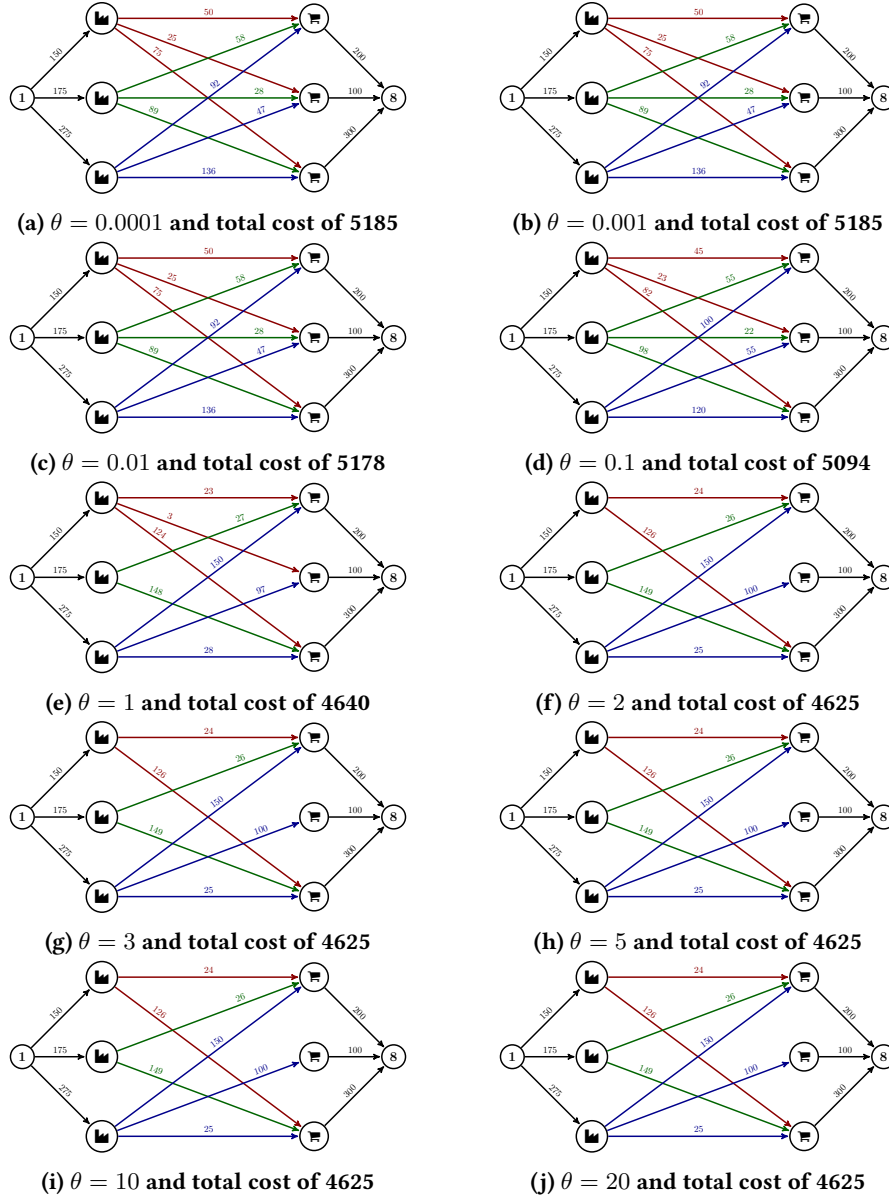


Figure 5.5: Representation of the solutions of the transportation problem presented in Table 5.2 with capacity constraint of 150 on each road provided by the Algorithm 3 for several θ values. The optimal total transportation cost of 4625 is reached for all $\theta \geq 2$.

low (close to zero), the paths are chosen according to their likelihood in the reference random walk, and therefore no longer consider costs (see Eq. 3.20). For that reason, some goods are allocated to sub-optimal roads in terms of cost, which increases the total transportation cost.

Let us now complicate the problem by adding a limitation of 150 to the total number of goods that can be transported on each road. This new constraint leads to an increase of the optimal solution from 4525 to 4625. The transportation solutions provided by the Algorithm 3 for several θ values are represented in Fig. 5.5. As expected, none of the transportation solutions changed for all θ below 1 because none of the allocations were greater than 150. From Figs. 5.5e to 5.5j, we can observe that the transportation solutions are well adapted to this new limitation of 150. As before, the optimal total transportation cost is reached for all θ larger than 1.

5.5.3 Empirical comparisons

We now compare the performance of Algorithm 2 (RSPOT) to the 5 IBFS methods introduced in Subsection 5.5.1 combined with the MODI and to the linear solver CVX [117]. The different methods is compared on 5 balanced transportation problems (BTP) and 5 unbalanced transportation problems (UTP) presented respectively in Tables 5.3 and 5.4.

We report the results of our comparison in Table 5.5. Moreover, we highlight the optimal total transportation costs in boldface for each TP. Note that we report only one column for the MODI because it gets the same results, the optimal solution, no matter which of the 5 IBFS methods is used. We also present the times in seconds required to solve the BTPs in Fig. 5.6 and UTPs in Fig. 5.7 for each method. All results were obtained with Matlab (version R2020a) running on Intel Core i7-8750H CPU@4.10GHz and 32 GB of RAM.

From Table 5.5, we can observe that the RSPOT reaches the optimal total transportation cost for all TPs like the MODI and the linear solver CVX. Regarding the IBFS methods, the VAM obtains the optimal solution on 5 TPs, the LCM on 2, and the RMM on 1. Overall, we can state that the VAM is the IBFS method that provides solutions closest to optimal or equivalent. The VAM is only outperformed on UTP3 and UTP5 by the CMM, and on BTP3 by the RMM that provides a solution one unit lower.

From Figs. 5.6 and 5.7, we can see that all the MODI combinations solve BTPs and UTPs faster than the linear solver CVX, except the MODI combined with the NWCM on BTP5. This is an expected result as the MODI is an adapted version of the simplex method for TPs. Regarding the RSPOT, it has comparable times to all the MODI combinations on all the TPs. Furthermore, as the MODI combinations, the RSPOT is faster than the linear solver CVX to solve each BTP and UTP.

	d_1	d_2	d_3	Supply
s_1	4	3	5	90
s_2	6	5	4	80
s_3	8	10	7	100
Demand	70	120	80	270

(a) Balanced transportation problem 1 (BTP1).

	d_1	d_2	d_3	d_4	Supply
s_1	4	6	9	5	16
s_2	2	6	4	1	12
s_3	5	7	2	9	15
Demand	12	14	9	8	43

(b) Balanced transportation problem 2 (BTP2).

	d_1	d_2	d_3	d_4	d_5	Supply
s_1	5	7	10	5	3	5
s_2	8	6	9	12	14	10
s_3	10	9	8	10	15	10
Demand	3	3	10	5	4	25

(c) Balanced transportation problem 3 (BTP3).

	d_1	d_2	d_3	d_4	d_5	d_6	Supply
s_1	12	4	13	18	9	2	120
s_2	9	16	10	7	15	11	80
s_3	4	9	10	8	9	7	50
s_4	9	3	12	6	4	5	90
s_5	7	11	15	18	2	7	100
s_6	16	8	4	5	1	10	60
Demand	75	85	140	40	95	65	500

(d) Balanced transportation problem 4 (BTP4).

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	Supply
s_1	12	7	3	8	10	6	6	60
s_2	6	9	7	12	8	12	4	80
s_3	10	12	8	4	9	9	3	70
s_4	8	5	11	6	7	9	3	100
s_5	7	6	8	11	9	5	6	90
Demand	20	30	40	70	60	80	100	400

(e) Balanced transportation problem 5 (BTP5).

Table 5.3: Five balanced transportation problems used in our comparisons.

	d_1	d_2	d_3	Supply
s_1	6	10	14	50
s_2	12	19	21	50
s_3	15	14	17	50
Demand	30	40	55	$\rightarrow 125/\downarrow 150$

(a) Unbalanced transportation problem 1 (UTP1) with an excess supply of 25.

	d_1	d_2	d_3	d_4	Supply
s_1	10	8	4	3	500
s_2	12	14	20	2	400
s_3	6	9	23	25	300
Demand	250	350	600	150	$\rightarrow 1350/\downarrow 1200$

(b) Unbalanced transportation problem 2 (UTP2) with an excess demand of 150.

	d_1	d_2	d_3	d_4	Supply
s_1	12	10	6	13	150
s_2	19	8	16	25	200
s_3	17	15	15	20	600
s_4	23	22	26	12	225
Demand	300	500	75	100	$\rightarrow 975/\downarrow 1175$

(c) Unbalanced transportation problem 3 (UTP3) with an excess supply of 200.

	d_1	d_2	d_3	d_4	d_4	Supply
s_1	5	8	6	6	3	800
s_2	4	7	7	6	5	500
s_3	8	4	6	6	4	900
Demand	400	400	500	400	800	$\rightarrow 2500/\downarrow 2200$

(d) Unbalanced transportation problem 4 (UTP4) with an excess demand of 300.

	d_1	d_2	d_3	d_4	d_5	Supply
s_1	5	4	8	6	5	600
s_2	4	5	4	3	2	400
s_3	3	6	5	8	4	1000
Demand	450	400	200	250	300	$\rightarrow 1600/\downarrow 2000$

(e) Unbalanced transportation problem 5 (UTP5) with an excess supply of 400.

Table 5.4: Five unbalanced transportation problems used in our comparisons.

Method: $\rightarrow$	NWCM	CMM	RMM	LCM	VAM	MODI	RSPOT	CVX
TP: $\downarrow$								
BTP1	1500	1500	1450	1450	1390	1390	1390	1390
BTP2	226	190	204	156	156	156	156	156
BTP3	234	215	186	191	187	183	183	183
BTP4	4285	2790	3090	2930	2325	2325	2325	2325
BTP5	3180	1940	1970	1900	1900	1900	1900	1900
UTP1	1815	1695	1925	1885	1745	1650	1650	1650
UTP2	18800	16900	11050	8800	8350	7750	7750	7750
UTP3	14725	12775	14625	14625	13225	12475	12475	12475
UTP4	13100	10300	9200	9800	9200	9200	9200	9200
UTP5	8150	5850	6450	6450	6000	5600	5600	5600

Table 5.5: Total transportation cost for the various methods obtained on the different transportation problems presented in Tables 5.3 and 5.4. The optimal total transportation costs are highlighted in boldface for each transportation problem. The RSPOT, the MODI, and the linear solver CVX are reaching the optimal total transportation costs for all TPs.

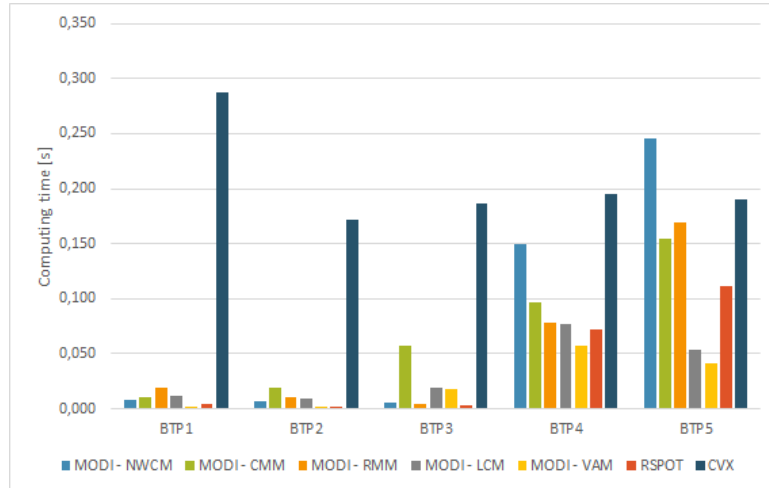


Figure 5.6: Computation time required to solve the BTPs for each method. The linear solver CVX is the slowest for all BTPs, except for BTP5 where it is the MODI - NWCM.

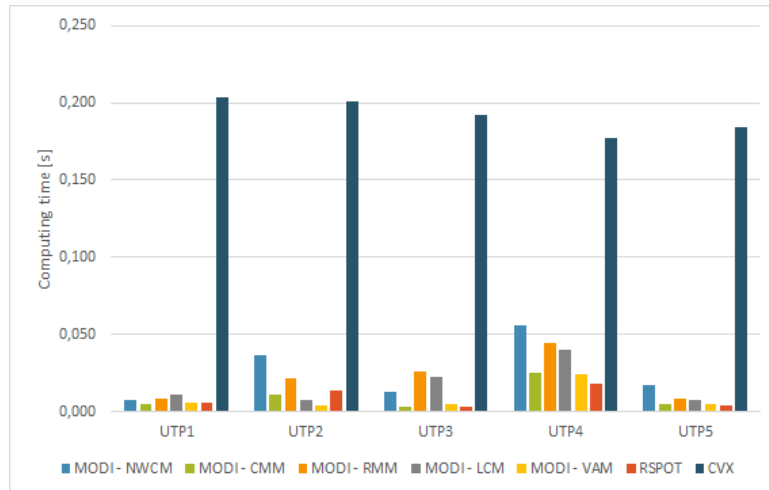


Figure 5.7: Computation time required to solve the UTPs for each method. The linear solver CVX is the slowest for all UTPs.

5.6 Conclusion

In this chapter, we extended the randomized shortest path (RSP) model by introducing two new algorithms solving the relative entropy-regularized minimum expected cost flow problem on a graph with and without capacity constraints. As in the previous chapter, these algorithms exploit both Lagrange duality and closed-form matrix expressions to compute the quantities of interest defined in the RSP model.

This new formulation of the problem in the RSP model extends the previous ones [120, 123] developed in the bag-of-paths model by integrating flow capacity constraints, which frequently appear in real-world applications [3].

Therefore, our model expands the scope of the previous models to a wider range of minimum cost problems such as transportation problems. The transportation problems include a wide variety of real-world problems which are categorized by transportation mode such as ship routing problem [150], helicopter routing problem [95], capacitated vehicle routing problem [274] and so forth. A comprehensive list of these problems can be found in [83, 141].

Through this chapter, we also showed how our algorithms can adapt to balanced and unbalanced transportation problems with and without capacity constraints. The experimental comparisons indicated that our first algorithm (Algorithm 2) is competitive with the other investigated methods. Moreover, the experiments highlighted that our algorithms are faster than standard linear solvers as Coin-or CLP [183] and CVX [117] to solve such problems, at least on the small investigated problems.

Moreover, the scalability experiment indicated that our algorithm without capacity constraints (Algorithm 2) is a bit slower than its counterpart in the bag-of-paths model proposed in [123], the margin-constrained bag-of-paths algorithm. On its side, the margin-constrained bag-of-hitting-paths algorithm also developed in [123] is clearly the fastest of the three algorithms. Therefore, as already mentioned in the introduction, our algorithm should only be used when capacity constraints are actually present.

Future work will aim to develop our model to incorporate both models developed in [123] and not just the margin-constrained bag-of-paths model. The integration of the margin-constrained bag-of-hitting-paths model would allow us to define the margin-constrained bag-of-hitting-paths surprisal distance [123] in our model.

Another interesting track would be to make the link between the present approach and electrical current in the case of an undirected graph [86].

Still another considered future work would be to apply our algorithms to solve larger real-world problems. Nevertheless, we should first investigate a way to reduce the complexity of our algorithms. Their current complexity of $\mathcal{O}(n^3)$ does not allow us to use them on graphs of several millions of nodes.

Bag-of-paths with path lengths weighted by a Poisson distribution

6

In this chapter, we investigate two extensions of the bag-of-paths model (BoP) presented in Section 3.1. In the first extension, we propose to weight the paths by a Poisson probability distribution on their length assuming the independence between path length and path likelihood. Interestingly, this simple model leads to a similarity measure between nodes which is closely related to the log-communicability measure (see [92, 134] for the communicability and the log-communicability measures), and thus provides a new interpretation of this quantity. In the second extension, we constrain the probability of sampling a path to be Poisson-distributed, with respect to its length. The present chapter is therefore part of a desire to broaden the scope of the bag-of-paths model.

Moreover, the main advantage of these new extensions is that the probability of a path can be expressed by explicitly taking into account the length $\ell(\varphi)$ of this path. This change in the expression of the probability can be considered as a way of penalizing longer paths in the network and tuning the mean length depending on the application. In this context, path lengths are weighted or constrained by a Poisson distribution whose parameter is the mean path length. From these models, two new distance measures between nodes of the graph are introduced the **Poisson weighted surprisal distance** and the **Poisson surprisal distance**, inspired by their BoP counterpart (see Subsection 3.4.1).

The main motivation of this chapter is therefore to extend the BoP model by proposing two ways to explicitly take into account the length $\ell(\varphi)$ of the path in the computation of its probability in order to allow the penalization of longer paths and to tune the mean length depending on the application. In short, the main contributions of this chapter are:

- The introduction of a Poisson probability distribution weighting path lengths, assuming independence between path length and path likelihood, in the bag-of-paths model;
- The introduction of a Poisson probability distribution constraining the

probability of sampling each path length in the bag-of-paths model;

- The definition of two new distance measures between nodes of a graph inside these extended models called the Poisson weighted surprisal distance and the Poisson surprisal distance;
- A new interpretation of the logarithmic communicability similarity measure introduced by Ivashkin and Chebotarev [134].

The chapter is organized as follows. First, Section 6.1 presents some necessary notations and quantities of interest needed to derive these extensions. Second, Section 6.2 introduces the first extension of the BoP model weighting the paths by a Poisson probability distribution on their length. A new distance measure between nodes of a graph is derived from this model in the same section. Then, Section 6.3 develops the second extension of the BoP constraining the probability of sampling a path length to be Poisson-distributed and the new distance measure derived from this model. Finally, Section 6.4 concludes the chapter.

6.1 Basic notions

As already discussed, the main contributions of this chapter are based on the BoP model introduced in Section 3.1. This section aims to recall some necessary notation and introduce some quantities of interest obtained from the BoP model and the randomized shortest path model (RSP) presented in Chapter 3, which are useful to derive the extension presented in Section 6.2.

Let us consider a weighted, strongly connected, directed¹, graph G containing n nodes $\in \mathcal{V}$ (the set of nodes), and non-negative costs c_{ij} together with affinities a_{ij} (adjacency matrix) associated with (directed) edges (see Subsections 2.1.1, 2.2.1, and 2.2.2 for more details on these concepts).

Recall from Section 3.3 that the RSP probability [156, 238, 300] of observing a path $\wp_{st}$ from *source* node s to *target* node t of length $\ell(\wp_{st})$ (or simply ℓ if there is no ambiguity), and thus node sequence $(i_0 = s, i_1, i_2, \dots, i_{\ell-1}, i_\ell = t)$ with i_τ being the index of the node at position τ in the sequence, is provided by a Gibbs-Boltzmann distribution

$$P^{\text{RSP}}(\wp_{st}) = \frac{\tilde{\pi}(\wp_{st}) \exp[-\theta \tilde{c}(\wp_{st})]}{\sum_{\wp' \in \mathcal{P}_{st}} \tilde{\pi}(\wp'_{st}) \exp[-\theta \tilde{c}(\wp'_{st})]} \quad (6.1)$$

where $\tilde{c}(\wp_{st})$ is the total cost of path $\wp_{st}$ when visiting the nodes in the sequential order and $\tilde{\pi}(\wp_{st})$ is the likelihood of the path $\wp_{st}$ according to the

¹If the graph G is undirected, each of its undirected edge $i \leftrightarrow j$ is considered as a set of two different directed edges going in opposite directions, $i \rightarrow j$ and $j \rightarrow i$.

natural random walk (see Section 3.1 for further details). In this expression, $\mathcal{P}_{st}$ is the set of all regular, non-hitting, non-absorbing, paths from node s to node t . Moreover, for convenience, let us denote by $\wp_{st}(\tau)$ the index of the node at position $\tau \in [0, \ell]$ on path $\wp_{st}$. In our sequence defined above, it corresponds to $\wp_{st}(\tau) = i_\tau$.

Using Eq. 3.5 that introduces the transition probability matrix $\mathbf{W}$, we have $w_{ij} = p_{ij} \exp[-\theta c_{ij}]$ and the numerator of Eq. 6.1 can be rewritten as

$$\begin{aligned} \tilde{\pi}(\wp_{st}) \exp[-\theta c(\wp_{st})] &= \prod_{\tau=1}^{\ell(\wp_{st})} (p_{i_{(\tau-1)}i_\tau} \exp[-\theta c_{i_{(\tau-1)}i_\tau}]) = \prod_{\tau=1}^{\ell(\wp_{st})} w_{i_{(\tau-1)}i_\tau} \\ &= w_{si_1} w_{i_1 i_2} \cdots w_{i_{(\ell-2)}i_{(\ell-1)}} w_{i_{(\ell-1)}t} \triangleq w(\wp_{st}) \end{aligned} \quad (6.2)$$

where p_{ij} is the element of the stochastic transition probabilities matrix of the reference random walk on the graph $\mathbf{P}$ (see Subsection 2.2.4).

As already mentioned in Section 3.1, the matrix $\mathbf{W}$ is sub-stochastic, meaning that all its row sums are lesser than or equal to 1 and there is at least one row sum strictly lesser than 1. Thus, this matrix represents a transition probability matrix between transient states, with a probability of giving up the walk in each node i provided by $1 - \sum_{j=1}^n w_{ij}$. We can then say that the random walker is killed in i .

In this context, Eq. 6.2 represents the likelihood of reaching (being killed in) the target node t through path $\wp_{st}$ when walking according to the sub-stochastic transition matrix $\mathbf{W}$ defining a killed random walk and starting from node s .

In a more general setting, let us now consider a bag of paths containing the set of all paths in the network $\mathcal{P} = \cup_{i,j=1}^n \mathcal{P}_{ij}$ as defined in Section 3.1. Inspired from the RSP (see Eq. 6.2), in its basic form, the BoP model assumes that the probability of drawing a specific path $\wp \in \mathcal{P}$ starting from some node i and reaching node j is *proportional to the likelihood of that path* [105],

$$\begin{aligned} \mathbf{P}^{\text{BoP}}(\wp \wedge \wp(0) = i \wedge \wp(\ell) = j) &\propto w_{i_0 i_1} w_{i_1 i_2} \cdots w_{i_{(\ell-2)}i_{(\ell-1)}} w_{i_{(\ell-1)}i_\ell} \delta_{i_0 i} \delta_{i_\ell j} \\ &= \delta(\wp(0) = i) \delta(\wp(\ell) = j) w(\wp) \end{aligned} \quad (6.3)$$

where $\delta(\wp(0) = i) = 1$ when the initial node of the path $\wp$ is i , and 0 otherwise.

In addition, we can also introduce *a priori weighting probabilities* of starting (in) in the node i and ending (out) in the node j [105]. In that situation, $\delta(\wp(0) = i) \delta(\wp(\ell) = j)$ is replaced by the product of the start and end probabilities, $\sigma_i^{\text{in}} \delta(\wp(0) = i) \sigma_j^{\text{out}} \delta(\wp(\ell) = j)$ with $\sum_{i=1}^n \sigma_i^{\text{in}} = \sum_{j=1}^n \sigma_j^{\text{out}} = 1$, providing

$$\mathbf{P}^{\text{BoP}}(\wp \wedge \wp(0) = i \wedge \wp(\ell) = j) \propto \sigma_i^{\text{in}} \delta(\wp(0) = i) \sigma_j^{\text{out}} \delta(\wp(\ell) = j) w(\wp) \quad (6.4)$$

Note that this model does not imply $P_{\text{BOP}}(\wp(0) = i) = \sigma_i^{\text{in}}$ nor $P_{\text{BOP}}(\wp(\ell) = j) = \sigma_j^{\text{out}}$ – for such a model, see [123] and Chapter 5.

In order to be well-defined, the quantity defined in Eq. 6.4 has to be properly normalized, $\sum_{\wp \in \mathcal{P}} \sum_{i,j \in \mathcal{V}} P^{\text{BOP}}(\wp \wedge \wp(0) = i \wedge \wp(\ell) = j) = 1$, which implies that the normalizing factor is $1 / \sum_{i,j=1}^n \sum_{\wp \in \mathcal{P}_{ij}} \sigma_i^{\text{in}} \sigma_j^{\text{out}} w(\wp)$, and thus

$$P^{\text{BOP}}(\wp \wedge \wp(0) = i \wedge \wp(\ell) = j) = \begin{cases} \frac{\sigma_i^{\text{in}} \sigma_j^{\text{out}} w(\wp)}{\sum_{i',j'=1}^n \sum_{\wp' \in \mathcal{P}_{i'j'}} \sigma_{i'}^{\text{in}} \sigma_{j'}^{\text{out}} w(\wp')} & \text{if } \wp \in \mathcal{P}_{ij} \\ 0 & \text{if } \wp \notin \mathcal{P}_{ij} \end{cases} \quad (6.5)$$

6.2 Bag-of-paths weighted by path lengths

In this section, we present the first main contribution of the chapter that introduces a probability mass weighting the length of the paths. For the sake of simplicity, we start without considering priors on the start and end nodes. First, we introduce the likelihood of a path including a Poisson distribution assigned to its path length in Subsection 6.2.1. Then, the joint probability of drawing a path starting in node i and ending in j , $P^{\text{WBOP}}(\wp(0) = i \wedge \wp(\ell) = j)$, is derived in Subsection 6.2.2. This is an important measure quantifying how close and how highly inter-connected the two nodes are. In addition, a straightforward distance measure between nodes is derived from these probabilities in Subsection 6.2.3. Finally, the results are easily extended to models weighted with probabilities on start and end nodes in Subsection 6.2.4.

6.2.1 A Poisson distribution on path lengths

Let us now consider that the length $\ell(\wp)$ of each path $\wp$ is weighted according to a Poisson distribution in such a way that the likelihood of a path is modulated by the probability of its length. More specifically, from Eq. 6.3, the likelihood of drawing a path $i \rightsquigarrow j$ of a given length τ is assumed to be proportional to

$$\begin{aligned} P^{\text{WBOP}}(\wp \wedge \wp(0) = i \wedge \wp(\ell) = j \wedge \ell(\wp) = \tau) \\ \propto \delta(\wp(0) = i) \delta(\wp(\ell) = j) \delta(\ell(\wp) = \tau) f(\tau, \lambda) w(\wp) \end{aligned} \quad (6.6)$$

with $f(\cdot)$ being the Poisson distribution [225, 285], providing the probability mass of drawing a path of length τ with parameter (mean length) λ ,

$$f(\tau, \lambda) = \frac{\lambda^\tau \exp(-\lambda)}{\tau!} \quad (6.7)$$

As before, Eq. 6.6 should be properly normalized in order to sum to 1 to define a probability distribution on the set of all paths $\mathcal{P}$. However, we leave the normalization for later. According to this model, the probability of drawing a path $\wp$ is thus proportional to

$$\begin{aligned} P^{\text{WBOP}}(\wp) &\propto \sum_{i,j \in \mathcal{V}} \sum_{\tau=0}^{\infty} \delta(\wp(0) = i) \delta(\wp(\ell) = j) \delta(\ell(\wp) = \tau) f(\tau, \lambda) w(\wp) \\ &= f(\ell(\wp), \lambda) w(\wp) \end{aligned} \quad (6.8)$$

which is the model investigated in this section. This choice is mainly motivated by the fact that (i) the model integrates a ‘scale parameter’ λ modulating the region of influence in the network based on path lengths, and (ii) the resulting quantities of interest can be computed in closed-form and are related to a previous similarity measure.

However, this model does not imply $P^{\text{WBOP}}(\ell(\wp) = \tau) = f(\tau, \lambda)$ because the length of a path is also affected by the path cost and the reference random walk probabilities. This is due to the fact that this simple model assumes the independence² between path likelihood and path length, which is not the case in general. This assumption is alleviated later in Section 6.3.

6.2.2 Computing the joint probability of drawing a path starting in i and ending in j

Let us now compute the interesting quantity $\pi_{ij} \triangleq P^{\text{WBOP}}(\wp(0) = i \wedge \wp(\ell) = j)$, that is, the joint probability of observing a path starting in i and ending in j when drawing paths from the bag of paths $\mathcal{P}$. A similar measure was derived in [105] and used in [80] in order to develop a paths-based modularity measure, but without any weighting by path length. From Eq. 6.6, we find

$$\begin{aligned} P^{\text{WBOP}}(\wp(0) = i \wedge \wp(\ell) = j) &\propto \sum_{\wp \in \mathcal{P}} \sum_{\tau=0}^{\infty} \delta(\wp(0) = i) \delta(\wp(\ell) = j) \delta(\ell(\wp) = \tau) f(\tau, \lambda) w(\wp) \\ &= \sum_{\wp \in \mathcal{P}} \delta(\wp(0) = i) \delta(\wp(\ell) = j) f(\ell(\wp), \lambda) w(\wp) \\ &= \sum_{\wp \in \mathcal{P}_{ij}} f(\ell(\wp), \lambda) w(\wp) = \sum_{\tau=0}^{\infty} \sum_{\wp \in \mathcal{P}_{ij}^{(\tau)}} f(\tau, \lambda) w(\wp) \\ &= \sum_{\tau=0}^{\infty} f(\tau, \lambda) \sum_{\wp \in \mathcal{P}_{ij}^{(\tau)}} w(\wp) = \sum_{\tau=0}^{\infty} f(\tau, \lambda) [\mathbf{W}^{\tau}]_{ij} \end{aligned}$$

²As, for instance, a simple bag-of-words model in information retrieval or a naive Bayes classification model in data science.

$$= \exp(-\lambda) \left[\sum_{\tau=0}^{\infty} \frac{\lambda^{\tau} \mathbf{W}^{\tau}}{\tau!} \right]_{ij} \propto [\text{expm}(\lambda \mathbf{W})]_{ij} \quad (6.9)$$

where expm is the matrix exponential and $\mathcal{P}_{ij}^{(\tau)}$ is the set of all paths connecting node i to node j in *exactly* τ steps. Recall also from Eq. 3.6 that $\sum_{\wp \in \mathcal{P}_{ij}^{(\tau)}} w(\wp) = [\mathbf{W}^{\tau}]_{ij}$ in the standard BoP model [105].

From Eq. 6.9, we easily deduce, after normalization,

$$\pi_{ij}(\lambda) = \mathbf{P}^{\text{WBOP}}(\wp(0) = i \wedge \wp(\ell) = j) = \frac{[\text{expm}(\lambda \mathbf{W})]_{ij}}{\sum_{i', j'=1}^n [\text{expm}(\lambda \mathbf{W})]_{i' j'}} \quad (6.10)$$

These probabilities quantify to which extent the two nodes i and j are easily accessible in the network by taking into account both the proximity in the network (the two nodes are close) and their amount of inter-connectivity (many paths connecting the two nodes). This quantity is closely related to the *communicability measure* developed in [92]. In matrix form, we have

$$\mathbf{\Pi}(\lambda) = \frac{\text{expm}(\lambda \mathbf{W})}{\mathbf{e}^T \text{expm}(\lambda \mathbf{W}) \mathbf{e}} \quad (6.11)$$

where $\mathbf{e}$ is a column vector full of 1's.

The time complexity of the matrix $\mathbf{\Pi}(\lambda)$ computation is dominated by the matrix exponential. Therefore, it is of order $\mathcal{O}(n^3)$ where n is the number of nodes [110].

6.2.3 A derived distance measure between nodes

A common procedure to derive a dissimilarity from a probability mass is to take minus the logarithm of this probability. The resulting distance matrix between nodes, called the directed³ *Poisson weighted surprisal distance*, is consequently defined as

$$\Delta^{\text{DPWSURP}} = -\log \mathbf{\Pi}(\lambda) \quad (6.12)$$

with the $\log$ being the natural elementwise logarithm. It computes the *surprisal* of observing a path starting in i and ending in j , and is an extension of the surprisal distance (see Subsection 3.4.1) introduced in [105] in the standard BoP framework. The resulting distance measure is the symmetrized quantity,

$$\Delta^{\text{PWSURP}} = \frac{1}{2} \left(\Delta^{\text{DPWSURP}} + (\Delta^{\text{DPWSURP}})^T \right). \quad (6.13)$$

where the diagonal elements are then set to zero by subtracting the $\text{Diag}(\Delta^{\text{PWSURP}})$ to Δ^{PWSURP} .

³In general, this matrix is non-symmetric.

This new distance is closely related to the *log-communicability measure* introduced in [134]. The log-communicability measure is the logarithmic version of the well-known communicability measure [92], or of the exponential diffusion kernel [160], and is defined as

$$\mathbf{K} = \log(\expm(\alpha \mathbf{A})) \quad (6.14)$$

where $\mathbf{A}$ is the adjacency matrix of G (see Subsection 2.2.1) and α is a discounting parameter favoring shorter paths by giving them a larger weight.

The main difference between these two measures is that the matrix exponential is taken on the adjacency matrix $\mathbf{A}$ instead of the $\mathbf{W}$ matrix in Eqs. 6.11 and 6.12. This observation therefore provides an interpretation to the log-communicability measure in the BoP framework, where the parameter α corresponds to the parameter λ . These two measures are compared on a graph-based semi-supervised classification task in Chapter 7.

6.2.4 Weighting start and end nodes

Considering now prior weighting factors $\sigma_i^{\text{in}}, \sigma_j^{\text{out}}$ (which are assumed to be known) on the start and end nodes and applying the same reasoning as in previous section (see Eq. 6.9) provides

$$\begin{aligned} & P^{\text{WBOP}}(\wp(0) = i \wedge \wp(\ell) = j) \\ & \propto \sum_{\wp \in \mathcal{P}} \sum_{\tau=0}^{\infty} \sigma_i^{\text{in}} \delta(\wp(0) = i) \sigma_j^{\text{out}} \delta(\wp(\ell) = j) \delta(\ell(\wp) = \tau) f(\tau, \lambda) w(\wp) \\ & = \sigma_i^{\text{in}} \sigma_j^{\text{out}} \sum_{\wp \in \mathcal{P}_{ij}} f(\ell(\wp), \lambda) w(\wp) \propto \sigma_i^{\text{in}} \sigma_j^{\text{out}} [\expm(\lambda \mathbf{W})]_{ij} \end{aligned} \quad (6.15)$$

In matrix form, this leads to

$$\mathbf{\Pi}(\lambda) = \frac{\mathbf{Diag}(\boldsymbol{\sigma}_{\text{in}}) \expm(\lambda \mathbf{W}) \mathbf{Diag}(\boldsymbol{\sigma}_{\text{out}})}{\boldsymbol{\sigma}_{\text{in}}^T \expm(\lambda \mathbf{W}) \boldsymbol{\sigma}_{\text{out}}} \quad (6.16)$$

where $\boldsymbol{\sigma}_{\text{in}}$ and $\boldsymbol{\sigma}_{\text{out}}$ are column vectors containing the weights on the nodes and $\mathbf{Diag}$ is an operator which, when applied to a vector, defines a diagonal matrix with this vector as diagonal. The distance is defined the same way as before through Eqs. 6.12 and 6.13. We refer to this distance as the *Poisson weighted surprisal distance with priors*. Note that when the probabilities are uniform, $\boldsymbol{\sigma} = \mathbf{e}/n$ where n is the number of nodes, the distance is equivalent (up to a scaling factor) to the *Poisson weighted surprisal distance* defined in the previous subsection. In addition, the time complexity of the matrix $\mathbf{\Pi}(\lambda)$ computation remains $\mathcal{O}(n^3)$.

As the previous measures, the performance of the Poisson weighted surprisal distance with priors are compared in terms of classification accuracy

on a graph-based semi-supervised classification task in Chapter 7. In this chapter, we consider that the priors on start and end nodes are equal, $\sigma_{\text{in}} = \sigma_{\text{out}} = \sigma$.

6.3 Bag-of-paths with Poisson-distributed path lengths

In this section, we arrive to the second main contribution of the chapter that constraints the probability of sampling a path to follow a Poisson probability distribution. This second contribution extends the previous one by avoiding independence between path likelihood and path length. First, we introduce the BoP model constraining the probability of sampling a path to be Poisson-distributed in Subsection 6.3.1. Then, the joint probability of drawing a path starting in node i and ending in j is derived in Subsection 6.3.2. Finally, a distance measure between nodes is derived from these probabilities in Subsection 6.3.3.

6.3.1 BoP model with Poisson-distributed path lengths

As in the standard BoP model introduced in Section 3.1, we minimize the free energy objective function (see Eq. 3.20) while satisfying some constraints on path lengths. The idea is to constrain the probability of sampling a path $\wp$ of length $\tau = \ell(\wp)$ to follow a Poisson probability distribution $f(\tau, \lambda)$ with parameter λ (see Eq. 6.7). This additional constraint allows to tune the optimal path length at which the relevant information can be found. The problem therefore becomes

$$\left| \begin{array}{ll} \text{minimize} & \sum_{s,t=1}^n \sum_{\tau=0}^{\infty} \sum_{\wp \in \mathcal{P}_{st}(\tau)} \left(P(\wp) \tilde{c}(\wp) + T P(\wp) \log \left(\frac{P(\wp)}{\tilde{\pi}(\wp)} \right) \right) \\ \text{subject to} & \sum_{s,t=1}^n \sum_{\wp \in \mathcal{P}_{st}(\tau)} P(\wp) = f(\tau, \lambda) \quad \text{for each } \tau \end{array} \right. \quad (6.17)$$

where $\tilde{c}(\wp)$ is the total cost of path $\wp$ when visiting the nodes in the sequential order and $\tilde{\pi}(\wp)$ is the likelihood of the path $\wp$ according to the natural random walk (see Section 3.1 for further details). Furthermore, $\mathcal{P}_{st}(\tau)$ is the set of regular paths connecting node s to node t whose length is exactly equal to τ .

Note that in this formulation of the problem, we do not explicitly constrain the probability distribution $P(\wp)$ to sum to 1. However, since the quantity $f(\tau, \lambda)$ is a Poisson probability mass, this implies that the probability distribution sums to 1, which means

$$\sum_{s,t=1}^n \sum_{\tau=0}^{\infty} \sum_{\wp \in \mathcal{P}_{st}(\tau)} P(\wp) = \sum_{\tau=0}^{\infty} f(\tau, \lambda) = 1 \quad (6.18)$$

The problem presented in Eq. 6.17 can be solved by optimizing the following Lagrange function integrating equality constraints

$$\begin{aligned} \mathcal{L}(\mathbf{P}(\wp), \boldsymbol{\mu}) = & \sum_{s,t=1}^n \sum_{\tau=0}^{\infty} \sum_{\wp \in \mathcal{P}_{st}(\tau)} \left(\mathbf{P}(\wp) \tilde{c}(\wp) + T \mathbf{P}(\wp) \log \left(\frac{\mathbf{P}(\wp)}{\tilde{\pi}(\wp)} \right) \right) \\ & + \sum_{\tau=0}^{\infty} \mu_{\tau} \left(f(\tau, \lambda) - \sum_{s,t=1}^n \sum_{\wp \in \mathcal{P}_{st}(\tau)} \mathbf{P}(\wp) \right) \end{aligned} \quad (6.19)$$

over the set of path probabilities $\mathbf{P}(\wp)$ with $\wp \in \mathcal{P} = \cup_{s,t=1}^n \cup_{\tau=1}^{\infty} \mathcal{P}_{st}(\tau)$ (the bag of all possible regular paths). Minimizing Eq. 6.19 can be done by setting its partial derivative with respect to the set of path probabilities $\mathbf{P}(\wp)$ of length $\ell(\wp) = \tau$ to zero, which gives

$$\frac{\partial \mathcal{L}(\mathbf{P}(\wp), \boldsymbol{\mu})}{\partial \mathbf{P}(\wp)} = \tilde{c}(\wp) + T \log \left(\frac{\mathbf{P}(\wp)}{\tilde{\pi}(\wp)} \right) + T - \mu_{\tau} = 0 \text{ for } \wp \in \mathcal{P}_{st}(\tau) \quad (6.20)$$

Isolating the logarithm and defining $\theta = 1/T$ provides

$$\log \left(\frac{\mathbf{P}(\wp)}{\tilde{\pi}(\wp)} \right) = -\theta \tilde{c}(\wp) - 1 + \theta \mu_{\tau} \quad (6.21)$$

By taking the exponential, we obtain

$$\mathbf{P}(\wp) = \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)] \exp[\theta \mu_{\tau} - 1] \quad (6.22)$$

We can now rewrite the constraint of Eq. 6.18, expressing that the probability of sampling a path must follow a Poisson probability distribution, from Eq. 6.22 as

$$f(\tau, \lambda) = \sum_{s,t=1}^n \sum_{\wp \in \mathcal{P}_{st}(\tau)} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)] \exp[\theta \mu_{\tau} - 1] \quad (6.23)$$

which means that

$$\exp[\theta \mu_{\tau} - 1] = \frac{f(\tau, \lambda)}{\sum_{s,t=1}^n \sum_{\wp' \in \mathcal{P}_{st}(\tau)} \tilde{\pi}(\wp') \exp[-\theta \tilde{c}(\wp')]} \quad (6.24)$$

Finally, the path probabilities $\mathbf{P}(\wp)$ can be defined from Eqs. 6.22 and 6.24 as

$$\mathbf{P}(\wp) = \frac{f(\tau, \lambda) \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\sum_{s,t=1}^n \sum_{\wp' \in \mathcal{P}_{st}(\tau)} \tilde{\pi}(\wp') \exp[-\theta \tilde{c}(\wp')]} = \frac{f(\tau, \lambda) \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\mathcal{Z}(\tau)} \quad (6.25)$$

where $\mathcal{Z}(\tau)$ is the partition function associated with the set of paths of length $\ell(\wp) = \tau$, that is, $\mathcal{P}_{st}(\tau)$.

6.3.2 Computing the joint probability of drawing a path starting in i and ending in j

We can now define the probability of drawing a path starting in node $s = i$ and ending in some other node $t = j$ considering the set of all paths connecting i to j in exactly τ steps as $\mathcal{P}_{ij}(\tau)$. From Eq. 6.25, we find

$$\begin{aligned}
 \pi_{ij} &= P(s = i, t = j) = \sum_{\tau=0}^{\infty} \sum_{\wp \in \mathcal{P}_{ij}(\tau)} P(\wp) \\
 &= \sum_{\tau=0}^{\infty} \sum_{\wp \in \mathcal{P}_{ij}(\tau)} \frac{f(\tau, \lambda) \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\sum_{i', j'=1}^n \sum_{\wp' \in \mathcal{P}_{i'j'}(\tau)} \tilde{\pi}(\wp') \exp[-\theta \tilde{c}(\wp')]} \\
 &= \sum_{\tau=0}^{\infty} f(\tau, \lambda) \frac{\sum_{\wp \in \mathcal{P}_{ij}(\tau)} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{\sum_{i', j'=1}^n \sum_{\wp' \in \mathcal{P}_{i'j'}(\tau)} \tilde{\pi}(\wp') \exp[-\theta \tilde{c}(\wp')]} \quad (6.26)
 \end{aligned}$$

Furthermore, we can define, the same way as in the standard BoP model (see Eqs. 3.7 and 3.8), the element of the fundamental matrix $\mathbf{Z}(\tau)$ as

$$z_{ij}(\tau) = \sum_{\wp \in \mathcal{P}_{ij}(\tau)} \frac{\tilde{\pi}(\wp) \exp[-\theta \tilde{c}(\wp)]}{w(\wp)} = \sum_{\wp \in \mathcal{P}_{ij}(\tau)} w(\wp) \quad (6.27)$$

Finally, from Eq. 6.27, the probability of drawing a path starting in node $s = i$ and ending in some other node $t = j$, presented in Eq. 6.26, becomes

$$\pi_{ij}(\tau, \lambda) = P(s = i, t = j) = \sum_{\tau=0}^{\infty} f(\tau, \lambda) \frac{z_{ij}(\tau)}{z_{\bullet\bullet}(\tau)} \quad (6.28)$$

To obtain the probability $\pi_{ij}(\tau, \lambda)$ in matrix form, we now derive a recurrence expression to compute each term of the series in turn, presented in Eq. 6.28. To do so, recall that the quantity $\mathbf{Z}(\tau)$ can be computed thanks to $\mathbf{Z}(\tau) = \mathbf{W}^\tau$ (see Eqs. 3.7 and 6.27). Therefore, we need to iterate the following expressions until convergence,

$$\begin{cases} \mathbf{Z}(\tau + 1) = \mathbf{Z}(\tau) \mathbf{W} \\ \mathbf{\Pi}(\tau + 1, \lambda) = \mathbf{\Pi}(\tau, \lambda) + f(\tau + 1, \lambda) \frac{\mathbf{Z}(\tau + 1)}{z_{\bullet\bullet}(\tau + 1)} \end{cases} \quad (6.29)$$

where $\mathbf{\Pi}(\tau, \lambda)$ contains the elements π_{ij} of the truncated series (see Eq. 6.28) up to length τ . This update of $\mathbf{Z}(\tau)$ should usually converge quickly because it is expected that the spatial interactions in real-world networks are mainly local, which means that only low values of λ are relevant. In that situation, $f(\tau, \lambda)$ quickly drops to zero. Furthermore, when $\tau = 0$, we initialize the

matrix $\mathbf{Z}(\tau)$ and $\mathbf{\Pi}(\tau, \lambda)$ by

$$\begin{cases} \mathbf{Z}(0) = \mathbf{I} \\ \mathbf{\Pi}(0, \lambda) = f(0, \lambda) \frac{\mathbf{I}}{n} \end{cases} \quad (6.30)$$

when n is the number of nodes and $\mathbf{I}$ is the identity matrix. This means that, for zero-length paths, the source node and the target node must be the same, and thus $\pi_{ij}(0, \lambda) = f(0, \lambda) \delta_{ij}/n$.

The time complexity of the matrix $\mathbf{\Pi}(\tau, \lambda)$ computation is dominated by the matrix product performed at each iteration. Therefore, it is of order $k \cdot \mathcal{O}(n^3)$ where n is the number of nodes and k is the number of iterations required. Moreover, the number of iterations is at least λ .

6.3.3 A derived distance measure between nodes

We now derive a distance measure between nodes following the same procedure as in Subsection 6.2.3. More specifically, we take minus the logarithm of the probability matrix $\mathbf{\Pi}(\tau, \lambda)$ obtained after convergence of Eq. 6.29. The resulting distance matrix between nodes, called the directed *Poisson surprisal distance*, is defined as

$$\Delta^{\text{DPSURP}} = -\log \mathbf{\Pi}(\tau, \lambda) \quad (6.31)$$

with the $\log$ being the natural elementwise logarithm. It computes the *surprisal* of observing a path starting in i and ending in j , and is also an extension of the surprisal distance (see Subsection 3.4.1) introduced in [105] in the standard BoP framework. The *Poisson surprisal distance* measure is the symmetrized quantity,

$$\Delta^{\text{PSURP}} = \frac{1}{2} \left(\Delta^{\text{DPSURP}} + (\Delta^{\text{DPSURP}})^T \right) \quad (6.32)$$

where the diagonal elements are then set to zero by subtracting the $\text{Diag}(\Delta^{\text{PSURP}})$ to Δ^{PSURP} .

Unlike the other distance measures presented earlier in this chapter, we do not include the Poisson surprisal distance in our graph-based semi-supervised classification experiment in Chapter 7. We noticed that the results obtained by the Poisson surprisal distance and the Poisson weighted surprisal distance are very similar in our experimental setup. An explanation of this phenomenon is presented in the next subsection. Therefore, in our experiment, we only keep the Poisson weighted surprisal distance which also includes the weighting of the start and end nodes.

6.3.4 Correlation analysis

In this subsection, we analyze the correlation between the two distances introduced in Subsections 6.2.3 and 6.3.3, the Poisson weighted surprisal distance (PWSurp) and the Poisson surprisal distance (PSurp).

More specifically, we computed the Pearson correlation coefficient ρ [191] between the PWSurp and the PSurp for several θ and λ values on the News-2cl-1 dataset [298, 299] (see Section 7.2 for more details). We reported and represented the correlation results in Fig. 6.1.

From Fig. 6.1, we can observe that the two distances are highly correlated for all θ and λ values. The correlation is highest when θ is small, decreases when $\theta = 1$ and increases again for larger values of θ .

We could also see that the distances between the nodes increase as θ increases. Indeed, unlike the free energy distance and the randomized shortest path dissimilarity (see Subsections 3.4.2 and 3.4.3), the surprisal distance and its Poisson versions do not interpolate between the shortest path distance and the commute time distance.

Nevertheless, we can multiply the surprisal distances by a factor $1/\theta$ to ensure that they evolve in the same direction as the free energy distance and the randomized shortest path dissimilarity as a function of θ . That is, the distance between the nodes increases when $\theta \rightarrow 0^+$ and decreases when $\theta \rightarrow \infty$.

6.4 Conclusion

In this chapter, we presented two new bag-of-paths models incorporating a Poisson probability distribution. These two models allow to define the probability of a path by explicitly taking into account the length $\ell(\varphi)$ of this path. More specifically, in the first model, the probabilities of the paths are weighted according to a Poisson probability distribution on their length. In the second model, the probability of sampling a path length is constrained to be Poisson-distributed. The main idea is to be able to tune the most relevant mean path length in terms of the considered graph.

From these models, two new distance measures between nodes of the graph were introduced, the Poisson weighted surprisal distance and the Poisson surprisal distance. As pointed out in our analysis, these two distances are highly correlated and provide similar results in our graph-based semi-supervised classification experiment in Chapter 7. However, the Poisson weighted surprisal distance has a slightly lower complexity than the Poisson surprisal distance.

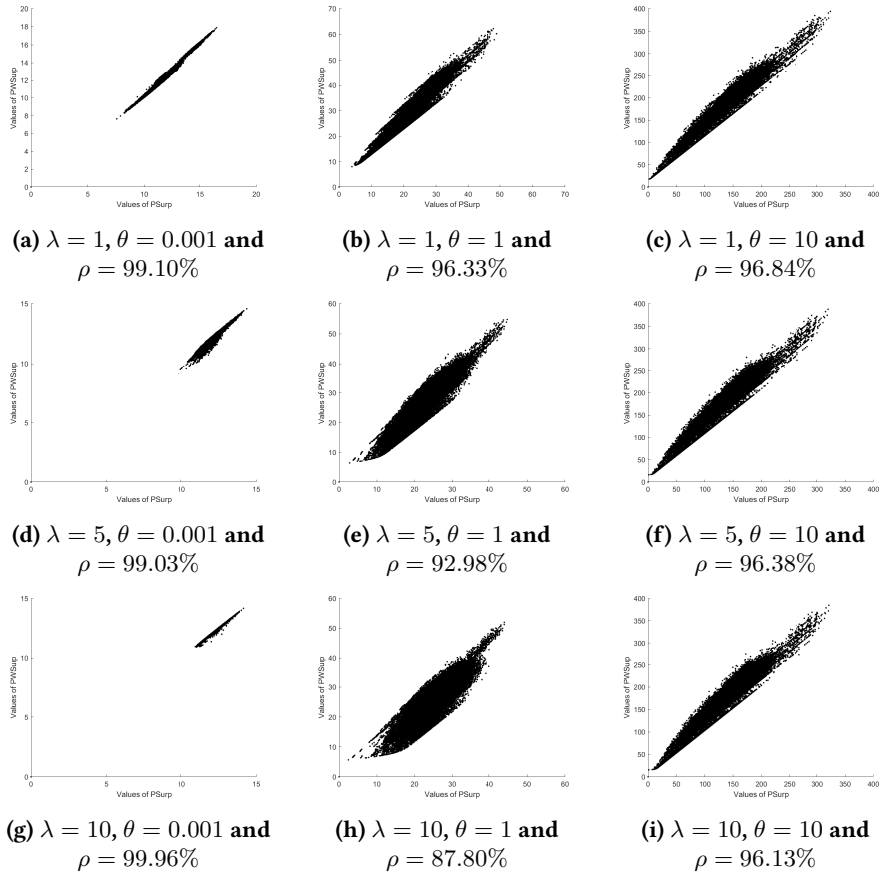


Figure 6.1: Pearson correlation coefficients ρ between the PSurp and the PW-Surp for several θ and λ values. The two distances are highly correlated regardless of the chosen parameters.

Moreover, a new interpretation of the logarithmic communicability kernel inspired by the first new model is proposed. Indeed, the logarithmic communicability kernel is similar to the Poisson weighted surprisal distance, except that it is based on the adjacency matrix $\mathbf{A}$ instead of the matrix $\mathbf{W}$.

One considered future work is to derive the randomized shortest path dissimilarity [156] (see Subsection 3.4.3) and the free energy distance [105] (see Subsection 3.4.2) within these new formulations of the bag-of-paths model.

Another interesting track would be to develop a similar approach in the randomized shortest path model instead of the bag-of-paths model. This extension would allow to penalize longer paths in the graph while considering flow and capacity constraints on the edges as presented in Chapters 4 and 5.

Still another one aims at investigating generalized Poisson distributions (see for instance [62]) on the path lengths which avoids the drawback of having a mean equal to the variance, as the standard Poisson distribution does.

We also plan to perform additional experiments on pattern recognition tasks, such as clustering, collaborative recommendation, and link prediction, to confirm the findings of our graph-based semi-supervised classification experiment presented in Chapter 7. Recent experiments in clustering and embedding have shown that the Poisson weighted surprisal distance and the Poisson surprisal distance also perform well in these tasks [58].

Graph-based semi-supervised classification

7

In this chapter, we propose a comprehensive comparison of 43 similarity and dissimilarity measures between nodes through **graph-based semi-supervised classification** tasks. The main objective of this experiment is to determine to which extent the different methods can deal with partial information about the graph structure, contained in only a few dimensions (5%, 10% and 20% node features extraction), as well as with the full information contained in the kernel matrix. This experiment is also the opportunity to evaluate for the first time the performance of the distances introduced in Chapters 5 and 6 in a semi-supervised classification context.

Graph-based semi-supervised classification or learning (GSSL) is a semi-supervised learning (SSL) method that has attracted the interest of many researchers in recent years, especially in the field of deep learning [57, 182, 186, 255, 259, 314]. The objective of GSSL is to build a graph along which the known labels can be easily propagated. This objective can be achieved by a two-step procedure. The first step is to build a graph from the data, including both labeled and unlabeled samples. The second step is to propagate the labeled information to the unlabeled samples through the graph. This whole procedure is based on the ‘local consistency’ assumption, according to which nodes that are close in the graph tend to have the same label [57, 255]. The same argument is used, for instance, in k-nearest neighbor classification.

Moreover, as all the other SSL methods, GSSL has the advantage of being able to work with a large amount of unlabeled data and few labeled data. This feature makes SSL methods very attractive because unlabeled data is more accessible than labeled data, which often depends on expensive human labor [101]. In addition, GSSL also has three other advantages compared with other SSL methods [57, 255]. The first one is that graph data are very common and are increasingly used in many areas (see Chapter 2). The second one is that most GSSL methods involve a convex optimization problem, which can be easily solved with various techniques [259]. The third one is that many new GSSL methods are designed to have linear complexity and to be parallelized to handle large datasets.

In recent years, a wide range of GSSL methods considering different approaches has been developed, such as graph regularization methods [24, 112, 288, 311], random-walk-based methods [14, 220, 261, 266], graph neural network methods [13, 152, 182, 186], and autoencoder-based methods [215, 277, 287] to name a few (see [57, 255, 314] and references therein for more details).

These GSSL methods are used in many real work applications such as fault detection in solar photovoltaic arrays [309], credit card fraud detection [170, 281, 313] (see Chapter 8), image segmentation [116], speech processing [181], sentiment analysis [216], word sense disambiguation [209] and many more.

The GSSL approach used in this chapter is inspired by previous comparisons of similarity and dissimilarity measures performed in [105, 122, 176]. This approach consists of first computing kernel matrices from the similarity and dissimilarity measures (see Section 2.4). Then, from these kernels, either a percentage of dominant eigenvectors is extracted and used as node features in a linear **support vector machine** (SVM), or the full kernels are directly fed into a kernel SVM (see Section 7.3 for more details).

However, none of the previous comparisons [105, 122, 176] compare that many similarity and dissimilarity measures. Moreover, these comparisons do not compare the performance of methods with partial and full information about the graph structure. Finally, none of them evaluates the performance of the distances introduced in Chapters 5 and 6.

The main motivations of this chapter are therefore to determine to which extent the different methods can deal with partial information and full information about the graph structure, and also to evaluate the performance of the distances introduced in Chapters 5 and 6 in a semi-supervised classification context. In short, the main contributions of this chapter are:

- ▶ An experimental comparison between the distances introduced in Chapters 5 and 6 as well as other state-of-the-art similarity and dissimilarity measures between nodes on a graph on semi-supervised classification problems;
- ▶ A comparison of the performance of all the investigated methods with partial information about the graph structure, contained in only a few dimensions (low dimensional space 5%, 10% and 20%), and with the full information contained in the kernel matrix;
- ▶ An analysis of the impact of the feature extraction technique (5%, 10%, 20% and kernel) on the classification results for the best-ranked investigated methods;
- ▶ An investigation of the best performing technique among all the best-ranked methods and all the feature extraction techniques (5%, 10% and

20% and kernel).

This chapter is organized as follows. First, the set of investigated methods is presented in Section 7.1. Second, the collection of network datasets used in our experiment is introduced in Section 7.2. Then, Section 7.3 provides details on the experimental design. Section 7.4 presents and discusses the results of the experiment. Finally, Section 7.5 concludes the chapter.

7.1 Investigated methods

In this section, we present the 43 similarity and dissimilarity measures investigated through our experiment. We have divided these methods into five categories to facilitate the analysis of the results in Section 7.4. However, we have to stress that this categorization is not absolute and that some methods could be associated with several categories. Moreover, most of the selected methods already showed competitive performance in previous semi-supervised [101, 105, 133, 176] and unsupervised tasks [134, 253, 254, 298, 299].

Before presenting the methods, let us first recall the concepts of similarity, dissimilarity, distance, and kernel, already presented in Chapter 2.

As explained in Subsection 2.3.1, a similarity between two nodes is a usually positive symmetric measure that increases with the proximity and degree of connectivity of these two nodes. Furthermore, the similarity between two nodes should in general (but not always) be lower or equal to the similarity measure of the node itself. A dissimilarity measure is the dual concept of the similarity measure and is always positive and symmetric. Moreover, a dissimilarity measure between two nodes is equal to zero if and only if these two nodes are identical. A distance is a dissimilarity measure that also satisfies the triangle inequality (see Eq. 2.15). A kernel is a positive semidefinite similarity measure (see Subsection 2.4.1 for details). Note that in our experiment, we transform all the methods into kernels. The details of this transformation are explained in Section 7.3.

Here are the different categories of investigated measures.

7.1.1 Bag-of-paths dissimilarities and similarities

This first category is composed of 16 similarity and dissimilarity measures defined in the bag-of-paths framework presented in Chapter 3. All these methods therefore share a common parameter θ which allows to monitor the balance between an optimal (exploitation) and a random (exploration) behavior.

As previously mentioned in Chapter 3, this parameter also allows to define dissimilarity measures which interpolate, for most of them, between the shortest path distance when θ is high and the commute time distance (up to a scaling factor) when θ is close to zero. Therefore, it seemed clear to us to

include the shortest path distance into this category. The commute distance in its kernel form is also included but in the fourth category presented in Subsection 7.1.4. The 16 methods are:

- The *shortest path distance (SP)* is defined in Subsection 2.3.2. As a reminder, the SP between two nodes i and j is defined as the path with the smallest cost between these two nodes, derived from the cost matrix $\mathbf{C}$ (see Subsection 2.2.2). This method is the most standard distance and has no parameter. In addition, this method is a baseline that should provide lower results than the other methods.
- The *bag-of-paths modularity matrix (BoPQ)* is the similarity measure studied in [80]. This measure is a random walk version of the modularity matrix (see Subsection 2.2.6 and Eq. 2.13) where the adjacency matrix $\mathbf{A}$ is replaced by the fundamental matrix $\mathbf{Z}$ of the bag-of-paths model defined in Eq. 3.7. This new modularity compares the probability of finding a path (instead of an edge) between two nodes within a cluster with the prior probability of this event on the whole graph. This method has no extra parameter.
- The *surprisal distance (Surp)*, the *free energy distance (FE)*, and the *randomized shortest path dissimilarity (RSP)* are the three dissimilarity measures developed in the origins of the bag-of-hitting-paths and randomized shortest path models [105, 156, 238]. These methods are respectively defined in Subsections 3.4.1, 3.4.2, and 3.4.3. Moreover, these methods have no extra parameter.
- The *Tsallis free energy distance (FETsa)* and the *Tsallis randomized shortest path dissimilarity (RSPTsa)* are the two dissimilarity measures defined in [176]. In this paper, the authors reformulate the randomized shortest path model by replacing the Kullback-Leibler divergence in Eq. 3.20 by a Tsallis divergence [275, 276]. These methods have one extra parameter r which has to be larger than 1. This parameter controls the impact of the costs on the probability distribution.
- The *margin-constrained bag-of-paths surprisal distance (cBoP)* is the distance introduced in [123] and re-derived in Subsection 5.3.5.2 from another point of view. This method has one additional parameter which is the non-negative weights vector $\mathbf{v}$. We tested three different types of prior vectors: the uniform weights, the L1-normalized degree weights, and the L1-normalized inverse degree weights defined as

- Uniform weights: $\mathbf{e}/n$;
- L1-normalized degree weights: $\mathbf{d}/(\mathbf{e}^T \mathbf{d})$;

- L1-normalized inverse degree weights: $(\mathbf{e} \div \mathbf{d}) / (\mathbf{e}^T (\mathbf{e} \div \mathbf{d}))$.

where $\div$ is the elementwise division, n is the number of nodes, and $\mathbf{d}$ is the degree vector (see Subsection 2.2.3). We refer to these methods as **cBoPUni**, **cBoPDegree** and **cBoPInvDegree**.

- The *margin-constrained bag-of-hitting-paths surprisal distance* (**cBoPH**) [123] is the counterpart of the previous method in terms of hitting paths. This method has one additional parameter which is the non-negative weights vector $\mathbf{v}$. We also tested the same three different types of prior vectors and refer to these methods as **cBoPHUni**, **cBoPHDegree** and **cBoPHInvDegree**.
- The *Poisson weighted surprisal distance with priors* (**PWSurp**) is the distance introduced in Subsection 6.2.4. This method has two additional parameters which are the prior vector $\boldsymbol{\sigma}$ and the parameter of the Poisson distribution λ . As with the two previous methods, we tested the same three different types of prior vectors and refer to these methods as **PWSurpUni**, **PWSurpDegree** and **PWSurpInvDegree**.

7.1.2 Covariance and correlation kernels

This second category includes the eight covariance and correlation kernels introduced in [122]. In this paper, the authors derive closed-form expressions to compute covariance and correlation kernels between nodes based on the presence and the number of occurrences of nodes on paths, for both regular and hitting paths, following a bag-of-paths approach. The idea is that two nodes are considered as similar when they often appear together on the same paths. More specifically, all the measures are derived from the fundamental matrices of regular paths $\mathbf{Z}$ (see Eq. 3.7) and of hitting paths $\mathbf{Z}_h$ (see Eq. 3.15). More information about the computation of these measures can be found in Subsection 5.1.1 of [122]. The eight covariance and correlation kernels are:

- The *covariance kernel based on the presence of nodes on regular paths* (**Cov**).
- The *covariance kernel based on the presence of nodes on hitting paths* (**CovH**).
- The *correlation kernel based on the presence of nodes on regular paths* (**Cor**).
- The *correlation kernel based on the presence of nodes on hitting paths* (**CorH**).

- The covariance kernel based on the number of occurrences of nodes on regular paths (**nCov**).
- The covariance kernel based on the number of occurrences of nodes on hitting paths (**nCovH**).
- The correlation kernel based on the number of occurrences of nodes on regular paths (**nCor**).
- The correlation kernel based on the number of occurrences of nodes on hitting paths (**nCorH**).

As in the previous section, all these kernels share a common parameter θ which controls the balance between an optimal (exploration) and a random (exploration) behavior.

7.1.3 Laplacian and adjacency based dissimilarities and similarities

This third category is composed of eight similarity and dissimilarity measures directly derived from the adjacency matrix $\mathbf{A}$ or the Laplacian matrix $\mathbf{L}^1$ introduced respectively in Subsections 2.2.1 and 2.2.5. The eight similarity and dissimilarity measures are:

- The *adjacency matrix* (**A**) presented in Subsection 2.2.1 generally allows to represent the structure of a graph. Nevertheless, this matrix can also be considered as a local similarity measure where two nodes are considered similar if there are connected. This method has no parameter and is a baseline that should provide lower results than the other methods.
- The *modularity matrix* (**Q**) [202, 204] defined in Eq. 2.12 is a similarity measure of the quality of a particular division of a network into communities. This method has no parameter and is also a baseline that should provide lower results than the other methods.
- The *Katz similarity matrix* (**Katz**) [147], or *Neumann similarity matrix* [241] is a method to compute degrees of influence, defined as $\mathbf{K} = \sum_{t=1}^{\infty} \alpha^t \mathbf{A}^t = (\mathbf{I} - \alpha \mathbf{A})^{-1} - \mathbf{I}$. This method has one parameter α which has to be chosen positive and smaller than the inverse of the spectral radius of $\mathbf{A}$, $\rho(\mathbf{A})$, to guarantee convergence. This parameter is a discounting factor which defines the attenuation in a link, where $\alpha = 0$ corresponds to a complete attenuation and $\alpha = 1$ to an absence of any attenuation [103].

¹Note that we do not include the Laplacian matrix $\mathbf{L}$ in our experiment because the eigenvectors of the matrix $\mathbf{L}$ and its Moore-Penrose pseudoinverse $\mathbf{L}^+$ are similar [103]. Henceforth, the results of these methods would be identical in our experimental setup. Therefore, we only kept the matrix $\mathbf{L}^+$ in the fourth category (see Subsection 7.1.4).

- The *Leicht similarity matrix (Leicht)*, also called the *degree-weighted Katz similarity matrix*, is an extension of the Katz developed by Leicht et al. in [174], defined as $\mathbf{K} = \mathbf{D}^{-1}(\mathbf{I} - \alpha\mathbf{A})^{-1}\mathbf{D}^{-1}$ where $\mathbf{D}$ is the degree matrix (see Subsection 2.2.3). In this extension, the authors of [174] propose to normalize each term of the Katz by dividing the number of paths of length t , provided by the power of the adjacency matrix $\mathbf{A}^t$, by the expected number of such paths if the nodes are randomly connected in the graph. This method has one parameter α which is identical to the parameter α of the Katz.
- The *local path index (LPI)* [184, 312] is also a variant of the Katz that focuses on local information, defined as $\mathbf{K} = \mathbf{A}^2 + \alpha\mathbf{A}^3$. Unlike the Katz which incorporates path contributions of all lengths, the LPI focuses only on path contributions of lengths two and three. This method has one parameter α which controls the weight given to paths of length three connecting each pair of nodes.
- The *regularized Laplacian kernel (RL)* [132], also called the *forest accessibility matrix* [53], is a kernel that quantifies the relative accessibility between nodes, defined as $\mathbf{K} = (\mathbf{I} - \alpha\mathbf{L})^{-1}$. This method has one parameter α which has to be chosen positive and smaller than the inverse of the spectral radius of $\mathbf{L}$, $\rho(\mathbf{L})$, to guarantee convergence.
- The *modified regularized Laplacian kernel (MRL)* [132] is, as its name suggests, a modified version of the RL where a new parameter γ is introduced to control the bias between importance and relatedness. This modified kernel is defined as $\mathbf{K} = (\mathbf{I} + \alpha\mathbf{L}_\gamma)^{-1}$ with $\mathbf{L}_\gamma = \gamma\mathbf{D} - \mathbf{A}$ where $\mathbf{D}$ is the degree matrix (see Subsection 2.2.3). This method has two parameters α and γ . The parameter α has to be chosen positive and smaller than the inverse of the spectral radius of $\mathbf{L}_\gamma$, $\rho(\mathbf{L}_\gamma)$, to guarantee convergence. The parameter γ must be between 0 and 1. When $\gamma = 1$, the MRL corresponds to the RL, while when $\gamma = 0$, the MRL is identical to the Katz, except for the presence of the second term $\mathbf{I}$ [103].
- The *logarithmic forest distance (LF)* is the distance proposed in [50] built on the matrix forest theorem [53]. This method has one parameter α which allows to interpolate up to a scaling factor between the shortest path distance ($\alpha \rightarrow 0^+$) and the resistance distance ($\alpha \rightarrow +\infty$) [157].

7.1.4 Commute time-based kernels

This fourth category includes five kernels derived from the commute time distance presented in Subsection 2.3.3. The five kernels are:

- The *commute time kernel (CT)* [104, 239] simply corresponds to the Moore-Penrose pseudoinverse of the Laplacian matrix $\mathbf{L}^+$ defined in Eq. 2.9. This method has no parameter.
- The *corrected commute time kernel (CCT)* proposed by von Luxburg et al. in [284] to alleviate the random walker getting ‘lost in space’ problem (see introduction of Chapter 3 for more details). This kernel is defined as $\mathbf{K} = \mathbf{H}\mathbf{D}^{-\frac{1}{2}}\mathbf{M}(\mathbf{I} - \mathbf{M})^{-1}\mathbf{M}\mathbf{D}^{-\frac{1}{2}}\mathbf{H}$ with $\mathbf{M} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} - \frac{\mathbf{d}\mathbf{d}^T}{\text{vol}(G)})\mathbf{D}^{-\frac{1}{2}}$ where $\mathbf{D}$ is the degree matrix (see Subsection 2.2.3) and $\mathbf{H}$ is the centering matrix (see Eq. 2.27). This method has no parameter.
- The *regularized commute time kernel (RCTK)* is a variant of the CT introduced in [310] based on the inversion of a regularized version of the matrix $\mathbf{L}$ instead of the matrix $\mathbf{L}^+$, defined as $\mathbf{K} = (\mathbf{D} - \alpha\mathbf{A})^{-1}$. This method has one regularization parameter α ranging from 0 to 1. Moreover, this kernel is used in the fraud detection field in Chapter 8.
- The *sigmoid commute time similarity (SCT)* and *sigmoid corrected commute time similarity (SCCT)* [298, 299] are obtained by applying an elementwise sigmoid transformation on the CT and on the CCT. The elementwise sigmoid transformation is defined as $\mathbf{K}_{ij}^s = \frac{1}{1 + \exp[-\alpha k_{ij}/\sigma]}$ where σ is the standard deviation of the elements of $\mathbf{K}$. This transformation allows to normalize the range of similarities between 0 and 1. It also increases the contrast between the different communities present in the graph [103]. These methods have one parameter α which controls the sharpness of the sigmoid transformation.

7.1.5 Diffusion-based similarities

This fifth and last category is composed of six similarity measures based on diffusion models. The six similarity measures are:

- The *random walk with restart similarity (RWWR)* [273], also called *rooted PageRank* [179], is inspired by the well-known PageRank algorithm [214]. This kernel is defined as $\mathbf{K} = (\mathbf{I} - \alpha\mathbf{P})^{-1}$ where $\mathbf{P}$ is the reference transition probability matrix (see Eq. 2.6). This method has one parameter α ranging from 0 to 1 which is the probability for a random walker to continue the walk. Moreover, this similarity is used in the fraud detection field in Chapter 8.
- The *exponential diffusion kernel (ExpD)* developed by Kondor and Lafferty [160], is defined as $\mathbf{K} = \text{expm}(\alpha\mathbf{A})$ where expm is the matrix exponential. The elements k_{ij} of this kernel correspond to the sum of the number of paths of each length, respectively discounted by their

length, between node i and j . This method has one positive parameter α which is a discounting parameter favoring shorter paths by giving them a larger weight. This kernel is equivalent to the communicability kernel introduced in the physics community by Estrada et al. [92] when $\alpha = 1$.

- The *heat kernel (Heat)* [160], also called *Laplacian exponential diffusion kernel* [252], derives its name from its interpretation as a diffusion process related to the heat equation [168]. This kernel is defined as $\mathbf{K} = \expm(-\alpha\mathbf{L})$ where $\mathbf{L}$ is the Laplacian matrix (see Eq. 2.8). This method has one positive discounting parameter α .
- The *logarithmic communicability similarity (LogCom)* is the measure introduced in [134] corresponding to the logarithmic version of the communicability kernel, or more precisely of the ExpD, presented in Subsection 6.2.3. Indeed, the authors of [134] observed that taking the elementwise logarithm of the ExpD significantly improves its results. This method has one positive discounting parameter α favoring the paths between two nodes with a short length by giving them a larger weight.
- The *Markov diffusion kernel (MD)* [104] is an adaptation of a diffusion distance defined in [201, 226] between nodes of a graph in a continuous-time diffusion model. This kernel is computed as $\mathbf{K} = \mathbf{H}\mathbf{P}^t\mathbf{W}(\mathbf{P}^T)^t\mathbf{H}$ where $\mathbf{P}$ is the reference transition probability matrix (see Eq. 2.6), $\mathbf{H}$ is the centering matrix (see Eq. 2.27) and $\mathbf{W}$ is a diagonal weighting matrix containing the inverse of the degrees. This method has one non-negative parameter t which is the number of considered steps of the Markov process.
- The *time-averaged Markov diffusion kernel (AMD)* [104] is a variant of the MD based on the average visiting rate of a random walker. Unlike the MD which requires a regular Markov chain, the AMD can also handle periodic Markov chains. This kernel is defined the same way as MD but the matrix $\mathbf{P}^t$ is replaced by the matrix $\mathbf{F}(t) = \frac{1}{t} \sum_{\tau=1}^t \mathbf{P}^\tau$ (the average among the t first steps), which gives $\mathbf{K} = \mathbf{H}\mathbf{F}(t)\mathbf{W}\mathbf{F}^T(t)\mathbf{H}$. This method has one non-negative parameter t which is the number of steps of the Markov process.

A summary of all the investigated methods with their acronym and their measure type is available in Table 7.1. Moreover, the parameters of all the investigated methods and the values tested for each parameter are summarized in Table 7.2.

Method	Method acronym	Measure type
Category 1		
Bag-of-paths modularity matrix	BoPQ	Similarity
Margin-constrained bag-of-paths surprisal distance	cBoP	Distance
Margin-constrained bag-of-hitting-paths surprisal distance	cBoPH	Distance
Poisson weighted surprisal distance with priors	PWSurp	Distance
Free energy distance	FE	Distance
Tsallis free energy distance	FETsa	Distance
Randomized shortest path dissimilarity	RSP	Dissimilarity
Tsallis randomized shortest path dissimilarity	RSPTsa	Dissimilarity
Shortest path distance	SP	Distance
Surprisal distance	Surp	Distance
Category 2		
Correlation kernel based on the number of occurrences of nodes on regular paths	nCor	Kernel
Correlation kernel based on the number of occurrences of nodes on hitting paths	nCorH	Kernel
Covariance kernel based on the number of occurrences of nodes on regular paths	nCov	Kernel
Covariance kernel based on the number of occurrences of nodes on hitting paths	nCovH	Kernel
Correlation kernel based on the presence of nodes on regular paths	Cor	Kernel
Correlation kernel based on the presence of nodes on hitting paths	CorH	Kernel
Covariance kernel based on the presence of nodes on regular paths	Cov	Kernel
Covariance kernel based on the presence of nodes on hitting paths	CovH	Kernel
Category 3		
Adjacency matrix	A	Similarity
Katz similarity matrix	Katz	Similarity
Leicht similarity matrix	Leicht	Similarity
Local path index	LPI	Similarity
Logarithmic forest distance	LF	Distance
Modified regularized Laplacian kernel	MRL	Kernel
Modularity matrix	Q	Similarity
Regularized Laplacian kernel	RL	Kernel
Category 4		
Commute time kernel	CT	Kernel
Corrected commute time kernel	CCT	Kernel
Regularized commute time kernel	RCTK	Kernel
Sigmoid commute time similarity	SCT	Similarity
Sigmoid corrected commute time similarity	SCCT	Similarity
Category 5		
Time-averaged Markov diffusion kernel	AMD	Kernel
Exponential diffusion kernel	ExpD	Kernel
Heat kernel	Heat	Kernel
Logarithmic communicability similarity	logCom	Similarity
Markov diffusion kernel	MD	Kernel
Random walk with restart similarity	RWWR	Similarity

Table 7.1: The methods investigated in our experiment with their acronym and their measure type.

Method acronym	Parameters	Parameters values
Category 1		
BoPQ	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
cBoPUni	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
cBoPDegree	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
cBoPInvDegree	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
cBoPHUni	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
cBoPHDegree	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
cBoPHInvDegree	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
PWSurpUni	$\theta - \lambda$	$\{10^{-6}, 10^{-5}, \dots, 10\} - \{1, 2, 3, 5, 10\}$
PWSurpDegree	$\theta - \lambda$	$\{10^{-6}, 10^{-5}, \dots, 10\} - \{1, 2, 3, 5, 10\}$
PWSurpInvDegree	$\theta - \lambda$	$\{10^{-6}, 10^{-5}, \dots, 10\} - \{1, 2, 3, 5, 10\}$
FE	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
FETsa	$\theta - r$	$\{10^{-6}, 10^{-5}, \dots, 10^4\} - \{1.1, 1.2, \dots, 2\}$
RSP	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
RSPTsa	$\theta - r$	$\{10^{-6}, 10^{-5}, \dots, 10^4\} - \{1.1, 1.2, \dots, 2\}$
SP	/	
Surp	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
Category 2		
nCor	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
nCorH	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
nCov	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
nCovH	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
Cor	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
CorH	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
Cov	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
CovH	θ	$\{10^{-6}, 10^{-5}, \dots, 10\}$
Category 3		
A	/	
Katz	α	$\{0.05, 0.15, \dots, 0.95\} * (\rho(\mathbf{A}))^{-1}$
Leicht	α	$\{0.05, 0.15, \dots, 0.95\} * (\rho(\mathbf{A}))^{-1}$
LPI	α	$\{0.1, 0.2, \dots, 1\}$
LF	α	$\{10^{-6}, 10^{-5}, \dots, 10\}$
MRL	$\alpha - \gamma$	$\{0.05, 0.15, \dots, 0.95\} * (\rho(\mathbf{L}_\gamma))^{-1} - \{0.1, 0.2, \dots, 1\}$
Q	/	
RL	α	$\{0.05, 0.15, \dots, 0.95\} * (\rho(\mathbf{L}))^{-1}$
Category 4		
CT	/	
CCT	/	
RCTK	α	$\{0.05, 0.15, \dots, 0.95\}$
SCT	α	$\{5, 10, \dots, 50\}$
SCCT	α	$\{5, 10, \dots, 50\}$
Category 5		
AMD	t	$\{5, 10, \dots, 50\}$
ExpD	α	$\{10^{-6}, 10^{-5}, \dots, 1\}$
Heat	α	$\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10\}$
LogCom	α	$\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10\}$
MD	t	$\{5, 10, \dots, 50\}$
RWWR	α	$\{0.05, 0.15, \dots, 0.95\}$

Table 7.2: Parameter ranges for the investigated methods by category. The symbol ‘/’ means that the method has no parameter.

7.2 Datasets

The different methods are compared on 14 well-known social network datasets, already used in previous semi-supervised [105, 122, 176] and unsupervised tasks [253, 254, 298, 299].

- **WebKB (DB1-DB4).** These datasets [187] come from networks of co-citation between webpages of computer science departments of 4 different universities: **WebKB-Cornell**, **WebKB-Texas**, **WebKB-Washington** and **WebKB-Wisconsin**. Each page of these websites has been manually labeled to form six different classes: course, department, faculty, project, staff and student.
- **IMDB (DB5).** This dataset comes from the well-known *Internet Movie Database* [187]. Nodes represent movies and the adjacency matrix contains the number of production companies that two movies have in common. There are only two labels in this dataset: box-office hit or not.
- **20 Newsgroups (DB6-DB14).** The *Newsgroup* dataset is composed of 20.000 documents taken from 20 discussion groups of the Usenet diffusion list [87]. Nine subsets have been extracted from this data (for details, see [298, 299]): **News-2cl-1**, **News-2cl-2**, **News-2cl-3**, **News-3cl-1**, **News-3cl-2**, **News-3cl-3**, **News-5cl-1**, **News-5cl-2**, and **News-5cl-3**. As their names suggest, there are different numbers of classes/topics in these datasets (e.g., **News-3cl-1** contains documents from 3 different topics considered as classes). For each of these datasets, a sparse graph structure was derived from the *term-document* matrix $\mathbf{T} \triangleq (t_{ij})$, with t_{ij} being the *tf-idf* score [188] of term i in document j . The corresponding adjacency matrix is then obtained by $\mathbf{A} = \mathbf{T}^T \mathbf{T}$.

All these datasets are described by an adjacency matrix $\mathbf{A}$ and a class label vector $\mathbf{y}$. Note that we consider that each graph is undirected and we assert it by using $\mathbf{A} = (\mathbf{A} + \mathbf{A}^T)/2$. We also divided all the elements of $\mathbf{A}$ by the mean of $\mathbf{A}$ to avoid scalability problem. Furthermore, the elements of the cost matrix $\mathbf{C}$ are defined as $c_{ij} = 1/a_{ij}$ as for electrical networks [105]. More details about these concepts can be found in Subsections 2.1.1, 2.2.1 and 2.2.2. A summary of the main characteristics of each dataset is available in Table 7.3.

7.3 Experimental methodology

Our experimental design is closely related to the one used in [105, 122, 176]. This design was initially inspired by [267, 268, 269, 306, 307]. This section

Dataset Name	#Labels	#Nodes	#Edges	Prior of the majority class
WebKB-Cornell (DB1)	6	346	13416	41.91%
WebKB-Texas (DB2)	6	334	16494	48.80%
WebKB-Washington (DB3)	6	434	15231	39.17%
WebKB-Wisconsin (DB4)	6	348	16625	44.54%
IMDB (DB5)	2	1126	20282	50.18%
News-2cl-1 (DB6)	2	400	33854	50.00%
News-2cl-2 (DB7)	2	398	21480	50.25%
News-2cl-3 (DB8)	2	399	36527	50.13%
News-3cl-1 (DB9)	3	600	70591	33.34%
News-3cl-2 (DB10)	3	598	68201	33.44%
News-3cl-3 (DB11)	3	595	64169	33.61%
News-5cl-1 (DB12)	5	998	176962	20.04%
News-5cl-2 (DB13)	5	999	164452	20.02%
News-5cl-3 (DB14)	5	997	155618	20.06%

Table 7.3: Main characteristics of the datasets used in our experiment.

presents the procedure and emphasizes the main differences.

The graph-based semi-supervised classification methodology is divided into two parts. The first part consists of extracting $\{5\%, 10\%, 20\%\}$ of the dominant eigenvectors of a kernel matrix to use them as node features in a linear support vector machine (SVM)². These extracted features contain condensed information about the graph structure. The second part consists of directly feeding the kernel matrix into a kernel SVM³. We refer to the first part as 5%F, 10%F, and 20%F and the second part as Ker.

The kernel matrices $\mathbf{K}$ are derived from the dissimilarity matrices using both classical multidimensional scaling (MDS) [35] and Gaussian transformation (Gauss) [241] (see Subsection 2.4.2). We also considered centering the Gaussian kernels (GaussCenter)⁴ by applying the following transformation $\mathbf{K} \leftarrow \mathbf{H}\mathbf{K}\mathbf{H}$ where $\mathbf{H}$ is the centering matrix (see Eq. 2.27). Regarding the similarity measures, we ensure that each of them is a valid kernel $\mathbf{K}$ by performing a spectral decomposition (see Subsection 2.4.1) and removing all negative eigenvalues.

For the first part of the classification method, we also try two different options to extract the node features. The first option is to weight the dominant eigenvectors by the square root of their corresponding eigenvalues before concatenating them into the data matrix $\mathbf{X}$. The matrix $\mathbf{X}$ contains the features of the nodes on its rows and is used as the input of the SVM. This option is equivalent to a spectral decomposition limited to a reduced number of dimensions and it is denoted as SD (see classical multidimensional scaling

²We use the LIBSVM library [48] with the options ‘-s 0’ and ‘-t 0’.

³We use the LIBSVM library [48] with the options ‘-s 0’ and ‘-t 4’.

⁴We do not apply this transformation to the MDS kernels as they are centered by construction.

in Subsection 2.4.1). The second option corresponds to directly concatenating the dominant eigenvectors into the matrix $\mathbf{X}$ and to normalize each row, in such a way that the resulting node feature vectors are of unit length. This normalization corresponds to a projection of the rows of $\mathbf{X}$ on the unit radius sphere centered at the origin that removes the effect of the size of the feature vectors (only the direction is relevant). We refer to this second option as NSD (normalized spectral decomposition).

For conciseness, we only report for each method the results of the best kernel transformation and feature extraction options according to the Nemenyi tests rankings [79]. The best combination for each method is reported in Table 7.4.

The performance of the different methods are evaluated in terms of classification accuracy for each extraction set. To avoid large variance in the results, all the methods are assessed by repeating 10 runs of a standard 5×5 nested cross-validation methodology with different folds of labeled/unlabeled nodes. In each external 5-folds cross-validation, methods are trained on 1 fold containing 20% of the labels, and the remaining 80% of the labels are hidden. Indeed, the most interesting setting in practical applications is the one where there are few labeled nodes compared to unlabeled. The parameters of each method are tuned on the training fold by performing an internal 5-fold cross-validation with a labeling rate of 80%. In each run, external and internal folds are kept identical for all methods. The final accuracy and standard deviation are obtained by averaging the 50 results of the external cross-validation folds. For clarity, the whole procedure is summarized in Fig. 7.1.

7.4 Results and discussion

In this section, we present and discuss the results of our experiment. The section is organized as follows. First, the influence of the weight parameters of the margin-constrained bag-of-paths methods and also the influence of the prior vectors of the Poisson weighted surprisal distances with priors on the accuracy scores are analyzed in Subsection 7.4.1. Second, the results of each method are assessed by category in Subsection 7.4.2. Then, the performance of the two best-ranked methods, according to the Nemenyi tests rankings [79], in each category and in each feature extraction set, are compared to each other in Subsection 7.4.3. After that, the impact of the feature extraction technique on the classification results for the best-ranked methods in each category is analyzed in Subsection 7.4.4. Finally, an investigation of the best performing technique among all the best-ranked methods and the feature extraction techniques is presented in Subsection 7.4.5.

Feature set →: Method ↓:	5%	10%	20%	Ker
Category 1				
BoPQ	SD	SD	SD	/
cBoPUni	Gauss SD	Gauss SD	GaussCenter SD	GaussCenter
cBoPDegree	GaussCenter SD	GaussCenter SD	GaussCenter SD	GaussCenter
cBoPInvDegree	GaussCenter SD	GaussCenter SD	GaussCenter SD	GaussCenter
cBoPHUni	Gauss SD	Gauss SD	GaussCenter SD	Gauss
cBoPHDegree	GaussCenter SD	Gauss SD	GaussCenter SD	Gauss
cBoPHInvDegree	Gauss SD	Gauss SD	GaussCenter SD	Gauss
PWSurpUni	Gauss SD	Gauss SD	GaussCenter SD	Gauss
PWSurpDegree	GaussCenter SD	GaussCenter SD	GaussCenter SD	Gauss
PWSurpInvDegree	Gauss SD	GaussCenter SD	MDS SD	MDS
FE	GaussCenter SD	GaussCenter SD	Gauss SD	Gauss
FETsa	GaussCenter SD	GaussCenter SD	GaussCenter SD	GaussCenter
RSP	GaussCenter SD	Gauss SD	GaussCenter SD	Gauss
RSPtSa	Gauss SD	Gauss SD	GaussCenter SD	Gauss
SP	Gauss SD	Gauss SD	GaussCenter SD	GaussCenter
Surp	Gauss SD	Gauss SD	GaussCenter SD	Gauss
Category 2				
nCor	SD	SD	SD	/
nCorH	SD	SD	SD	/
nCov	NSD	SD	SD	/
nCovH	NSD	SD	SD	/
Cor	SD	SD	SD	/
CorH	SD	SD	SD	/
Cov	SD	SD	SD	/
CovH	SD	SD	SD	/
Category 3				
A	SD	SD	SD	/
Katz	SD	SD	SD	/
Leicht	NSD	NSD	NSD	/
LPI	SD	SD	SD	/
LF	Gauss SD	GaussCenter SD	Gauss SD	Gauss
MRL	SD	SD	SD	/
Q	SD	SD	SD	/
RL	NSD	NSD	NSD	/
Category 4				
CT	NSD	NSD	NSD	/
CCT	NSD	NSD	NSD	/
RCTK	NSD	NSD	NSD	/
SCT	SD	SD	SD	/
SCCT	SD	SD	SD	/
Category 5				
AMD	NSD	NSD	NSD	/
ExpD	SD	SD	SD	/
Heat	NSD	NSD	SD	/
logCom	SD	SD	SD	/
MD	NSD	NSD	NSD	/
RWWR	SD	SD	SD	/

Table 7.4: The best combination of kernel transformation and feature extraction options for each method according to the Nemenyi tests rankings. The symbol ‘/’ means that no transformation is applied.

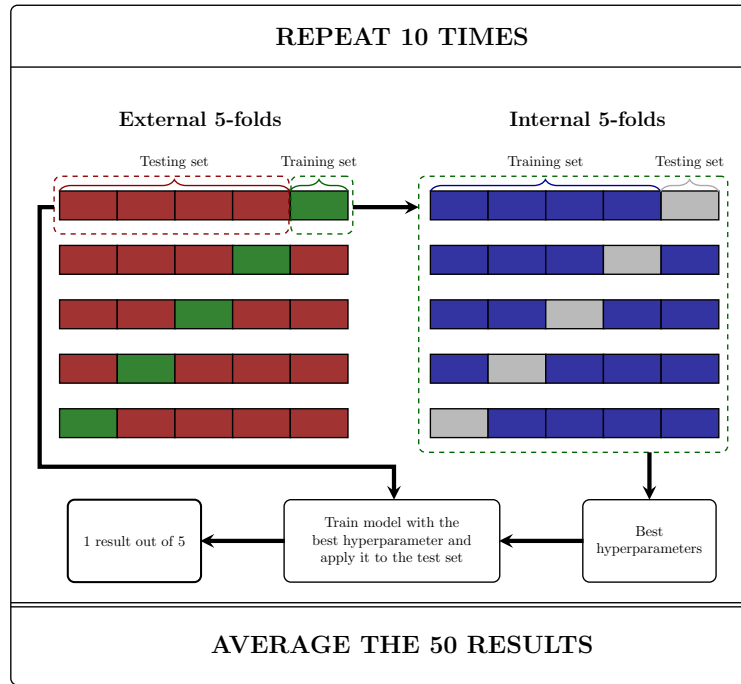


Figure 7.1: Representation of the experimental design used on our experiment. For each dataset and method, 10 runs of a 5×5 nested cross-validation methodology with different folds of labeled/unlabeled nodes on each run is performed. In each external fold, 20% of the labels are used to train the model and the remaining 80% are hidden to compute the accuracy score of the classifier. In each internal fold, the parameters are tuned on the external training fold by keeping 80% of the nodes labeled. The final accuracy is obtained by averaging the 50 accuracy score obtained on the external folds across the 10 runs.

7.4.1 Weights and priors analysis

In this subsection, as already mentioned, we focus our analysis on the influence of the three weight parameters $\mathbf{v}$ and the three prior vectors $\boldsymbol{\sigma}$ on the results of the cBoP, the cBoPH, and the PWSurp methods. Recall that we consider these parameters as external (see Subsection 7.1.1), i.e., we do not tune them in the cross-validation process. Therefore, the purpose is to determine which value of these external parameters provides the best performance in our experiment.

We report the classification accuracy and standard deviation averaged over the 50 runs of the cBoP, the cBoPH, and the PWSurp methods for the 14 datasets and the four extracted feature sets in Table 7.5. The best performing external parameter value is highlighted in boldface for each method, each dataset, and each feature set.

From the raw results in Table 7.5, it can be observed that the L1-normalized degree weights and priors obtain most of the time the best results across all the feature sets. More specifically, the cBoPDegree, the cBoPHDegree, and the PWSurpDegree appear respectively 31, 34, and 34 times across the 56 highlighted results of the cBoP, the cBoPH, and the PWSurp.

The next step of our analysis consists of comparing the results of different external parameters across all the 14 datasets through a Friedman test followed by a Nemenyi post-hoc test [79]. The Friedman test is a non-parametric equivalent of the repeated-measures ANOVA. The null hypothesis (H_0) of this test is that all the classifiers have the same average ranks. Most of the p -values of the Friedman tests are superior to the threshold α of 0.05, meaning that we cannot reject H_0 , except for the cBoPH in 5% and in Ker where the p -values are respectively 9.8×10^{-4} and 7.9×10^{-4} . Therefore, we cannot state that there is a statistical difference between the results obtained by the three external parameter for the cBoP, the cBoPH, and the PWSurp except for the cBoPH in 5%F and in Ker.

Even though not all the Friedman tests are positive, we perform Nemenyi tests, which determine whether or not the performance of each method differs significantly from another and provide a ranking of the methods. The results of these tests are reported in Fig. 7.2. First of all, we can observe that the rankings provided by the Nemenyi tests confirm that the L1-normalized degree is the external parameter value that provides the highest mean rank for each method in each feature set. Moreover, the tests highlight that the cBoPHDegree outperforms cBoPHInvDegree in 5%F and Ker, and also the cBoPHUni in 5%F.

We continue our analysis by performing multiple Wilcoxon signed-ranks tests for matched data [79, 293] to potentially discover other significant pairwise differences between the external parameter values. The Wilcoxon sign-

Method →: Dataset ↓:	cBoPUni	cBoPDegree	cBoPInvDegree	cBoPHUni	cBoPHDegree	cBoPHInvDegree	PWSurpUni	PWSurpDegree	PWSurpInvDegree
5%F									
DB1	58.83±3.83	59.41±3.88	59.31±2.97	58.61±3.21	58.82±2.82	58.75±2.51	58.97±3.01	60.06±2.79	58.68±3.23
DB2	78.74±2.92	78.62±2.34	78.18±2.65	76.83±2.96	77.44±3.05	76.33±3.07	77.89±2.78	78.50±2.22	76.95±2.97
DB3	70.55±2.87	71.10±3.02	69.60±2.82	68.82±2.43	69.65±2.74	68.18±2.32	69.68±2.85	70.89±2.90	68.43±2.67
DB4	79.22±1.90	79.43±1.99	77.11±1.92	76.72±2.34	78.53±2.07	77.06±2.24	78.53±1.93	79.09±1.76	77.20±1.68
DB5	77.55±1.66	77.61±1.20	77.77±1.44	78.09±1.59	78.29±1.74	77.88±1.43	77.72±1.43	78.01±1.35	77.55±1.45
DB6	95.52±1.31	95.50±1.30	95.58±1.39	95.71±1.20	95.84±1.04	95.71±1.21	95.56±1.51	95.90±1.29	95.58±1.44
DB7	91.84±2.13	91.64±2.22	91.81±1.93	93.15±1.44	93.29±1.50	92.74±1.60	92.60±1.69	92.54±1.54	92.76±1.49
DB8	95.90±1.11	96.18±1.01	95.99±1.19	96.29±1.58	96.47±0.81	96.53±0.92	96.33±1.01	96.40±0.86	96.40±0.89
DB9	93.00±1.31	92.66±1.28	92.82±1.17	92.86±1.49	93.05±1.43	92.99±1.30	92.75±1.19	93.01±1.33	92.68±1.17
DB10	91.97±1.74	92.00±1.51	92.12±1.62	92.29±1.61	92.50±1.38	92.43±1.50	92.96±1.49	92.61±1.43	92.88±1.69
DB11	92.06±1.28	92.14±1.65	91.97±1.34	92.66±1.47	92.91±1.36	92.48±1.27	92.51±1.31	92.74±1.60	92.59±1.11
DB12	88.54±1.17	88.49±0.95	88.10±1.11	88.73±1.05	88.79±1.05	88.69±1.05	88.72±1.13	88.80±1.20	88.72±1.06
DB13	82.74±1.53	82.76±1.42	82.73±1.69	83.20±1.38	83.23±1.33	83.25±1.51	83.46±1.33	83.12±1.53	83.15±1.41
DB14	81.84±1.50	82.19±1.36	81.60±1.56	82.41±1.67	82.51±1.73	82.64±1.51	82.95±1.61	82.55±1.58	82.90±1.56
10%F									
DB1	56.96±3.12	58.08±3.25	57.01±3.44	56.11±3.80	57.98±3.19	56.65±3.60	57.38±3.55	57.93±3.34	57.16±3.56
DB2	77.52±3.79	78.05±3.14	76.46±3.49	75.49±3.62	76.82±3.03	75.06±3.32	77.22±3.09	78.03±2.86	75.74±3.54
DB3	69.87±2.67	70.46±2.58	68.21±2.42	67.37±2.18	68.29±2.46	67.71±2.47	69.32±2.51	69.09±1.14	67.43±2.71
DB4	78.41±2.49	79.61±2.01	76.61±2.11	76.35±2.43	77.67±2.17	75.76±2.09	77.92±1.90	78.79±2.01	76.24±2.50
DB5	77.95±1.41	77.76±1.31	78.34±1.30	78.20±1.10	78.95±1.42	78.11±1.38	77.74±1.53	78.09±1.35	77.87±1.42
DB6	94.88±1.40	95.00±1.06	95.08±1.48	95.46±1.34	95.08±1.69	95.46±1.31	95.42±1.33	95.36±1.56	95.58±1.33
DB7	91.70±1.64	91.76±2.04	92.29±1.63	92.02±2.13	92.36±1.69	92.26±2.16	92.47±1.66	92.61±1.55	92.53±1.57
DB8	95.97±1.08	96.22±0.83	96.10±0.98	96.32±0.99	96.17±1.13	96.47±1.24	96.19±1.03	96.13±0.84	96.26±1.17
DB9	92.76±1.40	92.78±1.14	92.74±1.15	93.04±1.37	93.02±1.66	92.99±1.34	92.80±1.26	92.93±1.53	92.78±1.00
DB10	92.50±1.33	92.23±1.53	92.42±1.50	92.70±1.54	92.72±1.39	92.72±1.46	92.56±1.68	92.76±1.38	93.11±1.41
DB11	92.22±1.50	91.97±1.64	92.19±1.61	92.65±1.72	92.97±1.47	92.70±1.41	92.66±1.51	92.55±1.57	92.44±1.30
DB12	88.39±1.01	88.56±1.03	88.27±0.96	88.94±1.09	88.79±1.26	88.80±1.31	88.88±0.96	88.93±1.10	88.89±1.15
DB13	82.61±1.48	82.70±1.86	82.80±1.65	83.45±1.69	83.56±1.52	83.57±1.66	83.27±1.66	83.05±1.73	83.36±1.59
DB14	82.17±1.54	82.16±1.82	82.18±1.52	82.50±1.69	82.79±1.66	82.71±1.61	82.93±1.64	82.98±1.58	82.83±1.40
20%F									
DB1	57.04±3.22	57.72±3.44	56.29±3.16	56.54±3.47	57.59±2.75	56.41±4.52	56.53±3.42	57.44±3.41	57.63±2.86
DB2	77.18±3.58	77.66±3.48	76.06±3.59	75.40±3.59	75.91±3.95	74.65±3.75	76.34±3.63	77.26±2.84	76.32±3.46
DB3	70.10±2.23	70.41±2.31	67.98±2.56	67.55±2.08	68.95±2.23	67.59±2.91	68.61±2.31	70.49±2.23	68.74±2.91
DB4	78.13±2.36	79.21±2.12	75.80±2.23	75.02±2.72	76.98±2.15	73.87±3.14	77.43±2.30	78.36±2.29	75.31±2.79
DB5	77.87±1.77	77.75±1.47	78.19±1.44	78.00±1.61	78.83±1.84	77.82±1.89	77.62±1.77	78.05±1.58	77.13±1.61
DB6	94.93±1.37	94.76±1.44	95.11±1.23	95.13±1.32	95.16±1.37	95.25±1.25	95.23±1.30	95.28±1.31	94.39±2.19
DB7	91.71±1.80	91.31±2.14	91.58±1.81	92.07±2.29	91.92±1.88	92.10±2.00	91.85±2.05	92.31±2.05	92.29±2.13
DB8	96.34±1.05	96.29±0.88	96.24±1.06	96.27±1.50	96.35±0.93	96.39±1.09	96.26±0.98	96.24±1.07	96.02±1.06
DB9	92.91±1.13	92.69±1.39	92.83±1.19	93.09±1.09	93.00±1.25	93.01±1.14	92.77±1.44	93.18±1.29	92.58±1.57
DB10	92.27±1.48	92.32±1.21	92.15±1.40	92.93±1.24	92.99±1.22	92.93±1.25	92.86±1.36	92.75±1.31	92.96±1.43
DB11	92.11±1.43	92.15±1.44	92.10±1.42	92.74±1.20	92.71±1.19	92.71±1.25	92.45±1.60	92.98±1.41	92.34±1.25
DB12	88.32±1.22	88.55±1.12	88.24±1.14	88.85±1.07	88.91±1.07	88.92±1.07	89.08±0.99	89.02±0.96	89.00±1.40
DB13	82.81±1.20	82.85±1.48	82.73±1.39	83.58±1.39	83.48±1.52	83.59±1.40	83.11±1.60	83.32±1.65	83.64±1.47
DB14	82.00±1.63	82.42±1.57	82.02±1.61	82.85±1.82	82.81±1.80	82.88±1.66	82.73±1.64	82.84±1.72	83.24±1.73
Ker									
DB1	58.83±2.76	59.45±2.99	58.92±2.40	58.85±2.74	58.79±3.38	57.80±2.99	59.56±2.61	59.61±2.59	58.53±2.84
DB2	78.44±2.84	78.87±2.81	76.68±2.91	76.84±2.87	76.68±3.26	75.52±3.23	78.44±3.14	78.55±3.13	76.80±3.27
DB3	70.78±2.41	71.84±2.04	69.73±2.08	68.92±1.75	70.12±2.19	68.36±1.92	70.42±2.02	71.85±2.04	70.16±1.88
DB4	77.52±1.80	78.35±1.97	75.62±2.14	76.03±2.38	76.42±2.42	75.62±2.48	77.08±2.46	77.95±2.14	75.68±2.44
DB5	77.98±1.51	77.86±1.57	78.17±1.61	78.71±1.21	79.24±1.48	78.79±1.90	77.52±1.47	77.95±1.33	76.78±1.58
DB6	95.05±1.15	94.98±1.28	95.28±1.03	95.28±0.80	95.63±0.88	95.37±0.87	96.36±1.40	96.13±1.12	96.83±1.50
DB7	91.62±1.52	91.56±1.54	91.82±1.41	92.47±1.60	92.59±1.60	92.37±1.52	92.50±1.82	92.41±1.78	92.69±1.44
DB8	96.41±1.09	96.34±1.06	96.38±1.08	96.64±0.83	96.62±0.74	96.57±1.04	96.48±1.01	96.44±0.99	96.61±1.05
DB9	92.95±1.11	92.85±1.25	92.73±1.19	93.29±0.96	93.33±1.00	93.28±0.97	93.39±1.12	93.56±1.06	93.23±1.29
DB10	92.54±1.05	92.56±1.14	92.55±0.99	93.19±1.00	93.23±1.00	93.29±0.89	93.32±1.11	93.23±1.05	93.56±1.28
DB11	92.05±1.21	92.30±1.16	92.28±1.22	93.02±1.15	93.26±1.02	92.92±1.18	92.78±1.35	93.28±1.32	92.70±1.22
DB12	88.53±1.07	88.54±1.24	88.29±0.99	88.82±0.99	88.94±0.89	88.72±0.96	89.37±1.02	89.25±1.07	89.39±1.52
DB13	82.82±1.31	82.73±1.47	82.87±1.32	83.57±1.32	83.94±1.32	83.56±1.36	84.38±1.42	84.46±1.52	84.57±1.51
DB14	82.10±1.51	82.28±1.61	82.01±1.30	83.33±1.22	83.54±1.32	83.10±1.19	83.68±1.91	84.15±1.51	83.93±1.50

Table 7.5: Classification accuracy in percent $\pm$ standard deviation for the 6 margin-constrained bag-of-paths methods and the 3 Poisson weighted surprisal distances with priors, obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing weight parameter or prior vector is highlighted in boldface for each method, each dataset and each feature set. The L1-normalized degree weights (the cBoPDegree and the cBoPHDegree) and priors (the PWSurpDegree) obtain most of the time the best results across all the feature sets.

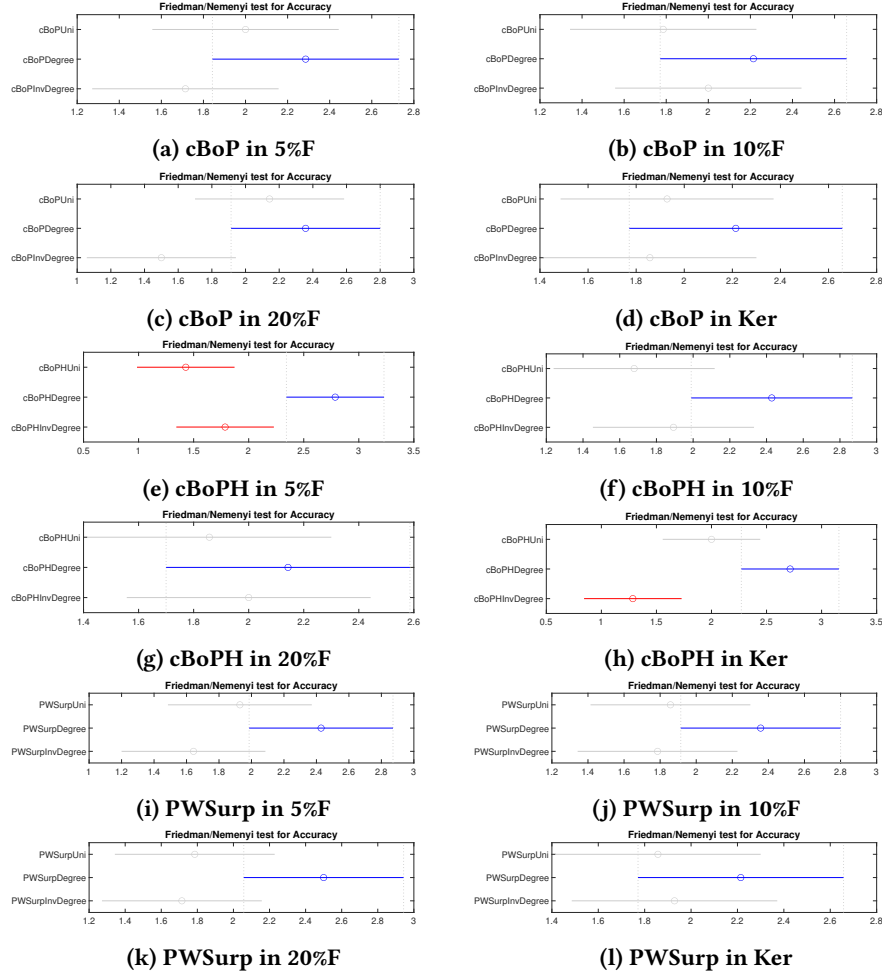


Figure 7.2: Mean ranks and 95% Nemenyi confidence intervals for the 6 margin-constrained bag-of-paths methods and the 3 Poisson weighted surprisal distances with priors across the 14 datasets for feature sets 5%F, 10%F, 20%F and Ker. Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are always obtained with the external parameter L1-normalized degree.

ed-ranks test is a non-parametric equivalent of the paired t -test. The null hypothesis (H_0) of this test is that the two compared classifiers perform equally well. Besides confirming the findings of the Friedman-Nemenyi tests, the Wilcoxon tests show that the cBoPInvDegree is outperformed by the cBoPUni in 20%F (p -value = 0.0494). The tests also highlight that the cBoPHDegree performs significantly better than the cBoPHUni in 10%F (p -value = 0.0419) and in Ker (p -value = 0.0085). Moreover, the cBoPHDegree (p -value = 0.0006) and the cBoPHUni (p -value = 0.0295) obtain results significantly superior to those of the cBoPHInvDegree in Ker. With regards to the PWSurpDegree, it outperforms the PWSurpUni (p -value = 0.0023) and the PWSurpInvDegree (p -value = 0.0419) in 20%F.

Overall, we can state that there are very few significant differences between the results obtained by the external parameters for the cBoP and the PWSurp. Regarding the cBoPH, the L1-normalized degree weights outperform the two other weights except for 20%F.

To avoid redundancy in our further comparisons, we only keep the results associated with the best external parameter according to the Nemenyi tests rankings presented in Fig. 7.2 for each method and each feature set. As already pointed out, the best external parameter is the L1-normalized degree weights and priors for all the methods and feature sets. Furthermore, to simplify the notation, we refer to the cBoPDegree, the cBoPHDegree and the PWSurpDegree as cBoP, cBoPH, and PWSurp in the following subsections.

7.4.2 Per-category analysis

We now discuss and present the results of each method by category, following the same methodology as in the previous subsection.

First, we have a look at the raw results of each method presented by category in Tables 7.6, 7.9, 7.12, 7.15, and 7.18. These tables contain the classification accuracy and standard deviation averaged over the 50 runs of each method by category, for the 14 datasets and the four extracted feature sets. In each table, the best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performing method overall (across all feature sets) for each dataset in each category.

Second, we compare the performance of the different methods by category across all the 14 datasets through a Friedman test followed by a Nemenyi post-hoc test [79]. The Friedman test allows us to determine if all the methods have the same average ranks. Whereas the Nemenyi test determines whether or not the performance of each method differs significantly from another.

Then, we perform multiple Wilcoxon signed-ranks tests for matched data [79, 293] to potentially discover other significant pairwise differences between the methods by category. For each test, we highlight in boldface the

p -values lower than our threshold α of 0.05 indicating that the two methods do not obtain equivalent results.

Finally, we select the two best-ranked methods, according to the Nemenyi tests rankings, in each category and in each feature extraction set to compare them in Subsection 7.4.3. All the selected methods are reported in Table 7.21.

Moreover, we summarize key findings of our analysis in boxes at the end of each category.

7.4.2.1 First category

In this subsection, we analyze the performance of the bag-of-paths methods composing the first category introduced in Subsection 7.1.1. For this purpose, we report the results of each method across all the data and the feature sets in Table 7.6.

As can be seen in Table 7.6, 11 of the best results over the 14 datasets are obtained by directly feeding the kernel to the SVM. Furthermore, eight of these 14 results are obtained by the PWSurp introduced in Chapter 6 and two by the cBoP re-derived in Chapter 5. Across all the feature sets, we observe that the cBoP seems to outperform all the methods on DB2 to DB4 except for 20%F and Ker where the PWSurp prevails on DB3. Regarding DB1, the PWSurp provides the best results for 5%F and Ker, while for 10%F and 20%F it is the RSPTsa and the RSP. The results on DB5 are also more contrasted: the FE performs best for 10%F and 20%F, whereas the BoPQ and the Sup are the highlighted methods respectively for 5%F and Ker. For the Newsgroup datasets (DB6-DB14), we can point out that the PWSurp, the SP, and the RSPTsa respectively dominate across all the feature sets on DB6, DB8, and DB10. Moreover, the BoPQ outperforms all the methods on DB7 except for 20%F where it is the RSPTsa. For the five remaining Newsgroup datasets, the best method is dataset-dependent and feature set-dependent. Nevertheless, we can underline that the PWSurp appears eight times among the 20 highlighted results. The other methods only appear respectively five times for the cBoPH and the FETsa, and two times for the RSPTsa. We can also point out that the performance of the different methods, except for the SP, are very close to each other, since all these methods are derived from the same framework.

As previously mentioned, we now perform Friedman tests [79] to globally rate the results of each method. The p -values of the Friedman tests are respectively 4.9×10^{-8} , 8.6×10^{-8} , 2.3×10^{-9} , and 1.2×10^{-7} for 5%F, 10%F, 20%F, and Ker. All these p -values are lower than the threshold α of 0.05, meaning that we can reject H_0 and that at least one classifier is significantly different from the others. As all the Friedman tests are positive, we perform Nemenyi tests [79] to identify the methods that obtain different results from each other. The results of the Nemenyi tests are illustrated in

Method →: Dataset ↓:	BoPQ	cBoP	cBoPH	PWSurp	FE	FETsa	RSP	RSPTsa	SP	Surp
5%F										
DB1	58.50±3.34	59.41±3.88	58.82±2.82	60.06±2.79	59.07±2.85	59.04±3.04	58.82±2.90	58.39±3.61	47.65±3.43	58.67±3.15
DB2	74.60±2.79	78.62±2.34	77.44±3.05	78.50±2.22	77.87±3.13	77.47±2.88	76.28±3.52	76.70±3.04	66.89±2.83	77.14±2.94
DB3	67.07±2.50	71.10±3.02	69.65±2.74	70.89±2.90	69.50±2.83	69.05±2.84	69.49±2.87	68.63±2.93	62.75±2.69	68.98±2.61
DB4	74.20±2.21	79.43±1.99	78.53±2.07	79.09±1.76	78.16±1.87	77.69±1.93	76.62±2.23	77.12±2.81	64.14±3.12	77.67±2.24
DB5	78.74±1.19	77.61±1.20	78.29±1.74	78.01±1.35	78.34±1.76	78.23±1.60	78.09±1.45	78.47±1.80	77.58±1.57	78.28±1.42
DB6	95.06±1.71	95.50±1.30	95.84±1.04	95.90±1.29	95.66±1.18	95.34±1.39	95.56±1.74	95.43±1.55	92.82±1.30	95.59±1.30
DB7	93.66±1.47	91.64±2.22	93.29±1.50	92.54±1.54	93.22±1.41	93.47±1.58	92.54±1.96	92.84±1.79	91.61±1.81	93.47±1.40
DB8	95.11±1.32	96.18±1.01	96.47±0.81	96.40±0.86	96.39±0.90	96.50±1.11	96.19±1.26	96.53±1.00	96.59±0.86	96.40±1.02
DB9	91.62±1.52	92.66±1.28	93.05±1.43	93.01±1.33	92.82±1.48	92.90±1.71	92.75±1.48	92.85±1.17	92.87±1.18	92.91±1.38
DB10	91.72±1.24	92.00±1.51	92.50±1.38	92.61±1.43	92.29±1.39	92.79±1.27	92.07±1.43	92.83±1.39	89.61±1.46	92.37±1.66
DB11	90.75±1.50	92.14±1.65	92.91±1.36	92.74±1.60	92.47±1.61	92.78±1.58	92.06±1.81	92.69±1.37	91.42±1.00	92.71±1.40
DB12	85.14±1.39	88.49±0.95	88.79±1.05	88.80±1.20	88.49±1.17	88.93±1.03	88.43±1.05	88.87±1.20	87.93±1.09	88.49±1.37
DB13	81.92±1.63	82.76±1.42	83.23±1.33	83.12±1.53	83.33±1.40	83.36±1.75	82.58±1.76	82.78±1.94	81.80±1.24	83.35±1.37
DB14	81.96±1.25	82.19±1.36	82.51±1.73	82.55±1.58	82.81±1.80	82.83±1.50	82.31±1.53	82.67±1.44	77.17±1.77	82.49±1.62
10%F										
DB1	57.10±3.29	58.08±3.25	57.98±3.19	57.93±3.34	57.79±3.10	57.63±3.34	57.95±3.95	58.11±3.71	48.29±3.27	57.00±3.89
DB2	73.80±2.83	78.05±3.14	76.82±3.03	78.03±2.86	76.81±3.27	76.17±3.11	76.48±2.49	75.61±3.27	65.86±3.25	76.04±3.46
DB3	66.16±2.64	70.46±2.58	68.29±2.46	69.09±3.14	68.63±2.15	68.03±2.69	69.59±2.78	68.76±3.27	62.40±2.69	67.75±2.54
DB4	73.49±2.20	79.61±2.01	77.67±2.17	78.79±2.01	77.59±1.97	77.36±2.45	77.06±2.05	77.04±2.20	63.79±2.59	76.52±2.11
DB5	78.42±1.37	77.76±1.31	78.95±1.42	78.09±1.35	79.10±1.38	78.64±1.24	78.83±1.36	78.93±1.58	77.57±1.56	78.63±1.55
DB6	93.73±2.30	95.00±1.06	95.08±1.69	95.36±1.56	95.06±1.30	95.16±1.44	95.17±1.43	95.06±1.59	93.15±1.19	94.90±1.73
DB7	93.01±1.47	91.76±2.04	92.36±1.69	92.61±1.55	92.32±1.64	92.36±1.84	91.87±1.73	91.98±2.36	90.78±2.17	92.41±1.52
DB8	95.07±1.20	96.22±0.83	96.17±1.13	96.13±0.84	96.29±1.02	96.37±0.98	96.36±1.03	96.41±1.05	96.72±0.74	96.28±1.14
DB9	92.31±1.98	92.78±1.14	93.02±1.66	92.93±1.53	93.08±1.25	93.12±1.08	92.63±1.53	93.09±1.27	92.83±1.09	92.69±1.46
DB10	91.32±1.69	92.23±1.53	92.72±1.39	92.76±1.38	92.23±1.46	92.83±1.71	92.10±1.56	93.09±1.49	89.72±1.55	92.25±1.67
DB11	89.26±1.76	91.97±1.64	92.97±1.47	92.55±1.57	92.23±1.48	92.59±1.65	92.21±1.27	92.47±1.41	91.72±1.12	92.76±1.46
DB12	83.36±1.60	88.56±1.03	88.79±1.26	88.93±1.10	88.71±0.96	88.90±1.25	88.45±1.30	89.06±1.23	88.01±0.81	88.89±1.07
DB13	80.71±1.87	82.70±1.86	83.56±1.52	83.05±1.73	83.18±2.02	83.40±1.90	82.42±1.96	83.02±2.14	81.55±1.35	83.36±1.88
DB14	79.95±1.64	82.16±1.82	82.79±1.66	82.98±1.58	82.70±1.96	83.03±1.66	82.58±1.63	83.26±1.51	77.52±1.80	82.59±1.71
20%F										
DB1	55.61±3.19	57.72±3.44	57.59±2.75	57.44±3.41	57.81±3.64	57.58±3.08	58.13±3.71	57.03±3.80	48.46±3.62	56.54±3.32
DB2	72.16±2.82	77.66±3.48	75.91±3.95	77.26±2.84	76.72±3.60	76.39±3.10	76.45±3.22	76.11±3.17	65.06±3.32	76.12±3.57
DB3	65.37±2.55	70.41±2.31	68.95±2.23	70.49±2.23	69.36±2.04	68.46±2.52	69.83±2.80	69.21±2.60	63.12±2.31	67.61±2.31
DB4	72.31±2.36	79.21±2.12	76.98±2.15	78.36±2.29	78.09±2.69	77.63±2.65	77.00±2.28	76.91±1.91	63.04±2.83	75.57±2.46
DB5	78.56±1.24	77.75±1.47	78.83±1.84	78.05±1.58	79.06±1.51	78.65±1.54	78.71±1.66	78.81±1.68	77.43±1.69	78.42±1.54
DB6	92.78±1.86	94.76±1.44	95.16±1.37	95.28±1.31	95.00±1.53	95.04±1.79	95.00±2.07	95.23±1.96	93.31±1.13	94.95±1.67
DB7	91.38±2.59	91.31±2.14	91.92±1.88	92.31±2.05	92.01±1.41	92.20±1.64	91.89±1.53	92.32±1.75	90.94±1.81	91.63±2.17
DB8	94.46±1.59	96.29±0.88	96.35±0.93	96.24±1.07	96.30±0.98	96.44±0.92	96.29±1.10	96.42±0.88	96.64±1.00	96.38±0.97
DB9	90.23±1.93	92.69±1.39	93.00±1.25	93.18±1.29	93.01±1.40	92.88±1.41	92.50±1.43	93.08±1.26	92.82±0.98	92.99±1.27
DB10	89.06±1.70	92.32±1.21	92.99±1.22	92.75±1.31	92.69±1.32	93.28±1.22	92.69±1.08	93.32±1.23	89.57±1.39	92.81±1.41
DB11	86.42±1.79	92.15±1.44	92.71±1.19	92.98±1.41	92.80±1.10	92.82±1.51	92.38±0.94	92.57±1.27	91.82±1.14	92.80±1.16
DB12	81.25±1.76	88.55±1.12	88.91±1.07	89.02±0.96	88.75±0.98	88.73±1.29	88.53±1.15	88.94±1.18	88.09±0.93	88.84±1.12
DB13	78.22±1.97	82.85±1.48	83.48±1.52	83.32±1.65	83.25±1.45	83.41±1.67	82.27±1.92	83.40±1.59	81.69±1.46	83.44±1.41
DB14	77.21±2.35	82.42±1.57	82.81±1.80	82.84±1.72	82.94±1.61	83.07±1.25	81.98±1.92	82.86±1.55	77.57±1.49	82.86±1.80
Ker										
DB1	42.54±1.02	59.45±2.99	58.79±3.38	59.61±2.59	58.51±2.81	57.55±3.68	58.30±3.76	57.88±3.91	48.58±3.68	58.71±2.75
DB2	53.87±3.24	78.87±2.81	76.68±3.26	78.55±3.13	77.11±3.17	75.96±3.54	76.88±2.93	75.44±3.14	64.50±3.57	76.91±3.05
DB3	65.71±1.41	71.84±2.04	70.12±2.19	71.85±2.04	69.46±2.02	68.69±2.39	69.78±3.14	68.93±3.18	63.08±2.55	69.05±2.20
DB4	66.27±2.04	78.35±1.97	76.42±2.42	77.95±2.14	77.26±2.24	76.69±2.61	75.69±2.32	75.47±2.32	63.96±2.93	76.25±2.32
DB5	79.05±1.19	77.86±1.57	79.24±1.48	77.95±1.33	78.95±1.51	78.83±1.62	78.99±1.21	78.90±1.41	77.57±1.97	79.58±1.52
DB6	95.54±1.31	94.98±1.28	95.63±0.88	96.13±1.12	95.40±1.21	95.68±1.06	95.66±1.50	95.70±1.46	93.41±1.08	95.45±0.88
DB7	92.85±1.43	91.56±1.54	92.59±1.60	92.41±1.78	92.27±1.61	92.83±1.53	92.04±1.60	92.31±1.63	91.25±1.56	92.32±1.97
DB8	96.28±0.84	96.34±1.06	96.62±0.74	96.44±0.99	96.62±0.78	96.63±0.86	96.47±0.96	96.67±0.76	96.79±0.78	96.72±0.84
DB9	93.39±0.82	92.85±1.25	93.33±1.00	93.56±1.06	93.24±0.92	93.32±0.94	92.80±1.32	93.31±1.03	92.84±1.00	93.29±1.02
DB10	93.02±0.99	92.56±1.14	93.23±1.00	93.23±1.05	92.84±1.05	93.38±1.27	92.65±1.16	93.62±0.99	89.75±1.28	93.26±0.97
DB11	92.15±1.24	92.30±1.16	93.26±1.02	93.28±1.32	92.78±1.16	92.96±1.27	92.42±1.27	92.74±1.28	91.89±0.79	92.98±1.21
DB12	87.82±1.22	88.54±1.24	88.94±0.89	89.25±1.07	88.78±1.02	88.95±1.05	88.57±1.03	88.94±1.06	88.16±0.91	88.87±0.96
DB13	83.42±1.32	82.73±1.47	83.94±1.32	84.46±1.52	83.26±1.20	83.44±1.57	82.66±1.64	83.64±1.56	81.66±1.38	83.70±1.27
DB14	82.43±1.57	82.28±1.61	83.54±1.32	84.15±1.51	83.17±1.17	83.19±1.35	82.43±1.75	82.90±1.22	77.62±1.46	83.52±1.20

Table 7.6: Classification accuracy in percent $\pm$ standard deviation for the methods of the first category obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. Most of the best results across all the feature sets are obtained by the PWSurp in Ker.

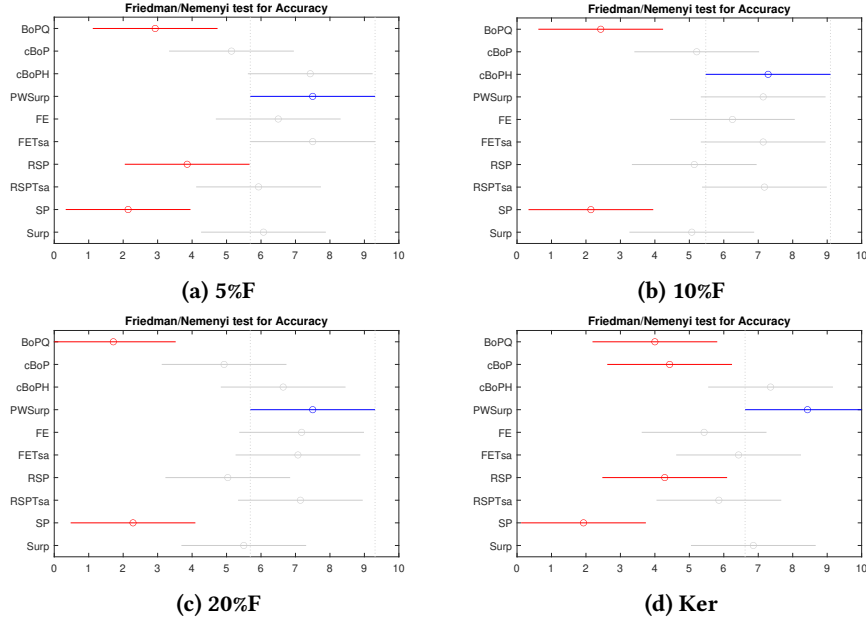


Figure 7.3: Mean ranks and 95% Nemenyi confidence intervals for the methods of the first category across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the PWSurp for all the feature sets, except for 10%F where it is the cBoPH.

Fig. 7.3. From Fig. 7.3, we can observe that the SP is outperformed by the PWSurp, the FETsa, the cBoPH, and the RSPTsa across all the feature sets. Moreover, the SP is also outpaced by the FE for all the feature sets except for Ker and by the Surp in 5%F and Ker. With regards to the BoPQ, it obtains results significantly inferior to those of the PWSurp, the FETsa, the cBoPH, and the RSPTsa across all the feature sets, except for Ker for the three last methods. The tests also highlight that the FE, the RSPTsa, and the Surp perform better than the BoPQ in 20%F and also in 10%F for the FE and the RSPTsa. Furthermore, we can notice that the PWSurp and the FETsa provide results significantly superior to those of the RSP in 5%F. The tests also show that the PWSurp outperforms the cBoP and the RSP in Ker.

We continue our analysis by reporting the p -values of the Wilcoxon tests [79] in Table 7.7 for 5%F and 10%F and in Table 7.8 for 20%F and Ker. First of all, we can point out that the Wilcoxon tests corroborate the previous observations obtained with the Nemenyi tests. In addition to these confirmations, we can see that the SP is outperformed by all the methods across all the fea-

Method $\rightarrow$: Method $\downarrow$:	BoPQ	cBoP	cBoPH	PWSurp	FE	FETsa	RSP	RSPTsa	SP	Surp
BoPQ	1.0000	0.0295	0.0012	0.0023	0.0006	0.0009	0.0107	0.0031	0.0419	0.0012
cBoP	0.0031	1.0000	0.6257	0.0031	0.6698	0.5416	0.4631	0.9515	0.0023	0.8077
cBoPH	0.0004	0.5830	1.0000	0.5416	0.3910	0.9515	0.0002	0.0245	0.0002	0.0085
Psurp	0.0006	0.2166	0.4263	1.0000	0.1353	0.5830	0.0004	0.1937	0.0004	0.0676
FE	0.0006	0.6257	0.2166	0.0419	1.0000	1.0000	0.0001	0.1353	0.0006	0.3258
FETsa	0.0006	0.6257	0.1531	0.5416	0.9515	1.0000	0.0052	0.0203	0.0004	0.0203
RSP	0.0012	0.5830	0.0419	0.0676	0.0906	0.1726	1.0000	0.0906	0.0006	0.0085
RSPTsa	0.0009	0.4263	0.6257	0.7148	0.7354	0.9515	0.1726	1.0000	0.0006	0.2676
SP	0.1726	0.0012	0.0004	0.0006	0.0004	0.0004	0.0006	0.0004	1.0000	0.0004
Surp	0.0012	0.8077	0.0017	0.0580	0.1353	0.0067	0.7609	0.1189	0.0006	1.0000

Table 7.7: The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the first category, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

Method $\rightarrow$: Method $\downarrow$:	BoPQ	cBoP	cBoPH	PWSurp	FE	FETsa	RSP	RSPTsa	SP	Surp
BoPQ	1.0000	0.0006	0.0001	0.0002	0.0001	0.0001	0.0001	0.0001	1.0000	0.0002
cBoP	0.5830	1.0000	0.5016	0.0580	0.3258	0.4631	0.5830	0.7609	0.0012	0.9032
cBoPH	0.0017	0.6257	1.0000	0.2166	0.3258	0.5416	0.2958	0.3575	0.0004	0.0494
Psurp	0.0067	0.0067	0.0295	1.0000	0.1531	0.3258	0.0295	0.2958	0.0004	0.0353
FE	0.0580	0.8552	0.0906	0.0085	1.0000	0.5416	0.0327	0.5830	0.0004	0.1040
FETsa	0.0040	0.9032	0.0906	0.0245	0.9515	1.0000	0.1353	0.8077	0.0004	0.0107
RSP	0.2958	0.9032	0.0009	0.0040	0.0295	0.3910	1.0000	0.1726	0.0006	0.8552
RSPTsa	0.0052	0.9032	0.0203	0.0203	0.6698	0.7148	0.4263	1.0000	0.0002	0.0134
SP	0.2166	0.0017	0.0002	0.0002	0.0002	0.0002	0.0006	0.0002	1.0000	0.0004
Surp	0.0052	0.7148	0.1726	0.0203	0.2166	0.3575	0.0134	0.0785	0.0002	1.0000

Table 7.8: The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the first category, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

ture sets, except for the BoPQ in 10%F, 20%F, and Ker. Regarding the BoPQ, it obtains results significantly inferior to those of all methods in 5%F, 10%F, and 20%F, except for the SP. Moreover, the BoPQ is outperformed by the cBoPH, the Surp, the RSPTsa, and the FETsa in Ker. The tests also highlight that the PWSurp performs significantly better than all the methods in Ker, the cBoP in 5%F, the FE in 10%F, and the Surp and the RSP in 20%F. On its side, the RSP is outperformed by the FE for all the feature sets except for 10%F, the Surp in 5%F and Ker, and the cBoPH in 5%F, 10%F and Ker. We can also notice that the cBoPH and the FETsa provide results significantly superior to those of the Surp and the RSPTsa in 5%F, as well as those of Surp in 10%F and 20%F. Furthermore, the cBoPH also performs better than the RSPTsa in Ker. Finally, the tests indicate that the RSPTsa has performance significantly superior to those of the Surp in 20%F.

Globally, the experiment shows that the SP and the BoPQ are outclassed by almost all methods through all the feature sets on the investigated datasets. Moreover the PWSurp, the cBoPH, and the FETsa stand out from the other

methods by consistently providing competitive results across all the feature sets.

Finally, following the Nemenyi tests rankings presented in Figs. 7.3a to 7.3d, the two best-ranked methods in each feature set selected for the comparison in Subsection 7.4.3 are: the PWSurp and the FETsa in 5%F, the cBoPH and the RSPTsa in 10%F, the PWSurp and the FE in 20%F, and the PWSurp and the cBoPH in Ker.

Summary - First category

- ▶ PWSurp, cBoPH, and FETsa stand out from the other methods
- ▶ SP and BoPQ are outclassed by almost all methods

7.4.2.2 Second category

We now focus on the covariance and correlation measures of the second category presented in Subsection 7.1.2. The results of these methods are reported in Table 7.9.

From the raw results of Table 7.9, we can observe that the best results on WebKB datasets (DB1-DB4) and DB7 are obtained by extracting 5% of the dominant eigenvectors, while the best results on the nine remaining datasets are obtained by directly using the kernel. The methods of the second category seem to work well with very little information as well as with all information on the investigated datasets. Across all the feature sets, we can see that the nCor performs best on DB6. Moreover, the nCor also provides the best results across three feature sets on DB5, DB10, DB11, and DB13. Regarding DB2, the nCorH seems to outperform all the methods across all the feature sets, except for 5%F where the nCor prevails. For the eight remaining datasets, the best method is dataset-dependent and feature set-dependent. Nevertheless, we can point out that the nCor and the nCorH appear 39 time across the 56 highlighted results.

As in the previous subsection, all the Friedman tests [79] are positive with respective p -values of 7.2×10^{-9} , 3.8×10^{-14} , 5.6×10^{-16} , and 5.9×10^{-10} for 5%F, 10%F, 20%F, and Ker. We then conduct Nemenyi tests [79] and report the outcomes in Fig. 7.4. First of all, we can observe that the nCor and the nCorH outperform all the covariance measures across all the feature sets, except for 5%F where there is no significant difference between the nCorH and the Cov. Regarding the CorH, it performs better than the nCov and the nCovH for all the feature sets except for 5%F. Furthermore, we can notice that the CorH obtains results superior to those of the CovH for all the feature sets except for 10%F. The tests also show that the Cor outperforms the CovH in 5%F, the nCov in 10%F and 20 %F, and the nCovH for all the feature sets except for 5%F.

Method →: Dataset ↓:	nCor	nCorH	nCov	nCovH	Cor	CorH	Cov	CovH
5%F								
DB1	59.30±2.96	58.93±3.79	57.10±2.81	56.38±2.68	59.73±2.73	59.64±2.85	58.44±2.30	58.24±2.75
DB2	76.29±2.97	75.85±3.13	72.78±2.84	73.05±2.90	75.62±3.51	75.85±3.12	73.47±2.93	73.09±1.80
DB3	68.37±2.74	68.42±2.64	69.02±3.12	69.87±2.51	68.06±2.52	67.98±2.61	68.35±2.15	67.73±2.09
DB4	77.44±1.93	77.87±1.78	71.32±2.28	71.34±2.18	77.01±2.15	77.31±2.18	76.19±1.62	74.93±1.08
DB5	79.04±1.03	78.73±1.20	76.87±1.91	76.80±1.78	78.35±1.20	78.76±1.33	78.04±1.20	78.24±1.07
DB6	95.04±1.54	94.59±1.45	93.98±2.07	93.81±1.89	93.91±1.59	94.66±1.64	94.54±1.70	93.74±1.68
DB7	93.39±1.52	93.09±1.86	90.30±3.53	90.68±2.62	92.47±1.61	93.24±1.60	93.39±1.51	91.88±1.74
DB8	95.73±1.19	95.82±1.23	95.16±1.42	94.73±2.02	95.91±1.09	95.68±0.97	94.26±2.00	94.50±1.55
DB9	92.09±1.40	92.14±1.37	90.63±2.11	90.65±2.17	91.26±1.75	91.62±1.52	91.31±1.45	90.58±1.61
DB10	92.07±1.33	92.10±1.31	89.72±1.67	89.72±2.09	90.48±1.45	91.50±1.27	91.16±1.57	89.94±1.65
DB11	91.22±1.40	91.04±1.37	89.58±1.97	90.22±1.64	90.73±1.66	90.84±1.52	90.15±1.54	89.49±2.05
DB12	85.51±1.62	85.63±1.52	86.05±1.20	86.10±1.37	85.70±1.81	85.57±1.68	84.76±1.39	83.51±1.53
DB13	81.63±1.90	81.32±1.68	80.80±1.68	80.59±1.67	81.46±1.84	81.24±1.64	80.30±1.52	77.83±1.93
DB14	81.68±1.51	81.83±1.36	79.00±1.76	78.89±2.09	81.51±2.08	81.40±1.45	81.06±1.56	79.84±1.58
10%F								
DB1	58.04±3.51	58.54±2.92	53.98±3.69	54.39±3.13	58.46±3.03	58.01±3.29	57.91±2.70	57.36±3.10
DB2	75.14±2.87	75.35±3.21	70.88±3.24	71.26±2.90	75.31±2.81	74.95±3.04	73.38±3.32	73.14±1.81
DB3	67.28±2.75	67.67±2.74	64.27±2.07	65.05±1.83	67.52±3.06	67.34±2.87	67.81±2.21	67.48±2.31
DB4	76.70±1.77	76.77±1.86	72.25±2.75	72.16±2.87	76.49±1.68	76.90±1.92	76.07±1.76	74.94±1.74
DB5	79.15±1.31	79.15±1.43	77.27±1.61	77.10±1.63	79.00±1.56	79.10±1.63	78.43±1.01	78.55±1.04
DB6	93.93±1.72	93.85±1.70	92.78±2.16	92.78±2.15	92.77±1.92	93.58±1.71	93.13±2.54	92.79±2.09
DB7	91.93±2.63	91.92±2.15	89.47±3.34	89.53±2.95	90.48±2.37	92.19±1.64	91.16±1.97	90.41±2.16
DB8	95.36±1.39	95.54±1.24	93.70±2.00	93.93±1.60	95.02±1.67	95.29±1.42	94.24±1.53	94.34±1.74
DB9	91.95±1.44	91.60±1.50	89.07±2.29	89.37±2.04	90.08±1.59	90.65±1.30	89.38±1.81	89.06±1.95
DB10	91.63±2.06	91.09±1.95	88.08±2.17	88.24±2.26	90.29±2.05	90.92±2.01	89.06±2.14	88.59±2.33
DB11	90.80±1.46	90.26±1.65	88.08±2.02	88.47±2.15	90.18±1.99	89.96±1.59	89.22±1.77	89.39±2.01
DB12	84.58±1.60	84.68±1.84	81.46±1.75	81.93±1.63	83.91±1.93	84.26±1.63	82.74±1.42	81.29±1.53
DB13	81.00±1.76	80.50±1.56	76.01±2.26	76.59±1.66	79.54±2.24	80.09±1.81	77.84±2.25	76.18±2.06
DB14	81.81±1.86	80.59±1.77	76.23±2.27	76.58±2.12	79.82±1.72	80.20±1.76	77.60±1.78	77.77±2.33
20%F								
DB1	58.06±3.18	57.87±3.06	51.83±4.54	53.14±3.83	57.54±2.86	58.74±2.85	55.46±3.84	56.78±3.11
DB2	75.22±3.42	75.60±2.74	70.60±3.23	70.05±4.09	74.56±2.62	74.99±2.50	72.76±3.24	72.87±2.11
DB3	67.91±2.06	68.59±2.41	64.95±2.37	65.05±2.50	68.13±2.13	67.61±2.50	67.40±2.23	66.99±2.25
DB4	76.13±1.66	76.06±1.77	72.26±3.08	72.13±3.30	76.12±1.77	76.39±1.78	75.31±1.88	75.22±1.31
DB5	79.53±1.22	79.55±1.10	77.32±1.37	77.30±1.34	79.27±1.15	79.41±1.13	78.48±1.24	78.55±1.19
DB6	92.86±1.72	92.37±1.98	90.88±3.51	90.81±3.58	91.67±1.73	92.06±2.07	92.12±2.21	91.33±3.00
DB7	91.19±2.11	90.81±2.28	87.63±3.83	88.05±3.49	89.61±2.28	90.63±1.94	89.76±2.64	88.72±2.66
DB8	94.67±1.27	94.86±1.25	92.99±2.63	93.25±2.19	94.19±1.85	94.51±1.29	93.48±1.88	92.52±2.25
DB9	90.50±1.77	90.23±1.80	87.41±2.48	87.25±2.36	89.04±1.79	89.55±1.68	88.40±1.87	87.34±2.59
DB10	90.19±1.80	90.05±1.75	86.36±2.21	85.99±2.53	88.68±1.78	89.46±1.72	87.83±1.74	86.87±2.03
DB11	89.41±1.90	89.26±2.10	86.70±2.44	86.79±2.34	88.41±1.88	88.58±1.81	87.87±2.06	87.22±1.65
DB12	83.86±1.57	83.74±1.47	77.99±1.88	78.37±1.89	82.54±1.69	82.95±1.37	80.65±2.06	77.43±1.80
DB13	78.60±2.13	77.84±2.21	72.81±2.21	73.50±2.28	76.50±2.06	76.94±1.73	73.53±2.50	72.03±2.24
DB14	78.97±1.85	78.26±1.64	73.43±2.31	73.51±2.37	76.63±1.96	77.19±1.91	74.74±1.89	74.10±1.93
Ker								
DB1	52.89±3.63	53.58±3.51	51.68±3.85	51.65±4.08	51.55±2.98	52.52±2.99	51.55±3.13	52.09±3.12
DB2	67.36±2.47	68.01±2.10	67.31±2.54	67.20±2.75	65.69±2.34	66.30±2.27	65.01±2.88	65.23±2.34
DB3	65.84±0.97	65.71±1.23	65.62±1.14	65.51±1.31	65.51±1.01	65.59±1.02	64.77±1.20	64.89±1.12
DB4	73.74±1.85	73.70±1.78	73.37±2.22	73.16±2.40	72.26±2.03	72.49±1.99	73.93±1.51	73.59±1.54
DB5	79.63±1.16	79.60±1.10	78.25±1.06	78.10±1.16	79.19±1.14	79.35±1.03	78.70±0.96	78.49±1.11
DB6	95.51±1.09	95.32±1.18	92.81±2.63	92.94±2.90	95.29±1.35	95.14±1.44	94.03±2.27	93.08±2.92
DB7	92.47±1.57	92.56±1.72	89.92±2.98	89.89±2.85	92.47±1.59	92.59±1.53	91.49±2.58	90.99±2.68
DB8	95.93±1.28	96.05±1.23	93.23±3.05	93.19±2.87	96.12±1.16	96.15±1.23	93.31±2.72	93.19±3.11
DB9	92.98±1.43	92.84±1.26	87.85±3.01	87.76±2.99	93.01±1.11	92.82±1.25	89.32±2.31	88.62±2.78
DB10	92.39±1.06	92.28±1.14	85.53±3.85	85.86±3.78	92.34±1.53	92.38±1.38	88.78±2.28	87.22±3.17
DB11	92.43±1.35	92.45±1.27	83.17±4.15	83.00±4.18	92.68±1.25	92.65±1.32	86.94±3.25	83.59±3.43
DB12	87.87±1.49	87.93±1.54	74.60±3.26	74.55±3.19	87.89±1.49	87.92±1.51	79.38±2.27	75.80±3.64
DB13	82.78±1.42	82.76±1.42	71.92±3.23	71.69±3.50	82.83±1.43	82.91±1.38	73.94±3.00	72.85±3.42
DB14	82.36±1.28	82.49±1.30	72.71±3.66	72.67±3.81	82.81±1.28	82.74±1.28	75.98±1.86	73.11±3.19

Table 7.9: Classification accuracy in percent $\pm$ standard deviation for the methods of the second category obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. The best results across all the feature sets are obtained either in Ker or in 5%F, but the methods providing these results differ from one dataset to another.

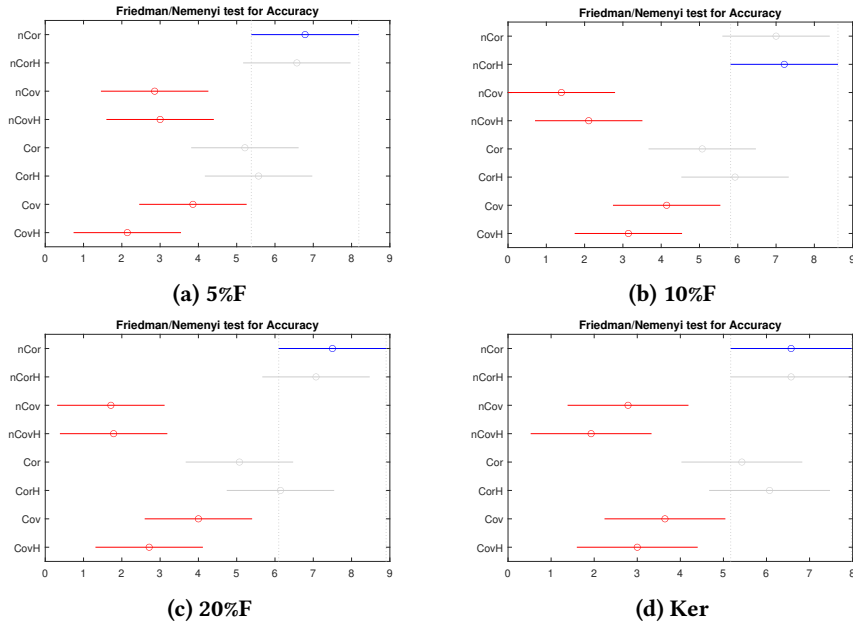


Figure 7.4: Mean ranks and 95% Nemenyi confidence intervals for the methods of the second category across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the nCor for all the feature sets, except for 10%F where it is the nCorH.

Method $\rightarrow$: Method $\downarrow$:	nCor	nCorH	nCov	nCovH	Cor	CorH	Cov	CovH
nCor	1.0000	0.2412	0.0009	0.0031	0.0134	0.0040	0.0002	0.0001
nCorH	0.5016	1.0000	0.0009	0.0023	0.0245	0.1353	0.0006	0.0001
nCov	0.0001	0.0001	1.0000	0.9032	0.0052	0.0031	0.0494	0.7609
nCovH	0.0001	0.0001	0.0034	1.0000	0.0085	0.0040	0.0906	1.0000
Cor	0.0085	0.0001	0.0002	0.0002	1.0000	0.0906	0.0906	0.0001
CorH	0.0245	0.0023	0.0001	0.0001	0.0419	1.0000	0.0031	0.0001
Cov	0.0004	0.0002	0.0001	0.0001	0.0040	0.0006	1.0000	0.0012
CovH	0.0002	0.0001	0.0012	0.0085	0.0002	0.0002	0.0052	1.0000

Table 7.10: The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the second category, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

Method $\rightarrow$: Method $\downarrow$:	nCor	nCorH	nCov	nCovH	Cor	CorH	Cov	CovH
nCor	1.0000	0.1353	0.0001	0.0001	0.0004	0.0067	0.0001	0.0001
nCorH	0.8077	1.0000	0.0001	0.0001	0.0002	0.0166	0.0001	0.0001
nCov	0.0001	0.0001	1.0000	0.3575	0.0001	0.0001	0.0001	0.0295
nCovH	0.0001	0.0001	0.0676	1.0000	0.0001	0.0001	0.0001	0.0785
Cor	0.2958	0.3910	0.0085	0.0052	1.0000	0.0052	0.0006	0.0001
CorH	0.2676	0.3910	0.0031	0.0017	0.1531	1.0000	0.0002	0.0001
Cov	0.0002	0.0002	0.0245	0.0134	0.0023	0.0023	1.0000	0.0203
CovH	0.0001	0.0001	0.0676	0.0676	0.0023	0.0012	0.0107	1.0000

Table 7.11: The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the second category, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

Again, to refine our analysis and potentially provide more information about the relative performance of the methods, we perform multiple Wilcoxon signed-rank tests [79] and report the results in Tables 7.10 and 7.11. The tests show that the nCor and the nCorH outperform all the methods except each other in 5%F, 10%F, and 20%F. With regards to the CorH, it performs better than the Cov, the nCovH, and the nCov for all the feature sets. Moreover, the CorH obtains results significantly superior to those of the Cor and the CovH in 10%F. The tests also highlight that the Cor outperforms the nCov, the CovH, and the nCovH for all the feature sets. Furthermore, the Cor performs better than the Cov for all the feature sets except for 5%F. We can also notice that the Cov provides results significantly superior to the nCov and the CovH for all feature sets, and to the nCovH in 10%F, 20%F, and Ker. On its side, the CovH outperform the nCov in 10%F and 20%F, and the nCovH in 10%F. Finally, the tests indicate that the nCovH has outcomes significantly superior to those of the nCov in 10%F.

In summary, the experiment on the investigated datasets demonstrates that the nCor and the nCorH outperform all the methods across all the feature sets, except the Cor and the CorH in Ker. Moreover, the nCov, the nCovH, and the CovH are outclassed by almost all methods through all the feature sets on the investigated datasets. The experiment also highlights that the correlation measures perform better than the covariance measures across all the feature sets.

Finally, not surprisingly, the two best-ranked methods in each feature set selected for the comparison in Subsection 7.4.3 are the nCor and the nCorH according to the Nemenyi tests rankings illustrated in Figs. 7.4a to 7.4d.

Summary - Second category

- ▶ nCor and nCorH are the best performing methods
- ▶ nCov, nCoH, and CovH are outclassed by almost all methods
- ▶ Correlation measures outperform covariance measures

7.4.2.3 Third category

We now examine the performance of the nine similarity and dissimilarity measures directly derived from the adjacency matrix $\mathbf{A}$ and the Laplacian matrix $\mathbf{L}$ of the third category introduced in Subsection 7.1.3. The performance of each of these methods across all the data and the feature sets are presented in Table 7.12.

First of all, we can observe from Table 7.12 that the LF stands out from the other methods by appearing 51 times among the 56 highlighted results and providing 13 of the best results over 14 datasets across all the features sets. Moreover, 10 of the best results are obtained by directly feeding the kernel to the SVM. Another observation is that the Leicht and the RL obtain slightly lower results in Ker compared to their performance in the other feature sets. The results provided by the Leicht in Ker are at best equivalent to the prior of the majority class of each dataset. The additional information provided by the kernel does not seem to be optimally exploited by these methods.

Then, in order to globally rate the results of each method, we investigate Friedman-Nemenyi tests [79] across all the datasets in each feature set. For the Friedman tests, we obtain p -values of 1.4×10^{-11} for 5%F, 5.1×10^{-10} for 10%F, 9.5×10^{-13} for 20%F, and 1.3×10^{-14} for Ker that are all lower than our threshold α of 0.05 indicating that at least one method is significantly different from the others. To identify these methods, we then carry out several Nemenyi tests and presented the results in Fig. 7.5.

Method →: Dataset ↓:	A	Katz	Leicht	LPI	LF	MRL	Q	RL
5%F								
DB1	59.23±3.34	58.42±3.25	46.78±3.37	58.71±3.19	59.99±2.32	59.54±3.25	56.91±3.31	43.97±2.19
DB2	72.41±2.83	71.43±3.04	66.29±3.59	73.34±2.92	78.35±2.18	75.44±2.58	70.67±2.73	55.72±3.68
DB3	69.21±2.06	67.96±1.97	63.06±2.59	69.16±2.00	71.22±1.87	69.28±2.11	67.76±2.50	63.02±2.78
DB4	75.17±1.71	73.93±0.87	68.41±2.02	75.03±2.47	78.97±2.25	75.24±1.83	73.98±1.81	60.30±3.86
DB5	77.10±1.54	77.18±1.93	76.25±2.01	77.23±1.73	78.50±1.82	78.06±1.68	76.45±1.47	79.05±1.12
DB6	93.36±1.82	93.90±1.85	85.64±2.11	94.16±1.77	94.79±2.16	94.72±2.24	93.53±1.50	95.40±1.36
DB7	90.27±2.67	91.05±2.51	78.10±4.82	91.16±1.92	92.84±1.55	92.29±1.74	89.94±2.37	91.26±1.41
DB8	93.99±1.48	93.97±1.96	88.07±2.82	93.87±1.54	95.37±1.38	94.79±1.27	94.42±1.48	94.43±0.90
DB9	89.85±2.00	90.55±1.82	79.55±2.23	89.59±1.84	92.65±1.31	91.00±2.13	90.38±1.99	90.41±1.39
DB10	89.39±1.98	89.67±2.13	77.65±2.41	88.97±1.63	91.51±1.57	90.73±1.63	89.81±1.71	89.62±1.24
DB11	88.66±2.08	89.08±2.22	77.39±2.42	90.00±1.94	91.74±1.96	89.84±1.83	89.69±2.18	89.68±1.41
DB12	83.68±2.29	84.66±1.64	63.13±2.75	83.47±1.83	86.58±1.57	84.22±1.53	84.10±2.24	78.55±1.47
DB13	77.81±2.23	78.59±2.17	63.43±2.44	77.32±1.84	82.07±2.08	80.45±1.78	78.58±1.95	75.47±1.93
DB14	79.31±1.86	79.43±1.84	64.65±2.13	77.50±2.39	82.65±1.31	80.99±1.40	79.56±1.84	74.24±1.97
10%F								
DB1	59.52±3.33	59.07±2.79	49.94±3.68	58.71±3.40	58.44±3.26	58.12±2.43	57.31±3.58	50.97±3.41
DB2	74.96±3.28	73.43±3.29	65.77±2.73	73.64±3.02	78.44±2.31	74.61±2.99	73.20±3.74	65.40±3.57
DB3	68.91±2.22	68.16±2.33	60.09±3.26	69.26±2.53	71.14±2.26	67.12±2.75	67.11±2.56	62.70±2.63
DB4	75.52±1.72	74.20±1.08	64.07±2.45	75.94±1.84	79.14±2.06	73.53±2.32	73.71±2.56	73.48±2.21
DB5	77.26±1.69	77.42±1.91	75.53±1.49	77.43±1.65	79.11±1.38	76.29±1.99	75.75±1.87	77.88±1.01
DB6	92.23±2.18	93.46±2.23	91.77±1.92	93.38±1.79	93.94±2.16	92.69±2.18	92.44±2.12	93.44±1.52
DB7	89.54±2.65	90.51±2.21	86.84±2.92	89.77±2.18	91.43±2.13	91.83±2.34	88.84±3.05	91.63±1.78
DB8	93.98±2.17	94.59±1.74	90.21±3.00	94.78±1.48	95.36±1.22	92.86±1.82	94.58±1.79	93.33±1.15
DB9	89.03±2.01	89.44±1.90	85.28±2.10	89.80±2.02	91.96±1.70	90.60±2.00	89.55±2.15	89.83±1.27
DB10	88.85±2.16	89.49±1.98	86.72±2.33	89.36±1.85	91.93±1.37	90.16±1.80	89.30±2.35	90.28±1.68
DB11	88.17±2.14	89.47±1.82	84.06±2.72	89.54±1.98	90.96±2.05	88.38±1.65	89.13±2.45	90.83±1.43
DB12	83.16±2.33	84.09±1.82	76.43±1.48	84.23±1.58	86.50±1.65	82.39±1.52	83.90±2.06	83.76±1.08
DB13	77.49±2.61	78.68±1.86	68.45±2.19	78.73±1.66	81.94±2.06	79.64±1.44	78.36±2.01	77.50±1.95
DB14	79.88±1.87	80.17±1.70	73.47±2.06	78.13±2.40	82.29±1.72	80.17±1.49	80.06±2.01	80.56±1.60
20%F								
DB1	59.19±3.28	59.12±2.70	49.32±2.76	58.38±4.23	58.50±2.92	57.12±2.75	58.01±3.09	55.39±3.31
DB2	75.35±2.77	74.29±2.77	62.75±3.58	73.62±2.85	78.53±2.65	73.34±3.17	74.58±2.81	74.63±2.80
DB3	69.10±1.98	68.41±2.25	57.79±2.75	69.15±2.27	71.09±2.39	67.61±2.75	67.68±2.41	65.89±2.15
DB4	76.04±1.99	74.37±1.45	66.36±2.78	76.06±1.72	79.43±2.03	69.46±2.97	74.66±2.49	74.81±1.57
DB5	77.21±2.73	77.46±1.98	76.23±1.45	77.58±1.53	79.09±1.34	75.19±1.89	76.25±1.93	78.24±1.36
DB6	91.69±2.24	93.11±2.55	92.28±2.08	94.02±1.51	94.17±1.97	89.15±2.47	91.83±1.94	87.69±2.09
DB7	88.71±2.76	90.67±2.05	88.17±3.15	90.28±1.88	91.66±1.60	87.31±2.82	88.73±2.44	87.17±2.18
DB8	93.68±1.74	93.96±1.90	92.11±2.00	94.97±1.42	95.28±1.61	90.38±2.02	94.02±1.44	90.40±1.38
DB9	88.15±1.99	88.93±1.76	88.20±1.97	90.83±1.73	92.08±1.51	85.82±2.72	88.54±1.74	84.48±1.51
DB10	88.90±1.85	89.32±1.95	88.02±1.63	90.24±1.74	91.97±1.55	86.54±1.79	88.85±2.44	85.83±2.14
DB11	87.81±2.45	88.74±1.87	88.53±2.18	90.53±1.69	91.72±1.68	84.07±2.32	88.30±2.50	87.76±1.50
DB12	84.33±1.84	84.76±1.68	80.77±1.83	85.31±1.63	86.85±1.48	80.02±1.55	84.33±1.83	81.41±1.27
DB13	78.01±1.65	78.65±1.49	73.26±2.01	79.81±1.63	82.39±1.74	76.62±1.90	78.31±1.58	75.52±1.94
DB14	78.85±1.82	79.56±1.49	76.16±2.02	79.37±2.17	82.22±1.78	75.86±2.02	79.00±1.62	75.83±1.65
Ker								
DB1	59.53±3.26	59.27±2.60	41.91±0.06	58.54±3.92	59.96±2.65	41.91±0.09	58.17±2.94	41.91±0.06
DB2	75.79±2.98	74.39±2.78	48.80±0.23	73.81±3.00	79.46±2.72	49.30±1.95	74.88±2.58	48.80±0.23
DB3	69.41±2.00	68.51±2.31	39.17±0.05	69.34±2.30	71.87±1.99	64.95±1.39	68.12±2.19	63.04±4.92
DB4	76.11±1.85	74.34±1.47	44.54±0.08	76.07±1.75	79.48±1.95	50.12±0.91	74.84±2.19	44.54±0.08
DB5	77.35±1.74	77.43±1.99	52.50±7.36	77.42±1.63	79.19±1.36	77.82±2.42	76.23±2.31	66.40±6.81
DB6	91.55±2.34	93.58±2.12	51.36±5.43	94.28±1.45	94.95±1.72	94.46±1.64	91.86±1.93	88.47±5.69
DB7	89.23±2.06	91.31±1.95	51.88±7.76	90.81±1.82	92.27±1.54	91.70±1.70	89.30±2.09	85.65±6.92
DB8	93.90±1.76	94.27±1.59	51.52±5.97	95.11±1.43	95.53±1.56	95.02±1.26	94.19±1.64	80.55±10.87
DB9	88.82±1.38	89.04±1.89	36.94±8.60	91.00±1.60	92.70±1.13	91.91±1.51	88.93±1.85	72.86±10.08
DB10	89.33±1.55	89.68±1.57	34.16±4.16	90.70±1.66	92.35±1.33	91.52±1.50	89.48±1.59	74.14±8.21
DB11	88.61±1.70	89.43±1.63	33.58±0.12	90.85±1.64	92.36±1.43	91.26±1.56	88.80±1.80	63.47±6.31
DB12	84.69±1.67	85.18±1.80	20.76±3.30	85.51±1.61	87.95±1.17	86.26±1.67	84.65±1.67	57.99±6.14
DB13	78.21±1.87	79.06±1.58	21.82±4.87	80.14±1.48	82.69±1.84	80.94±2.03	78.40±1.83	52.49±8.17
DB14	79.43±1.75	79.86±1.62	20.54±1.67	79.75±2.04	82.50±1.72	80.94±1.71	79.47±1.69	62.53±5.77

Table 7.12: Classification accuracy in percent $\pm$ standard deviation for the methods of the third category obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. The LF stands out from the other methods by providing most of the highlighted results.

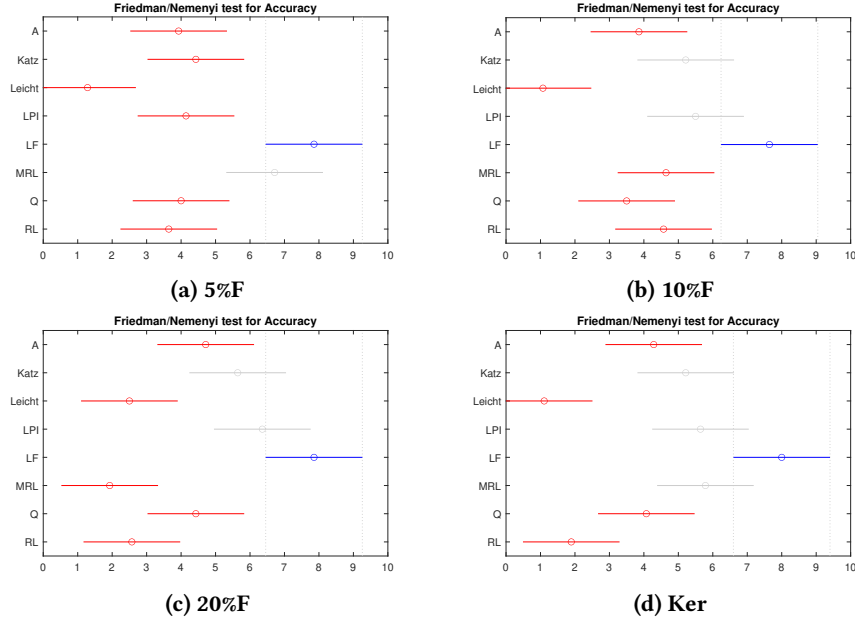


Figure 7.5: Mean ranks and 95% Nemenyi confidence intervals for the methods of the third category across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the LF for all the feature sets.

The Nemenyi tests indicate that the Leicht performs lower than the LF, the Katz, the MRL and the LPI across all the feature sets except the MRL in 20%F. In addition, the Leicht obtains results inferior to those of the RL in 10%F, as well as the A and the Q in Ker. We can also notice that the RL is outperformed by the LF for all the feature sets, the LPI in 20%F and Ker, the Katz in Ker, and the MRL in 5%F and Ker. With regards to the MRL, it performs lower than the LF in 10%F and 20%F, as well as the LPI and the Katz in 20%F. The tests also highlight that the LF has performance significantly superior to those of the Katz and the LPI in 5%F. Furthermore, the LF outperforms the Q and the A for all the feature sets.

To obtain more precise information regarding the relative performance of the methods, we also run Wilcoxon signed-ranks tests [79] and listed the p -values in Tables 7.13 and 7.14. Besides confirming the previous observations, the Wilcoxon tests show that the LF provides results significantly superior to those of all the methods across all the feature sets. The tests also highlight that the Leicht is outperformed by all the methods across all the feature sets except the MRL and the RL in 20%F. On its side, the MRL performs better than

Method →: Method ↓:	A	Katz	Leicht	LPI	LF	MRL	Q	RL
A	1.0000	0.9032	0.0001	0.8552	0.0001	0.0001	0.6257	0.1353
Katz	0.3910	1.0000	0.0001	1.0000	0.0001	0.0002	0.1189	0.0906
Leicht	0.0001	0.0001	1.0000	0.0001	0.0001	0.0001	0.0001	0.0245
LPI	0.2676	0.6257	0.0001	1.0000	0.0001	0.0004	0.5016	0.1040
LF	0.0002	0.0004	0.0001	0.0002	1.0000	0.0001	0.0001	0.0006
MRL	1.0000	0.5830	0.0001	0.5016	0.0004	1.0000	0.0001	0.0040
Q	0.4263	0.0004	0.0001	0.0085	0.0001	0.2676	1.0000	0.0676
RL	1.0000	0.3910	0.0002	0.3575	0.0004	0.6698	0.9032	1.0000

Table 7.13: The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the third category, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

Method →: Method ↓:	A	Katz	Leicht	LPI	LF	MRL	Q	RL
A	1.0000	0.2166	0.0031	0.0166	0.0002	0.0001	0.5016	0.0006
Katz	0.3258	1.0000	0.0001	0.0245	0.0002	0.0001	0.0017	0.0017
Leicht	0.0001	0.0001	1.0000	0.0001	0.0001	0.8077	0.0017	0.4263
LPI	0.0419	0.0353	0.0001	1.0000	0.0001	0.0001	0.0009	0.0006
LF	0.0001	0.0001	0.0001	0.0001	1.0000	0.0001	0.0001	0.0001
MRL	0.9032	0.9032	0.0001	0.9515	0.0001	1.0000	0.0001	0.9032
Q	0.6257	0.0107	0.0001	0.0012	0.0001	0.9032	1.0000	0.0040
RL	0.0001	0.0001	0.0010	0.0001	0.0001	0.0001	0.0001	1.0000

Table 7.14: The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the third category, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

all the methods except the LF in 5%F. We can also notice that the LPI obtains results superior to those of the A and the Katz in 20%F and Ker. Furthermore, the LPI and the Katz outperform the Q for all the feature sets except for 5%F. Finally, we can notice that the A and the Q outperform the MRL in 20%F and the RL in 20%F and Ker.

Overall, we can state that the LF is the method providing the best performance in our experiment across all the data and the feature sets. We can also point out that the MRL, the LPI, and the Katz obtain competitive results on the investigated dataset, but these results are not consistent across all the feature sets. Moreover, the experiment shows that the Leicht is outclassed by all the methods through almost all the feature sets. Regarding the two baseline methods, the A and the Q, we can state that overall they are not outclassed by the others methods and achieve consistent results across all the feature sets. Indeed, only the LF manages to significantly outperform them for all the feature sets.

Finally, following the Nemenyi tests rankings presented in Figs. 7.5a to 7.5d, the two best-ranked methods in each feature set selected for the com-

parison in Subsection 7.4.3 are: the LF for all the feature sets, the MRL in 5%F and in Ker, and the LPI in 10%F and 20%F.

Summary - Third category

- ▶ LF is the method providing the best performance
- ▶ Leicht is outclassed by all the methods

7.4.2.4 Fourth category

In this subsection, we analyze the performance of the five kernels derived from the commute time distance, composing the fourth category introduced in Subsection 7.1.4. For this purpose, we report the results of each method across all the data and the feature sets in Table 7.15.

As can be seen in Table 7.15, the best results over the 14 datasets are distributed across all the feature sets, but predominantly in 20%F and Ker. Moreover, the SCCT obtains nine of these 14 results and provides 40 of the 52 highlighted outcomes through all the feature sets. The SCCT also provides the best results on the Newsgroup datasets (DB6-DB14) across all the feature sets, except for DB8 where the SCT prevails in 20%F. Regarding DB5, the CT outperforms all the methods except for 10%F where it is the CCT. For the WebKB datasets (DB1-DB4), the results are more contrasted, except for DB1 where the SCCT dominates across all the feature sets. Whereas on DB2 to DB4, the highlighted results are shared between the SCCT and the SCT, except for DB3 where the CCT prevails in 20%F. We can also point out that the additional information provided by the kernel decreases the performance of the CT and the RCTK.

As previously, to globally rate the results of each method, we conduct some Friedman tests [79], which are all positive with respective p -values of 5.3×10^{-9} for 5%F, 1.1×10^{-9} for 10%F, 2.0×10^{-9} for 20%F, and 2.6×10^{-9} for Ker. We then perform Nemenyi tests [79] and report the results in Fig. 7.6. We can observe that the SCCT outperforms the RCTK, the CCT, and the CT across all the feature sets, except for 20%F where there is no significant difference between the SCCT and the CT. With regards to the SCT, it also obtains results superior to those of the RCTK across all the feature sets. In addition, the SCT performs better than the CCT in 20%F and the CT in Ker.

Then, to potentially discover other significant pairwise differences between the methods, we perform multiple Wilcoxon signed-rank tests [79] and report the results in Tables 7.16 and 7.17. First of all, we can observe that the RCTK is outperformed by all the methods for all the feature sets. The tests also highlight that the SCCT provides results significantly superior to those of

Method $\rightarrow$: Dataset $\downarrow$:	CT	CCT	RCTK	SCT	SCCT
5%F					
DB1	54.10 $\pm$ 3.91	59.78 $\pm$ 2.24	49.37 $\pm$ 3.22	59.18 $\pm$ 2.63	59.96$\pm$2.61
DB2	67.24 $\pm$ 3.40	76.34 $\pm$ 1.94	63.00 $\pm$ 3.06	76.72$\pm$3.34	76.56 $\pm$ 3.05
DB3	65.59 $\pm$ 2.10	66.11 $\pm$ 1.89	60.30 $\pm$ 2.15	69.24$\pm$2.53	68.13 $\pm$ 2.58
DB4	71.88 $\pm$ 3.53	73.82 $\pm$ 1.84	64.25 $\pm$ 2.55	77.21 $\pm$ 2.1	77.78$\pm$2.06
DB5	78.90$\pm$1.31	77.52 $\pm$ 1.50	58.65 $\pm$ 2.62	78.27 $\pm$ 1.22	77.45 $\pm$ 1.35
DB6	96.64 $\pm$ 0.73	93.38 $\pm$ 1.83	81.38 $\pm$ 2.84	96.14 $\pm$ 1.37	97.11$\pm$0.59
DB7	91.17 $\pm$ 1.34	91.41 $\pm$ 1.62	69.33 $\pm$ 3.65	92.30 $\pm$ 1.68	93.00$\pm$1.10
DB8	94.08 $\pm$ 0.68	93.80 $\pm$ 2.04	87.81 $\pm$ 1.55	95.81 $\pm$ 1.32	95.86$\pm$1.04
DB9	90.52 $\pm$ 1.32	89.41 $\pm$ 1.96	80.19 $\pm$ 2.03	93.34 $\pm$ 1.27	93.90$\pm$0.87
DB10	89.92 $\pm$ 1.23	89.48 $\pm$ 1.77	75.51 $\pm$ 2.09	91.48 $\pm$ 1.83	93.24$\pm$1.11
DB11	89.63 $\pm$ 1.21	90.54 $\pm$ 1.25	70.69 $\pm$ 1.94	92.53 $\pm$ 1.30	92.93$\pm$0.92
DB12	78.50 $\pm$ 1.55	84.19 $\pm$ 1.53	70.32 $\pm$ 2.24	86.96 $\pm$ 1.18	87.84$\pm$0.92
DB13	76.04 $\pm$ 1.61	79.44 $\pm$ 1.98	66.45 $\pm$ 1.25	81.74 $\pm$ 1.63	82.74$\pm$1.13
DB14	75.25 $\pm$ 1.78	79.59 $\pm$ 1.57	59.16 $\pm$ 1.94	82.35 $\pm$ 1.53	83.15$\pm$1.41
10%F					
DB1	53.51 $\pm$ 3.28	59.15 $\pm$ 1.87	46.65 $\pm$ 3.29	58.52 $\pm$ 3.16	59.48$\pm$2.89
DB2	70.26 $\pm$ 2.71	76.03 $\pm$ 2.67	64.11 $\pm$ 3.94	76.41 $\pm$ 3.32	76.60$\pm$2.88
DB3	63.53 $\pm$ 2.56	67.53 $\pm$ 1.90	55.53 $\pm$ 2.21	68.89$\pm$2.58	68.68 $\pm$ 2.16
DB4	73.00 $\pm$ 1.36	75.58 $\pm$ 2.18	59.42 $\pm$ 2.28	77.61 $\pm$ 2.22	77.83$\pm$2.12
DB5	77.35 $\pm$ 1.31	78.50$\pm$1.75	63.19 $\pm$ 3.20	78.01 $\pm$ 1.42	77.38 $\pm$ 1.26
DB6	93.30 $\pm$ 1.64	89.59 $\pm$ 2.93	81.51 $\pm$ 3.13	95.83 $\pm$ 1.26	97.00$\pm$0.80
DB7	91.79 $\pm$ 1.79	88.67 $\pm$ 2.56	71.18 $\pm$ 4.48	92.38 $\pm$ 1.44	92.87$\pm$1.20
DB8	93.53 $\pm$ 1.17	91.56 $\pm$ 2.07	86.98 $\pm$ 1.48	95.91 $\pm$ 0.93	96.07$\pm$0.73
DB9	89.84 $\pm$ 1.26	85.66 $\pm$ 1.93	79.15 $\pm$ 2.25	93.18 $\pm$ 1.12	93.84$\pm$0.74
DB10	90.28 $\pm$ 1.56	86.44 $\pm$ 2.08	71.90 $\pm$ 2.53	91.95 $\pm$ 1.10	93.31$\pm$0.96
DB11	90.87 $\pm$ 1.33	88.32 $\pm$ 2.13	72.39 $\pm$ 2.64	92.37 $\pm$ 1.36	93.09$\pm$0.89
DB12	83.95 $\pm$ 1.16	81.10 $\pm$ 2.00	67.75 $\pm$ 1.89	86.61 $\pm$ 1.46	88.17$\pm$0.97
DB13	77.20 $\pm$ 1.74	77.07 $\pm$ 1.85	62.61 $\pm$ 1.94	81.93 $\pm$ 1.69	82.84$\pm$1.28
DB14	80.50 $\pm$ 1.82	77.43 $\pm$ 2.05	58.49 $\pm$ 2.03	82.18 $\pm$ 1.66	83.39$\pm$1.27
20%F					
DB1	56.19 $\pm$ 2.87	58.04 $\pm$ 2.51	48.94 $\pm$ 5.31	58.35 $\pm$ 3.48	59.30$\pm$2.95
DB2	75.18 $\pm$ 1.91	72.99 $\pm$ 2.16	66.43 $\pm$ 2.51	76.22 $\pm$ 3.21	76.86$\pm$3.02
DB3	65.59 $\pm$ 1.66	71.00$\pm$1.78	63.20 $\pm$ 2.25	68.92 $\pm$ 2.61	68.59 $\pm$ 2.09
DB4	74.48 $\pm$ 1.71	73.10 $\pm$ 1.70	55.34 $\pm$ 2.85	77.85$\pm$2.44	77.49 $\pm$ 2.14
DB5	78.38$\pm$1.48	77.05 $\pm$ 1.53	61.57 $\pm$ 3.11	78.18 $\pm$ 1.52	77.53 $\pm$ 1.22
DB6	87.15 $\pm$ 2.29	84.45 $\pm$ 2.53	76.17 $\pm$ 5.09	95.75 $\pm$ 1.31	96.94$\pm$0.73
DB7	86.75 $\pm$ 2.48	82.06 $\pm$ 2.48	63.33 $\pm$ 6.05	91.85 $\pm$ 1.62	92.79$\pm$1.15
DB8	90.46 $\pm$ 1.53	87.19 $\pm$ 2.34	84.70 $\pm$ 2.84	96.09$\pm$1.00	96.07 $\pm$ 0.83
DB9	84.03 $\pm$ 1.95	79.72 $\pm$ 2.14	77.03 $\pm$ 2.25	93.39 $\pm$ 1.03	93.90$\pm$0.62
DB10	85.74 $\pm$ 1.89	81.60 $\pm$ 2.88	66.88 $\pm$ 2.74	92.14 $\pm$ 1.02	93.31$\pm$0.86
DB11	87.42 $\pm$ 2.06	83.20 $\pm$ 2.22	63.03 $\pm$ 3.39	92.57 $\pm$ 1.13	93.14$\pm$0.84
DB12	81.21 $\pm$ 1.66	75.19 $\pm$ 1.75	61.52 $\pm$ 1.91	86.97 $\pm$ 1.12	88.05$\pm$1.06
DB13	75.69 $\pm$ 1.82	71.13 $\pm$ 2.05	55.62 $\pm$ 1.83	81.86 $\pm$ 1.74	82.90$\pm$1.04
DB14	75.81 $\pm$ 1.84	70.65 $\pm$ 2.19	55.90 $\pm$ 2.70	81.93 $\pm$ 1.73	83.50$\pm$1.36
Ker					
DB1	42.05 $\pm$ 0.40	42.30 $\pm$ 0.81	41.91 $\pm$ 0.06	58.92 $\pm$ 2.89	59.13$\pm$2.94
DB2	50.13 $\pm$ 2.04	51.95 $\pm$ 4.03	48.80 $\pm$ 0.23	76.54 $\pm$ 3.49	76.80$\pm$3.01
DB3	44.44 $\pm$ 5.54	60.36 $\pm$ 2.83	39.16 $\pm$ 0.09	69.49$\pm$2.49	68.66 $\pm$ 2.18
DB4	53.28 $\pm$ 4.26	60.15 $\pm$ 2.27	44.54 $\pm$ 0.08	77.70$\pm$2.29	77.64 $\pm$ 2.03
DB5	79.15$\pm$1.08	79.11 $\pm$ 1.11	50.07 $\pm$ 0.39	78.03 $\pm$ 1.65	77.41 $\pm$ 1.43
DB6	90.21 $\pm$ 7.25	68.74 $\pm$ 19.15	44.75 $\pm$ 11.32	95.84 $\pm$ 1.03	97.07$\pm$0.67
DB7	92.08 $\pm$ 2.03	66.65 $\pm$ 18.66	46.04 $\pm$ 9.37	92.22 $\pm$ 1.41	92.50$\pm$1.29
DB8	87.54 $\pm$ 9.94	84.23 $\pm$ 17.81	48.93 $\pm$ 6.22	96.00 $\pm$ 1.00	96.06$\pm$0.75
DB9	63.96 $\pm$ 15.90	71.26 $\pm$ 13.57	30.94 $\pm$ 5.47	93.43 $\pm$ 1.02	93.94$\pm$0.63
DB10	54.96 $\pm$ 15.96	68.32 $\pm$ 17.08	33.10 $\pm$ 0.08	92.25 $\pm$ 1.15	93.36$\pm$0.85
DB11	52.11 $\pm$ 12.71	52.34 $\pm$ 13.39	33.11 $\pm$ 0.10	92.64 $\pm$ 1.12	93.11$\pm$0.78
DB12	27.06 $\pm$ 7.41	56.47 $\pm$ 11.22	20.04 $\pm$ 0.01	86.96 $\pm$ 1.15	88.00$\pm$1.03
DB13	30.23 $\pm$ 8.69	56.13 $\pm$ 10.85	18.85 $\pm$ 2.69	81.86 $\pm$ 1.59	82.96$\pm$0.85
DB14	25.76 $\pm$ 7.35	62.40 $\pm$ 8.37	19.46 $\pm$ 1.88	82.17 $\pm$ 1.77	83.56$\pm$1.18

Table 7.15: Classification accuracy in percent $\pm$ standard deviation for the methods of the fourth category obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. Most of the best results across all the feature sets are obtained by the SCCT.

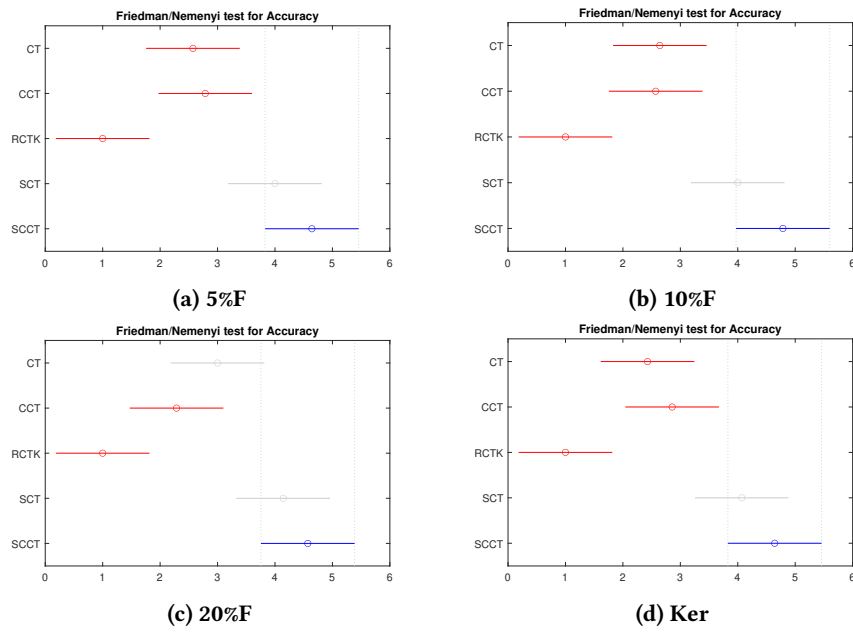


Figure 7.6: Mean ranks and 95% Nemenyi confidence intervals for the methods of the fourth category across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the SCCT for all the feature sets.

Method $\rightarrow$: Method $\downarrow$:	CT	CCT	RCTK	SCT	SCCT
CT	1.0000	0.1189	0.0001	0.0006	0.0004
CCT	0.6698	1.0000	0.0001	0.0004	0.0002
RCTK	0.0001	0.0001	1.0000	0.0001	0.0001
SCT	0.0001	0.0012	0.0001	1.0000	0.0785
SCCT	0.0001	0.0006	0.0001	0.0040	1.0000

Table 7.16: The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the fourth category, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

Method $\rightarrow$: Method $\downarrow$:	CT	CCT	RCTK	SCT	SCCT
CT	1.0000	0.0203	0.0001	0.0002	0.0002
CCT	0.1189	1.0000	0.0001	0.0006	0.0006
RCTK	0.0001	0.0001	1.0000	0.0001	0.0001
SCT	0.0004	0.0002	0.0001	1.0000	0.0107
SCCT	0.0004	0.0002	0.0001	0.0353	1.0000

Table 7.17: The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the fourth category, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

all the methods across all the feature sets except for the SCT in 5%F. Regarding the SCT, it performs better than the CT and the CCT for all the feature sets. Moreover, we can notice that the CT outperforms the CCT in 20%F.

Globally, the experiment shows that two methods stand out from the others: the SCCT and the SCT. These methods achieve to consistently outperform the other methods through all the feature sets on the investigated datasets. Therefore, it confirms that applying a sigmoid transformation to the CT and the CCT significantly increases their performance. This observation has already been made in node clustering experiment [299]. Moreover, the experiment highlights that the RCTK does not provide competitive results in our current setup.

Finally, the SCCT and the SCT are the two best-ranked methods in each feature set selected for the comparison in Subsection 7.4.3 according to the Nemenyi tests rankings presented in Figs. 7.6a and 7.6d.

Summary - Fourth category

- ▶ SCCT and SCT are the best performing methods
- ▶ RCTK is outclassed by all methods

7.4.2.5 Fifth category

We now examine the performance of the six similarity measures based on diffusion models of the fifth category introduced in Subsection 7.1.5. The performance of each of these methods across all the data and the feature sets are presented in Table 7.18.

First of all, we can observe from Table 7.18 that the LogCom stands out from the other methods appearing 43 times among the 56 highlighted results and providing 10 of the best results over 14 datasets across all the features sets. Moreover, 12 of the best results are obtained by directly feeding the kernel to the SVM. Across all the feature sets, we observe that the LogCom seems to outperform all the methods on DB1 to DB4 except for 5%F and 20%F where the RWR and the AMD respectively prevail on DB1. Regarding DB5, the Heat provides the best results through all the feature sets except for Ker where it is the RWR. For the Newsgroup datasets (DB6-DB14), the LogCom dominates across all the feature sets on DB6 and DB9 to DB14, except for DB11 where the Heat prevails in 10%F. The results on DB7 and DB8 are more contrasted: the Heat performs best for 20%F and Ker, whereas the RWR and the LogCom are the highlighted methods respectively for 5%F and 10%F.

Then, we investigate Friedman-Nemenyi tests [79] across all the datasets in each feature set. For the Friedman tests, we obtain p -values of 7.3×10^{-6} for 5%F, 4.2×10^{-5} for 10%F, 8.6×10^{-7} for 20%F, and 5.9×10^{-10} for Ker that are all lower than our threshold α of 0.05 indicating that at least one method is significantly different from the others. To identify these methods, we then carry out several Nemenyi tests and present the results in Fig. 7.7.

The Nemenyi tests indicate that the MD provides results inferior to those of the LogCom across all the feature sets. Moreover, the MD is outperformed by the ExpD and the Heat in 20%F and Ker, as well as the RWR in Ker. The tests also show that the LogCom perform better than the Heat in 5%F, the ExpD in 5%F and 10%F, the RWR in 10%F and 20%F, and the AMD across all the feature sets except for 5%F. Regarding the AMD, it has performance inferior to those of the Heat, the ExpD, and the RWR in Ker. Furthermore, we can point out that the RWR outperforms the ExpD and the Heat in 5%F.

To refine our analysis, we also perform multiple Wilcoxon signed-rank tests [79] and report the results in Tables 7.19 and 7.20. From these tables, we can observe that the LogCom outperforms all the methods across all the

Method →: Dataset ↓:	AMD	ExpD	Heat	LogCom	MD	RWWR
5%F						
DB1	56.89±2.63	57.55±3.82	43.97±2.19	58.74±2.98	57.75±3.42	59.90±2.98
DB2	74.51±2.55	72.14±2.91	56.06±3.34	78.71±2.28	74.45±3.95	74.88±2.28
DB3	66.92±2.33	68.54±2.11	63.02±2.78	70.12±2.34	66.00±2.73	66.81±2.34
DB4	72.61±2.70	73.99±2.49	60.07±3.88	78.88±1.71	74.51±2.90	76.26±1.71
DB5	78.34±1.07	76.96±1.54	79.05±1.12	77.12±1.57	77.73±1.45	78.00±1.57
DB6	94.31±1.53	94.70±1.87	95.40±1.36	95.43±1.29	94.64±1.82	93.84±1.29
DB7	91.94±1.58	91.48±2.47	91.31±1.42	91.46±2.02	91.21±1.76	91.97±2.02
DB8	94.24±1.11	93.95±1.98	94.58±0.88	95.04±1.80	94.50±1.97	95.39±1.80
DB9	89.64±1.74	87.58±5.76	90.41±1.39	92.82±1.19	90.28±1.75	90.80±1.19
DB10	90.24±1.36	88.91±3.07	89.41±1.72	91.50±1.29	89.74±1.66	90.33±1.29
DB11	90.31±1.42	88.63±2.18	89.66±1.35	90.75±1.40	89.74±1.76	90.43±1.40
DB12	85.13±1.26	84.22±2.15	78.55±1.47	86.83±1.10	83.63±2.08	84.85±1.10
DB13	79.61±2.06	78.20±1.88	75.47±1.92	82.54±1.19	79.16±2.83	81.13±1.19
DB14	79.99±1.50	79.55±1.52	74.24±1.97	81.83±1.63	79.01±2.34	81.09±1.63
10%F						
DB1	57.34±2.34	56.50±3.66	50.88±3.68	58.09±2.93	56.68±3.88	57.71±2.93
DB2	75.55±2.51	72.75±3.02	65.12±3.56	78.54±2.18	73.46±4.18	73.30±2.18
DB3	66.90±1.93	66.65±3.11	62.70±2.63	70.83±2.40	68.62±2.96	66.53±2.40
DB4	74.60±1.57	71.27±7.62	73.46±2.19	78.74±1.92	73.62±3.33	73.40±1.92
DB5	77.66±1.30	76.69±1.51	77.88±1.01	77.40±1.87	77.55±2.37	77.84±1.87
DB6	90.31±2.26	95.14±1.38	93.29±1.60	95.38±1.56	91.24±2.62	91.24±1.56
DB7	90.21±2.61	91.68±2.29	91.63±1.78	91.88±1.89	89.06±3.12	88.70±1.89
DB8	92.84±1.69	94.16±1.96	93.31±1.30	95.39±1.43	93.05±2.75	93.13±1.43
DB9	87.91±2.42	86.60±4.54	89.83±1.27	92.84±1.37	86.18±2.52	87.64±1.37
DB10	88.70±1.99	88.51±2.79	90.28±1.68	92.07±1.45	85.83±2.85	88.60±1.45
DB11	89.79±2.26	88.87±2.39	90.83±1.43	90.75±1.95	86.74±2.28	88.76±1.95
DB12	82.81±1.89	82.84±1.84	83.76±1.07	86.74±1.53	80.55±2.12	81.96±1.53
DB13	78.09±2.04	77.55±2.30	77.50±1.95	82.46±1.93	76.42±2.10	78.53±1.93
DB14	79.48±1.55	78.72±1.62	80.68±1.48	82.18±2.07	75.66±2.89	78.39±2.07
20%F						
DB1	59.23±3.21	55.67±3.90	47.32±2.72	58.02±3.06	53.60±4.99	55.23±3.06
DB2	77.40±2.14	73.44±2.97	64.75±2.16	78.28±2.87	69.78±4.30	70.31±2.87
DB3	68.00±2.05	66.30±3.22	43.05±3.15	70.48±2.32	68.89±2.13	65.43±2.32
DB4	74.48±1.66	71.32±3.19	69.80±1.82	78.72±1.91	71.29±3.32	70.08±1.91
DB5	77.80±1.42	76.40±1.80	78.42±1.23	77.28±1.59	76.33±2.12	78.29±1.59
DB6	85.28±2.50	95.23±1.53	94.04±1.90	95.63±1.36	82.99±2.94	88.21±1.36
DB7	85.26±2.16	91.83±1.78	92.85±1.61	92.26±1.49	80.82±3.61	85.82±1.49
DB8	88.11±2.41	93.95±1.96	95.19±1.20	95.09±1.61	84.51±4.22	90.65±1.61
DB9	81.58±1.78	84.60±5.08	92.11±1.26	92.95±1.10	78.55±2.91	84.45±1.10
DB10	83.24±2.02	88.88±3.31	90.93±1.41	92.14±1.10	79.85±3.89	84.64±1.10
DB11	84.39±2.13	88.78±2.93	86.67±1.80	91.24±1.41	81.17±3.20	84.37±1.41
DB12	79.18±1.53	82.24±1.87	80.38±2.20	87.22±1.26	74.03±2.16	77.35±1.26
DB13	73.02±1.77	75.46±1.84	76.42±2.40	82.78±1.58	70.65±2.14	72.66±1.58
DB14	73.62±1.85	76.90±2.23	78.62±2.22	82.39±1.58	69.14±3.09	72.66±1.58
Ker						
DB1	41.91±0.06	58.84±3.58	48.18±2.89	58.85±2.88	41.91±0.06	40.22±5.65
DB2	48.79±0.30	73.64±2.67	61.74±3.23	79.04±2.47	48.78±0.25	51.08±5.68
DB3	48.07±9.03	67.86±2.44	53.44±4.08	71.35±2.41	39.17±0.05	65.36±1.36
DB4	44.53±0.10	71.03±4.12	65.99±3.48	78.44±2.26	44.54±0.08	63.11±4.35
DB5	58.14±7.60	77.64±1.50	78.24±1.67	77.29±1.64	52.22±5.97	79.05±1.10
DB6	70.86±20.03	95.19±1.23	95.10±1.37	95.80±1.31	92.75±0.63	94.42±1.76
DB7	65.80±18.36	92.07±1.59	92.88±1.40	92.32±1.55	75.14±20.62	91.98±1.81
DB8	71.99±20.84	94.21±1.40	95.69±0.94	95.27±1.59	86.26±18.33	95.65±1.19
DB9	61.06±16.82	89.36±3.92	92.50±1.42	93.28±1.11	76.12±11.90	92.74±1.19
DB10	60.09±18.93	90.05±2.00	91.13±1.48	92.36±1.00	79.72±12.13	92.29±1.36
DB11	48.07±13.88	89.86±2.57	85.92±2.20	91.61±1.19	61.22±4.62	91.26±1.73
DB12	41.22±12.12	84.29±2.04	77.00±3.11	87.77±1.46	61.10±9.36	87.48±1.26
DB13	42.09±10.32	80.21±1.59	71.96±4.17	83.16±1.51	57.01±5.7	82.77±1.43
DB14	43.50±10.77	81.08±1.92	75.22±2.50	82.73±1.79	59.78±2.91	82.18±1.52

Table 7.18: Classification accuracy in percent $\pm$ standard deviation for the methods of the fifth category obtained on the different datasets. Results are reported for the four feature sets (5%, 10%, 20%, and Ker). The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. The LogCom stands out from the other methods by providing most of the highlighted results.

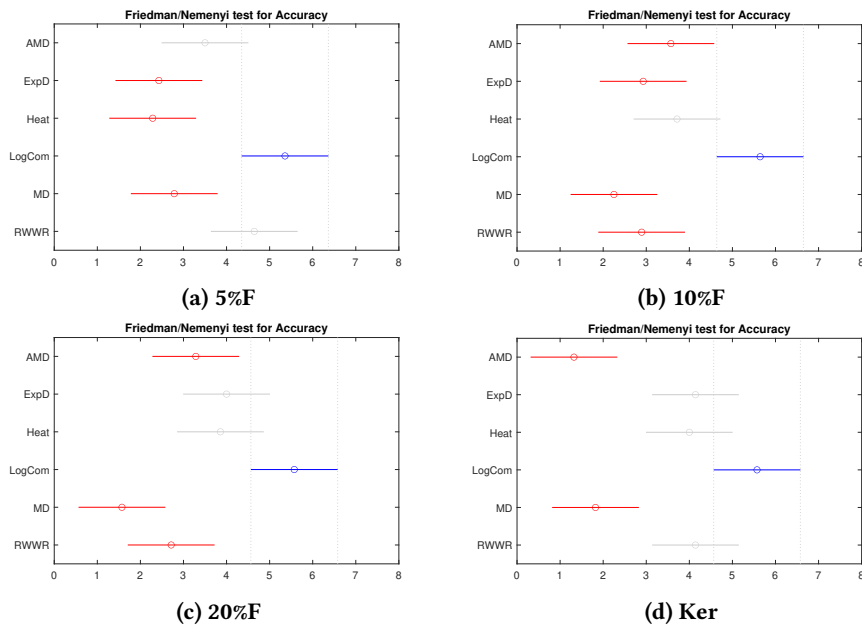


Figure 7.7: Mean ranks and 95% Nemenyi confidence intervals for the methods of the fifth category across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the LogCom for all the feature sets.

Method→: Method↓:	AMD	ExpD	Heat	LogCom	MD	RWWR
AMD	1.0000	0.1040	0.0245	0.0023	0.3575	0.0580
ExpD	0.2958	1.0000	0.1353	0.0002	0.1531	0.0067
Heat	0.8077	1.0000	1.0000	0.0012	0.0494	0.0107
LogCom	0.0002	0.0001	0.0009	1.0000	0.0006	0.0203
MD	0.0134	0.0906	0.4631	0.0002	1.0000	0.0023
RWWR	0.1189	0.9032	0.5830	0.0004	0.0803	1.0000

Table 7.19: The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the fifth category, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

Method→: Method↓:	AMD	ExpD	Heat	LogCom	MD	RWWR
AMD	1.0000	0.1040	0.6257	0.0009	0.0002	0.5416
ExpD	0.0001	1.0000	0.5016	0.0001	0.0009	0.0012
Heat	0.0001	0.0419	1.0000	0.0023	0.0785	0.3258
LogCom	0.0001	0.0006	0.0031	1.0000	0.0001	0.0002
MD	0.0081	0.0001	0.0001	0.0001	1.0000	0.0052
RWWR	0.0002	0.9032	0.3910	0.0166	0.0004	1.0000

Table 7.20: The p -values provided by pairwise Wilcoxon signed-rank tests on the methods of the fifth category, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

features sets. Moreover, we can notice that the RWWR performs better than the MD for all the feature sets except for 10%F. Regarding the MD, it has performance superior to those of the Heat in 5%F, as well as those of the AMD in Ker. Moreover, the AMD outperforms the MD in 10%F and in 20%F and the Heat in 5%F. Finally, the tests indicate that the ExpD performs better than the Heat in Ker and the RWWR in 20%F.

In summary, we can state that the LogCom is the method providing the best performance in our experiment across all the data and the feature sets. We can also point out that the RWWR, the Heat, and the ExpD obtain competitive results on the investigated dataset, but these results are not consistent across all the feature sets. Overall, the MD and the AMD perform worse than the other methods.

Finally, following the Nemenyi tests rankings presented in Figs. 7.7a to 7.7d, the two best-ranked methods in each feature set selected for the comparison in Subsection 7.4.3 are: the LogCom and the RWWR in 5%F and in Ker, the LogCom and the Heat in 10%F, and the LogCom and the ExpD in 20%F.

Feature set →: Category ↓:	5%F	10%F	20%F	Ker
First category	PWSurp	cBoPH	PWSurp	PWSurp
First category	FETsa	RSPTsa	FE	cBoPH
Second category	nCor	nCor	nCor	nCor
Second category	nCorH	nCorH	nCorH	nCorH
Third category	LF	LF	LF	LF
Third category	MRL	LPI	LPI	MRL
Fourth category	SCT	SCT	SCT	SCT
Fourth category	SCCT	SCCT	SCCT	SCCT
Fifth category	LogCom	LogCom	LogCom	LogCom
Fifth category	RWWR	Heat	ExpD	RWWR

Table 7.21: The best-ranked methods selected according to the Nemenyi tests rankings in each category and in each feature set.

Summary - Fifth category

- LogCom is the method providing the best performance
- AMD and MD obtain lower results

7.4.3 Best-ranked methods analysis

In this subsection, we analyze the results of the best-ranked methods selected across the category analysis for each feature set. The selected methods are repeated in Table 7.21 and their results are summarized in Table 7.22.

From the raw results in Table 7.22, it can be seen that 11 of the best results out of the 14 datasets are obtained by directly feeding the kernel to the SVM. Across all the feature sets, we observe that the LF seems to outperform all the methods on DB3 to DB4 except for 5%F where the PWSurp prevails on DB4. Regarding DB5, the nCor provides the best results through all the feature sets except for 20%F where it is the nCorH. We can also point out that the SCCT dominates on DB6, DB9 to DB11, and DB14 across all the feature sets, except for Ker on DB11 and DB14 where the PWSurp prevails. Moreover, the PWSurp obtains the best results through all the feature sets on DB13. For the five remaining datasets, the best method is dataset-dependent and feature set-dependent.

Then, we compare the different methods across all the 14 datasets through Friedman tests followed by Nemenyi post-hoc tests [79] to evaluate the overall results of each method. The p -values of the Friedman tests are respectively 9.8×10^{-9} for 5%F, 2.9×10^{-10} for 10%F, 1.3×10^{-11} for 20%F, and 5.7×10^{-9} for Ker. All these p -values are lower than the threshold α of 0.05, meaning

Method →: Dataset ↓:	PWSSurp	FETsa	nCor	nCorH	LF	MRL	SCT	SCCT	LogCom	RWWR
5%F										
DB1	60.06±2.79	59.04±3.04	59.30±2.96	58.93±3.79	59.99±2.32	59.54±3.25	59.18±2.63	59.96±2.61	58.74±2.98	59.90±2.98
DB2	78.50±2.22	77.47±2.88	76.29±2.97	75.85±3.13	78.35±2.18	75.44±2.58	76.72±3.34	76.56±3.05	78.71±2.28	74.88±2.28
DB3	70.89±2.90	69.05±2.84	68.37±2.74	68.42±2.64	71.22±1.87	69.28±2.11	69.24±2.53	68.13±2.58	70.12±2.34	66.81±2.34
DB4	79.09±1.76	77.69±1.93	77.44±1.93	77.87±1.78	78.97±2.25	75.24±1.83	77.21±2.10	77.78±2.06	78.88±1.71	76.26±1.71
DB5	78.01±1.35	78.23±1.60	79.04±1.03	78.73±1.20	78.50±1.82	78.06±1.68	78.27±1.22	77.45±1.35	77.12±1.57	78.00±1.57
DB6	95.90±1.29	95.34±1.39	95.04±1.54	94.59±1.45	94.79±2.16	94.72±2.24	96.14±1.37	97.11±0.59	95.43±1.29	93.84±1.29
DB7	92.54±1.54	93.47±1.58	93.39±1.52	93.09±1.86	92.84±1.55	92.29±1.74	92.30±1.68	93.00±1.10	91.46±2.02	91.97±2.02
DB8	96.40±0.86	96.50±1.11	95.73±1.19	95.82±1.23	95.37±1.38	94.79±1.27	95.81±1.32	95.86±1.04	95.04±1.80	95.39±1.80
DB9	93.01±1.33	92.90±1.71	92.09±1.40	92.14±1.37	92.65±1.31	91.00±2.13	93.34±1.27	93.90±0.87	92.82±1.19	90.80±1.19
DB10	92.61±1.43	92.79±1.27	92.07±1.33	92.10±1.31	91.51±1.57	90.73±1.63	91.48±1.83	93.24±1.11	91.50±1.29	90.33±1.29
DB11	92.74±1.60	92.78±1.58	91.22±1.40	91.04±1.37	91.74±1.96	89.84±1.83	92.53±1.30	92.93±0.92	90.75±1.40	90.43±1.40
DB12	88.80±1.20	88.93±1.03	85.51±1.62	85.63±1.52	86.58±1.57	84.22±1.53	86.96±1.18	87.84±0.92	86.83±1.10	84.85±1.10
DB13	83.12±1.53	83.36±1.75	81.63±1.90	81.32±1.68	82.07±2.08	80.45±1.78	81.74±1.63	82.74±1.13	82.54±1.19	81.13±1.19
DB14	82.55±1.58	82.83±1.50	81.68±1.51	81.83±1.36	82.65±1.31	80.99±1.40	82.35±1.53	83.15±1.41	81.83±1.63	81.09±1.63
Method →: Dataset ↓:	cBoPH	RSPTsa	nCor	nCorH	LF	LPI	SCT	SCCT	LogCom	Heat
10%F										
DB1	57.98±3.19	58.11±3.71	58.04±3.51	58.54±2.92	58.44±3.26	58.71±3.40	58.52±3.16	59.48±2.89	58.09±2.93	50.88±3.68
DB2	76.82±3.03	75.61±3.27	75.14±2.87	75.35±3.21	78.44±2.31	73.64±3.02	76.41±3.32	76.60±2.88	78.54±2.18	65.12±3.56
DB3	68.29±2.46	68.76±3.27	67.28±2.75	67.67±2.74	71.14±2.26	69.26±2.53	68.89±2.58	68.68±2.16	70.83±2.40	62.70±2.63
DB4	77.67±2.17	77.04±2.20	76.70±1.77	76.77±1.86	79.14±2.06	75.94±1.84	77.61±2.22	77.83±2.12	78.74±1.92	73.46±2.19
DB5	78.95±1.42	78.93±1.58	79.15±1.31	79.15±1.43	79.11±1.38	77.43±1.65	78.01±1.42	77.38±1.26	77.40±1.87	77.88±1.01
DB6	95.08±1.69	95.06±1.59	93.93±1.72	93.85±1.70	93.94±2.16	93.38±1.79	95.83±1.26	97.00±0.80	95.38±1.56	93.29±1.60
DB7	92.36±1.69	91.98±2.36	91.93±2.63	91.92±2.15	91.43±2.13	89.77±2.18	92.38±1.44	92.87±1.20	91.88±1.89	91.63±1.78
DB8	96.17±1.13	96.41±1.05	95.36±1.39	95.54±1.24	95.36±1.22	94.78±1.48	95.91±0.93	96.07±0.73	95.39±1.43	93.31±1.30
DB9	93.02±1.66	93.09±1.27	91.95±1.44	91.60±1.50	91.96±1.70	89.80±2.02	93.18±1.12	93.84±0.74	92.84±1.37	89.83±1.27
DB10	92.72±1.39	93.09±1.49	91.63±2.06	91.09±1.95	91.93±1.37	89.36±1.85	91.95±1.10	93.31±0.96	92.07±1.45	90.28±1.68
DB11	92.97±1.47	92.47±1.41	90.80±1.46	90.26±1.65	90.96±2.05	89.54±1.98	92.37±1.36	93.09±0.89	90.75±1.95	90.83±1.43
DB12	88.79±1.26	89.06±1.23	84.58±1.60	84.68±1.84	86.50±1.65	84.23±1.58	86.61±1.46	88.17±0.97	86.74±1.53	83.76±1.07
DB13	83.56±1.52	83.02±2.14	81.00±1.76	80.5±1.56	81.94±2.06	78.73±1.66	81.93±1.69	82.84±1.28	82.46±1.93	77.50±1.95
DB14	82.79±1.66	83.26±1.51	81.81±1.86	80.59±1.77	82.29±1.72	78.13±2.40	82.18±1.66	83.39±1.27	82.18±2.07	80.68±1.48
Method →: Dataset ↓:	PWSSurp	FE	nCor	nCorH	LF	LPI	SCT	SCCT	LogCom	ExpD
20%F										
DB1	57.44±3.41	57.81±3.64	58.06±3.18	57.87±3.06	58.50±2.92	58.38±4.23	58.35±3.48	59.30±2.95	58.02±3.06	55.67±3.90
DB2	77.26±2.84	76.72±3.60	75.22±3.42	75.60±2.74	78.53±2.65	73.62±2.85	76.22±3.21	76.86±3.02	78.28±2.87	73.44±2.97
DB3	70.49±2.23	69.36±2.04	67.91±2.06	68.59±2.41	71.09±2.39	69.15±2.27	68.92±2.61	68.59±2.09	70.48±2.32	66.30±3.22
DB4	78.36±2.29	78.09±2.69	76.13±1.66	76.06±1.77	79.43±2.03	76.06±1.72	77.85±2.44	77.49±2.14	78.72±1.91	71.32±3.19
DB5	78.05±1.58	79.06±1.51	79.53±1.22	79.55±1.10	79.09±1.34	77.58±1.53	78.18±1.52	77.53±1.22	77.28±1.59	76.40±1.80
DB6	95.28±1.31	95.00±1.53	92.86±1.72	92.37±1.98	94.17±1.97	94.02±1.51	95.75±1.31	96.94±0.73	95.63±1.36	95.23±1.53
DB7	92.31±2.05	92.01±1.41	91.19±2.11	90.81±2.28	91.66±1.60	90.28±1.88	91.85±1.62	92.79±1.15	92.26±1.49	91.83±1.78
DB8	96.24±1.07	96.30±0.98	94.67±1.27	94.86±1.25	95.28±1.61	94.97±1.42	96.09±1.00	96.07±0.83	95.09±1.61	93.95±1.96
DB9	93.18±1.29	93.01±1.40	90.50±1.77	90.23±1.80	92.08±1.51	90.83±1.73	93.39±1.03	93.90±0.62	92.95±1.10	84.60±5.08
DB10	92.75±1.31	92.69±1.32	90.19±1.80	90.05±1.75	91.97±1.55	90.24±1.74	92.14±1.02	93.31±0.86	92.14±1.10	88.88±3.31
DB11	92.98±1.41	92.80±1.10	89.41±1.90	89.26±2.10	91.72±1.68	90.53±1.69	92.57±1.13	93.14±0.84	91.24±1.41	88.78±2.93
DB12	89.02±0.96	88.75±0.98	83.86±1.57	83.74±1.47	86.85±1.48	85.31±1.63	86.97±1.12	88.05±1.06	87.22±1.26	82.24±1.87
DB13	83.32±1.65	83.25±1.45	78.60±2.13	77.84±2.21	82.39±1.74	79.81±1.63	81.86±1.74	82.90±1.04	82.78±1.58	75.46±1.84
DB14	82.84±1.72	82.94±1.61	78.97±1.85	78.26±1.64	82.22±1.78	79.37±2.17	81.93±1.73	83.50±1.36	82.39±1.58	76.90±2.23
Method →: Dataset ↓:	PWSSurp	cBoPH	nCor	nCorH	LF	MRL	SCT	SCCT	LogCom	RWWR
Ker										
DB1	59.61±2.59	58.79±3.38	52.89±3.63	53.58±3.51	59.96±2.65	41.91±0.09	58.92±2.89	59.13±2.94	58.85±2.88	40.22±5.65
DB2	78.55±3.13	76.68±3.26	67.36±2.47	68.01±2.10	79.46±2.72	49.30±1.95	76.54±3.49	76.80±3.01	79.04±2.47	51.08±5.68
DB3	71.85±2.04	70.12±2.19	65.84±0.97	65.71±1.23	71.87±1.99	64.95±1.39	69.49±2.49	68.66±2.18	71.35±2.41	65.36±1.36
DB4	77.95±2.14	76.42±2.42	73.74±1.85	73.70±1.78	79.48±1.95	50.12±0.91	77.70±2.29	77.64±2.03	78.44±2.26	63.11±4.35
DB5	77.95±1.33	79.24±1.48	79.63±1.16	79.60±1.10	79.19±1.36	77.82±2.42	78.03±1.65	77.41±1.43	77.29±1.64	79.05±1.10
DB6	96.13±1.12	95.63±0.88	95.51±1.09	95.32±1.18	94.95±1.72	94.46±1.64	95.84±1.03	97.07±0.67	95.80±1.31	94.42±1.76
DB7	92.41±1.78	92.59±1.60	92.47±1.57	92.56±1.72	92.27±1.54	91.70±1.70	92.22±1.41	92.50±1.29	92.32±1.55	91.98±1.81
DB8	96.44±0.99	96.62±0.74	95.93±1.28	96.05±1.23	95.53±1.56	95.02±1.26	96.00±1.00	96.06±0.75	95.27±1.59	95.65±1.19
DB9	93.56±1.06	93.33±1.00	92.98±1.43	92.84±1.26	92.70±1.13	91.91±1.51	93.43±1.02	93.94±0.63	93.28±1.11	92.74±1.19
DB10	93.23±1.05	93.23±1.00	92.39±1.06	92.28±1.14	92.35±1.33	91.52±1.50	92.25±1.15	93.36±0.85	92.36±1.00	92.29±1.36
DB11	93.28±1.32	93.26±1.02	92.43±1.35	92.45±1.27	92.36±1.43	91.26±1.56	92.64±1.12	93.11±0.78	91.61±1.19	91.26±1.73
DB12	89.25±1.07	88.94±0.89	87.87±1.49	87.93±1.54	87.95±1.17	86.26±1.67	86.96±1.15	88.00±1.03	87.77±1.46	87.48±1.26
DB13	84.46±1.52	83.94±1.32	82.78±1.42	82.76±1.42	82.69±1.84	80.94±2.03	81.86±1.59	82.96±0.85	83.16±1.51	82.77±1.43
DB14	84.15±1.51	83.54±1.32	82.36±1.28	82.49±1.30	82.50±1.72	80.94±1.71	82.17±1.77	83.56±1.18	82.73±1.79	82.18±1.52

Table 7.22: Classification accuracy in percent $\pm$ standard deviation for the two best-ranked methods of each category in each feature set (5%, 10%, 20%, and Ker) obtained on the different datasets. The best performing method is highlighted in boldface for each data and feature set. Bold values highlighted in grey indicate the best performance overall for each dataset, across all the feature sets. The best results across all the feature sets are obtained either in Ker or in 5%F, but the methods providing these results differ from one dataset to another.

that the performance of at least one classifier is significantly different from the others. As all the Friedman tests are positive, we can perform Nemenyi tests which determine whether or not the outcomes of each method differs significantly from each other. The results of these tests are reported in Fig. 7.8.

First of all, we observe from Fig. 7.8a that the PWSurp, the FETsa, the LF, and the SCCT outperform the MRL and the RWWR in 5%F. Moreover, the SCT performs better than the RWWR in 5%F. From Fig. 7.8b, we can see that the SCCT obtains results significantly superior to those of the nCor, the nCorH, the LPI and the Heat in 10%F. We can also notice that the cBoPH, the RSPTsa, the LF, and the SCT perform better than the LPI and the Heat in 10%F. In addition, the Heat is outperformed by the LogCom in 10%F. As can be seen in Fig. 7.8c, the PWSurp, the SCCT, the FE, the LF, the SCT, and the LogCom provide results significantly superior to those of the ExpD in 20%F. Furthermore, the PWSurp, the SCCT, the FE and the LF perform better than the nCorH in 20%F. The tests also highlight that the PWSurp, the SCCT, and the FE outperform the nCor in 20%F. Regarding the PWSurp and the SCCT, they outperform as well the LPI in 20%F. From Fig. 7.8d, we can point out that the MRL obtains results significantly inferior to those of all the methods except the RWWR in Ker. On its side, the RWWR is outperformed by the PWSurp, the cBoPH, and the SCCT in Ker.

We continue our analysis by performing multiple Wilcoxon signed-ranks tests for matched data [79] to potentially discover other significant pairwise differences between the methods. The results of these tests are presented in Table 7.23 for 5%F and 10%F and in Table 7.24 for 20%F and Ker.

Besides confirming the findings of the Friedman-Nemenyi tests, the Wilcoxon tests indicate that the RWWR and the MRL are outperformed by all the methods except each other in 5%F. We can also notice that the nCor and the nCorH obtain results significantly inferior to those of the SCCT, the LF, the FETsa, and the PWSurp in 5%F. With regards to the FETsa and the PWSurp, they perform both better than the SCT in 5%F. Moreover, the PWSurp outperforms the LF and the LogCom in 5%F. In 10%F, the tests highlight that the Heat and the LPI provide outcomes inferior to those of all the methods except each other. We can also point out that the cBoPH, the RSPTsa, the SCT, the LF, and the LogCom outperform the nCor and the nCorH in 10%F. In addition, the SCCT performs better than the SCT in 10%F. Regarding 20%F, the tests show that the ExpD is outperformed by all the methods. We can also notice that the FE, the LF, the LogCom, and the SCT provide results superior to those of the LPI, the nCor, and the nCorH in 20%F. On its side, the SCT performs worse than the PWSurp and the SCCT in 20%F. In Ker, we can see that the PWSurp outperforms all the methods except the LF. The tests also highlight that the MRL obtains results significantly inferior to those of all the methods in Ker. Furthermore, the tests indicate that the RWWR is outperformed by all

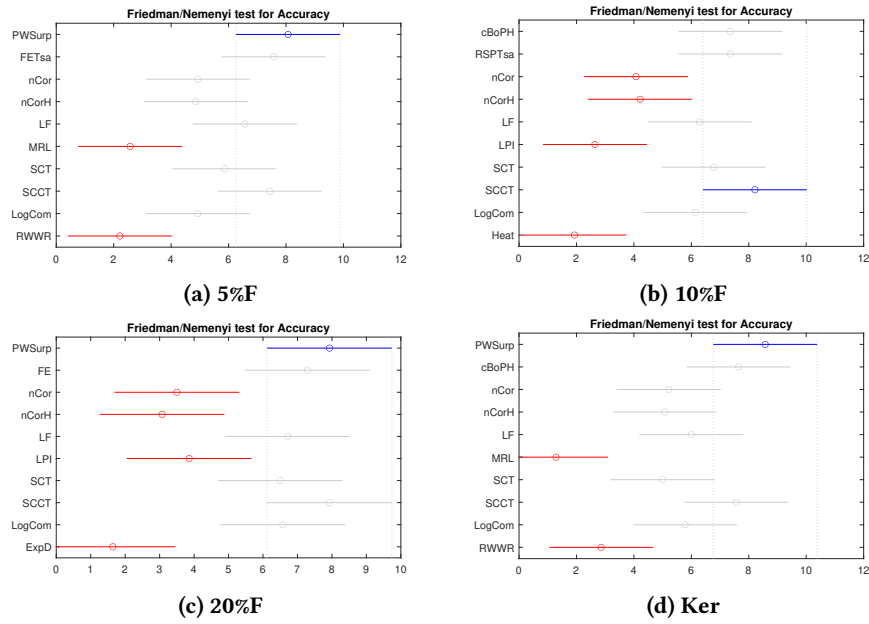


Figure 7.8: Mean ranks and 95% Nemenyi confidence intervals for the best-ranked methods across the 14 datasets for feature sets 5%F (a), 10%F (b), 20%F (c), and Ker (d). Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best methods highlighted in blue are the PWSurp for all the feature sets, except for 10%F where it is the SCCT.

Method 5%F $\rightarrow$: Method 10%F $\downarrow$:	PWSurp	FETsa	nCor	nCorH	LF	MRL	SCT	SCCT	LogCom	RWWR
cBoPH	1.0000	0.5830	0.0085	0.0017	0.0494	0.0002	0.0134	0.5016	0.0006	0.0001
RSPtsa	0.6257	1.0000	0.0067	0.0017	0.4263	0.0012	0.0494	0.7148	0.0906	0.0004
nCor	0.0006	0.0006	1.0000	0.2412	0.0494	0.0017	0.1937	0.0245	0.5830	0.0009
nCorH	0.0009	0.0023	0.5016	1.0000	0.0166	0.0067	0.1040	0.0245	0.3910	0.0031
LF	0.3575	0.4631	0.0107	0.0134	1.0000	0.0001	0.4263	0.4631	0.1353	0.0002
LPI	0.0006	0.0006	0.0107	0.0031	0.0002	1.0000	0.0017	0.0031	0.0295	0.7609
SCT	0.1189	0.3575	0.0031	0.0031	1.0000	0.0006	1.0000	0.0785	0.6698	0.0012
SCCT	0.3575	0.2166	0.0040	0.0023	0.3575	0.0006	0.0040	1.0000	0.1937	0.0006
LogCom	0.3575	0.4631	0.0245	0.0166	0.7148	0.0009	1.0000	0.1726	1.0000	0.0203
Heat	0.0001	0.0001	0.0002	0.0012	0.0004	0.3258	0.0001	0.0002	0.0009	1.0000

Table 7.23: The p -values provided by pairwise Wilcoxon signed-rank tests on the best-ranked methods, for 5%F in the upper right triangle and 10%F in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

Method 20%F $\rightarrow$: Method Ker $\downarrow$:	PWSurp	FE	nCor	nCorH	LF	LPI	SCT	SCCT	LogCom	ExpD
PWSurp	1.0000	0.1531	0.0009	0.0009	0.3910	0.0004	0.0353	0.9032	0.1353	0.0001
cBoPH	0.0295	1.0000	0.0006	0.0006	0.4631	0.0004	0.0676	0.5416	0.5416	0.0004
nCor	0.0040	0.0009	1.0000	0.1353	0.0002	0.3258	0.0012	0.0023	0.0023	0.0067
nCorH	0.0040	0.0006	0.8077	1.0000	0.0004	0.2412	0.0017	0.0031	0.0023	0.0134
LF	0.2166	0.8552	0.7609	0.8077	1.0000	0.0001	0.6698	0.6257	0.8077	0.0006
MRL	0.0001	0.0001	0.0001	0.0001	0.0001	1.0000	0.0006	0.0006	0.0012	0.0052
SCT	0.0002	0.0295	0.3575	0.4263	0.0580	0.0001	1.0000	0.0107	0.6698	0.0001
SCCT	0.0353	0.6698	0.0052	0.0085	0.9515	0.0002	0.0353	1.0000	0.2676	0.0001
LogCom	0.0040	0.5016	0.2412	0.2166	0.2676	0.0006	0.2958	0.3910	1.0000	0.0001
RWWR	0.0012	0.0001	0.0001	0.0006	0.0031	0.0107	0.0676	0.0031	0.0166	1.0000

Table 7.24: The p -values provided by pairwise Wilcoxon signed-rank tests on the best-ranked methods, for 20%F in the upper right triangle and Ker in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

the methods except the SCT and the MRL in Ker. Finally, we observe that the cBoPH and the SCCT perform better than the nCor, the nCorH, and the SCT in Ker.

Overall, the experiment shows that the RWWR, the MRL, the Heat, the LPI, and the ExpD are outclassed by most of the other methods in their respective feature set. We can also point out that the nCor and the nCorH are outperformed by the BoP methods of the first category and the SCCT across all the feature sets. Moreover, the nCor and the nCorH obtain results significantly inferior to those of the LF through all the feature sets except for Ker, as well as those of the LogCom and the SCT in 10%F and 20%F. Regarding the SCT, it is outperformed by the SCCT across all the feature sets except for 5%F, the PWSurp through all the feature sets except for 20%F where the PWSurp was not selected, the FETsa in 5%F and the cBoPH in Ker. We can also notice that the PWSurp is the only method that manages to distinguish itself from the remaining methods by obtaining results superior to those of the LogCom and the LF in 5%F and those of all the other methods except the LF in Ker.

7.4.4 Comparison of the impact of the different extracted feature sets

We now analyze the impact of the feature extraction technique (with a growing number of extracted features) on the classification results, limited to the 16 best-ranked methods sectioned for the comparison in the previous subsection for conciseness.

To analyze the results of the 16 methods across all the feature sets (5%, 10%, 20% and Ker), we follow the same procedure as before. First, we perform Friedman tests [79] to identify if there are some differences between the results across all the feature sets for each method. All the p -values of these tests are not lower than our threshold α of 0.05⁵, however we pursue our analysis by performing Nemenyi tests [79] to at least obtain a ranking and report the results in Figs. 7.9 and 7.10.

From these figures, we can observe that the Ker outperforms the 10%F feature sets for the cBoPH, the FE, the FETsa, the LPI, the LF, and the ExpD. We can also notice that the results obtained by the Ker are significantly superior to those of the 20%F for the cBoPH, the PWSurp, the MRL, nCor, the nCorH, the ExpD, and the RWWr. Moreover, the Ker performs better than the 5%F for the LPI, the LF, and the LogCom. With regards to the 5%F, it outperforms the 20%F for the MRL, the nCor, the nCorH, and the RWWr. The tests also highlight that the 5%F provides results superior to those of the 10%F for the MRL.

To refine our analysis, we also perform multiple Wilcoxon signed-ranks tests performing pairwise comparisons [79] and report the results on Table 7.25. In contrast to our previous analyses, the Wilcoxon tests do not confirm all the findings of the Friedman-Nemenyi tests. The Wilcoxon tests do not support that the Ker outperforms the 20%F for the nCor, the nCorH, the MRL, and the RWWr. In addition, the Wilcoxon tests do not confirm that the Ker performs better than the 10%F for the FETsa.

However, the Wilcoxon tests indicate that the Ker outperforms all the feature sets for the LPI, the ExpD, and the LogCom. We can also notice that the Ker provides results significantly superior to those of the 10%F for the PWSurp and the SCT. Furthermore, the Ker performs better than the 20%F for the FE. On its side, the 20%F achieves results inferior to those of the 5%F for the ExpD. The tests also highlight that the 5%F outperforms the 10%F for the nCor, the nCorH, the ExpD, and the RWWr. Regarding the 10%F, it performs better than the 20%F for the nCor, the nCorH, the MRL, and the RWWr. Moreover, the 10%F obtains results superior to those of the Ker for Heat. We can also point out that the 20%F outperforms the 10%F for the LPI and the LF, as well

⁵The p -values of the Friedman tests are respectively 0.4180 for the RSPTsa, 0.2273 for the SCT, 0.5274 for the SCCT and 0.4936 for the Heat. All the other p -values are inferior to 0.05.

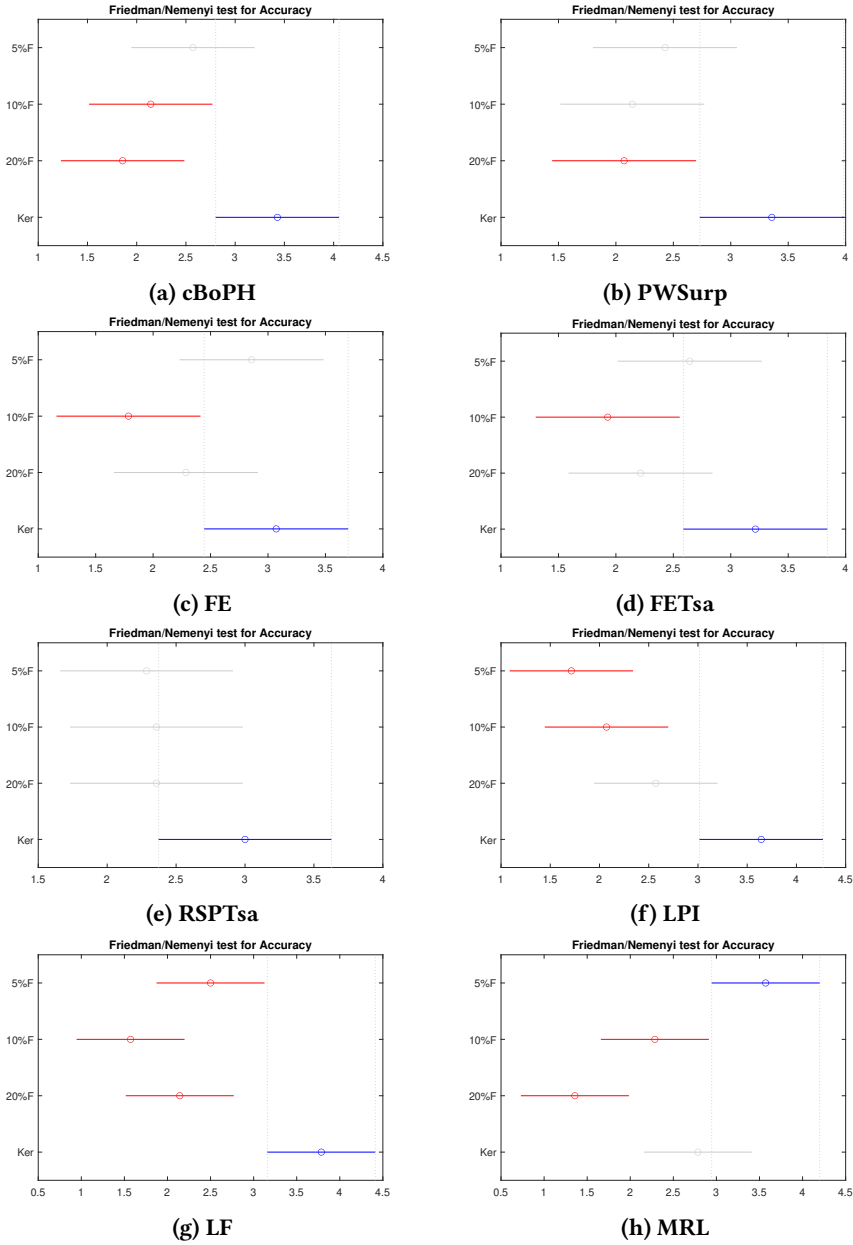


Figure 7.9: Mean ranks and 95% Nemenyi confidence intervals for the eight best-ranked method of the first and third categories across the four feature sets (5%F, 10%F, 20%F, and Ker). Two feature sets are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the feature sets: the higher the rank, the best the results obtained on the feature set. Each time, the best feature set highlighted in blue is either the Ker or the 5%F.

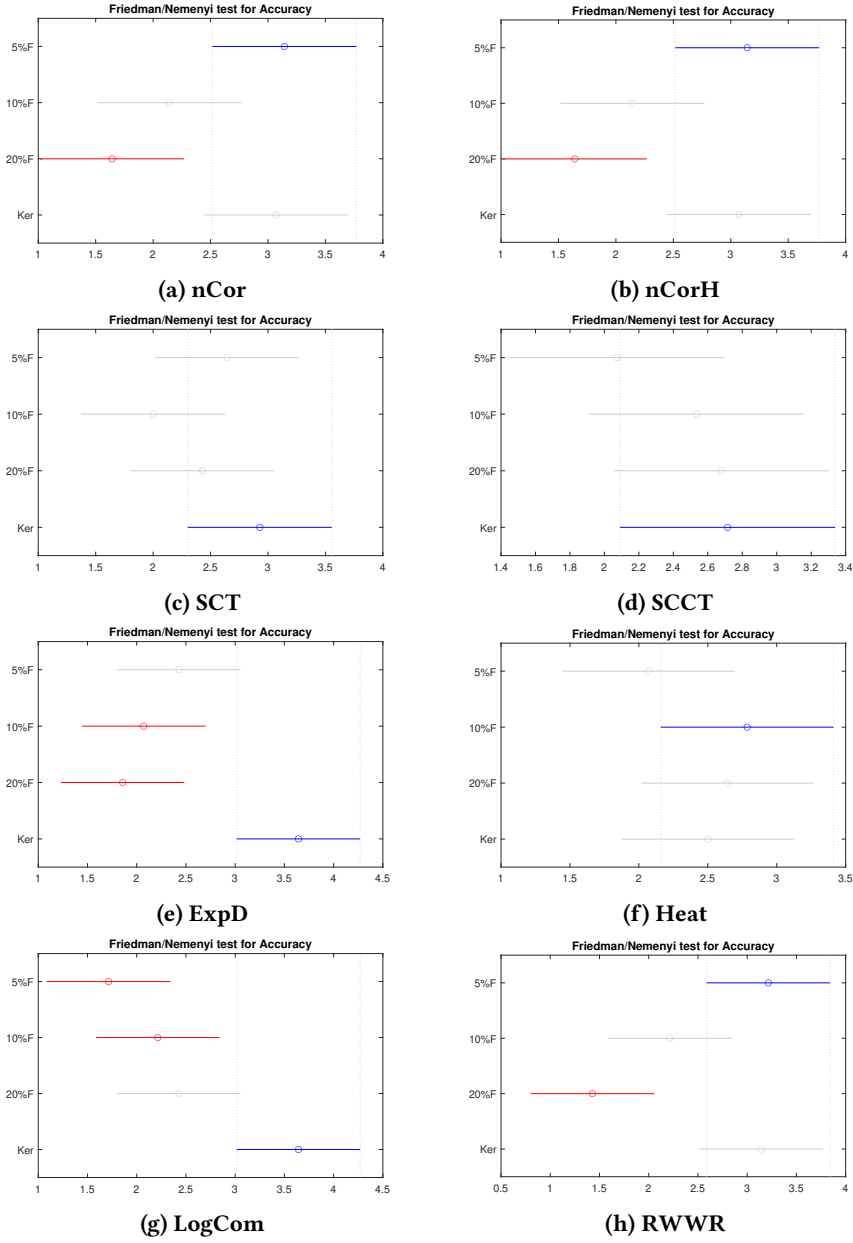


Figure 7.10: Mean ranks and 95% Nemenyi confidence intervals for the eight best-ranked method of the second, fourth and fifth categories across the four feature sets (5%F, 10%F, 20%F, and Ker). Two feature sets are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the feature sets: the higher the rank, the best the results obtained on the feature set. Each time, the best feature set highlighted in blue is either the Ker or the 5%, except for the SCCT where it is 10%F.

Feature set $\rightarrow$: Feature set $\downarrow$:	5%F	10%F	20%F	Ker
5%F	1.0000	0.1531	0.1531	0.3910
10%F	0.1040	1.0000	0.4631	0.0134
20%F	0.2958	0.8552	1.0000	0.0040
Ker	0.0906	0.0107	0.0040	1.0000

(a) cBoPH & PWSurp

Feature set $\rightarrow$: Feature set $\downarrow$:	5%F	10%F	20%F	Ker
5%F	1.0000	0.9032	1.0000	0.7148
10%F	0.0006	1.0000	0.5016	0.3258
20%F	0.0004	0.0067	1.0000	0.3575
Ker	0.8077	0.6698	0.3575	1.0000

(b) FE & FETSa

Feature set $\rightarrow$: Feature set $\downarrow$:	5%F	10%F	20%F	Ker
5%F	1.0000	0.0006	0.0009	1.0000
10%F	0.1937	1.0000	0.0040	0.7148
20%F	0.0134	0.0052	1.0000	0.3910
Ker	0.0023	0.0012	0.0006	1.0000

(c) RSPTsa & nCor

Feature set $\rightarrow$: Feature set $\downarrow$:	5%F	10%F	20%F	Ker
5%F	1.0000	0.1189	0.5830	0.0067
10%F	0.0001	1.0000	0.0166	0.0001
20%F	0.0001	0.0002	1.0000	0.0001
Ker	0.5016	0.8552	0.5016	1.0000

(d) nCorH & LPI

Feature set $\rightarrow$: Feature set $\downarrow$:	5%F	10%F	20%F	Ker
5%F	1.0000	0.2676	0.5416	0.8077
10%F	0.3575	1.0000	0.5830	0.0107
20%F	0.3575	0.7480	1.0000	0.0906
Ker	0.3258	0.9515	0.9032	1.0000

(e) LF & MRL

Feature set $\rightarrow$: Feature set $\downarrow$:	5%F	10%F	20%F	Ker
5%F	1.0000	0.0353	0.0419	0.0419
10%F	0.1040	1.0000	0.1040	0.0006
20%F	0.1531	0.1937	1.0000	0.0017
Ker	0.5016	0.0494	0.5416	1.0000

(f) SCT & SCCT

Feature set $\rightarrow$: Feature set $\downarrow$:	5%F	10%F	20%F	Ker
5%F	1.0000	0.3910	0.1040	0.0017
10%F	0.0001	1.0000	0.3258	0.0023
20%F	0.0002	0.0002	1.0000	0.0017
Ker	0.7148	0.4631	0.2958	1.0000

(g) ExpD & Heat

(h) LogCom & RWWR

Table 7.25: The p -values provided by pairwise Wilcoxon signed-rank tests for the 16 methods. Each table contains the p -values of two methods, for the first method in the upper right triangle and the second method in the lower left. All the p -values lower than our threshold α of 0.05 are highlighted in boldface.

as the 5%F for the LPI.

To conclude, the experiment highlights that the results obtained by extracting 5% of dominant eigenvectors are not significantly different from the results obtained by using all the information contained in the kernel for all the methods except for the LPI, the ExpD, the LF, and the LogCom. This finding seems to show that most of the selected methods, especially the seven techniques developed in the bag-of-paths framework and the two sigmoid versions of the commute time kernel, can perform equally well with only some parts of the information as with all of them, at least on the investigated datasets and following the stated methodology.

7.4.5 Investigation of the best performing method

In this subsection, we examine which of the combinations of the 16 best-ranked methods and the four feature sets (5%, 10%, 20% and Ker) provides

the best overall performance in our experimental setup. We also include in this comparison the results obtained by the three baseline methods (the SP, the A and the Q), in order to ensure that the selected methods actually perform better than these baselines.

To examine the results of these 76 methods, we once again use the Nemenyi test ranking [79]. However, we do not draw statistical conclusions because the number of methods compared is much higher than the number of data per method which could bias the conclusions. The ranking of the 76 methods is reported in Table 7.26.

From Table 7.26, it can be seen that the best-ranked method overall in our experimental setup is the PWSurp in Ker. In addition, we can point out that seven of the top-10 ranked methods comes from the bag-of-paths framework. The three other places are occupied by the SCCT in 10%F, 20%F and Ker. We can also notice that among the top-10 ranked methods, five are obtained in Ker and three in 5%F.

Regarding the baseline methods, they appear respectively for the first time in the ranking at positions 49 for the SP in Ker, 55 for the A in Ker and 65 for the Q in Ker. Between the rank 49 and 55, we find the nCor and the nCorH in 20%F, as well as the other combinations of the SP. While the methods ranked between the A and the Q in Ker are the LPI in 5%F and 10%F, the MRL in 10%F and Ker, the Heat in 20%F and Ker, and the other combinations of A. Below the rank 65 of the Q in Ker, we find the Heat in 5%F and 10%F, the RWWR in 10%F and 20%F, the MRL in 20%F, the ExpD for all the feature sets except Ker, as well as the other combinations of Q.

Overall, we can state that PWSurp in Ker is the best performing method in our experimental setup. Moreover, we can point out that the Heat, the ExpD, and the MRL in most of the feature sets are not better ranked than the three baseline methods in Ker.

7.5 Conclusion

In this chapter, we proposed an comprehensive comparison of 43 similarity and dissimilarity measures through graph-based semi-supervised classification tasks for four feature extraction sets.

The experimental comparison showed that the Poisson weighted surprisal distance with L1-normalized degree priors, introduced in Subsection 6.2.4 of Chapter 6, stands out the most from the other investigated methods, especially when using all the information contained in the kernel.

Furthermore, the comparison of the impact of the four extracted feature sets has shown that the performance of the two sigmoid versions of the commute time kernel and the seven best-ranked bag-of-paths methods do not decrease significantly with partial information about the graph structure, in

Method	Position	Mean rank	Method	Position	Mean rank
PWSurp-Ker	1	67.93	nCor-Ker	39	42.43
cBoPH-Ker	2	64.54	nCorH-Ker	40	41.32
PWSurp-5%F	3	61.75	MRL-5%F	41	35.00
cBoPH-5%F	4	61.14	nCor-10%F	42	33.93
FE-Ker	5	59.61	nCorH-10%F	43	33.93
FETsa-5%F	6	59.57	RWWR-5%F	44	32.36
SCCT-10%F	7	59.11	RWWR-Ker	45	32.36
FETsa-Ker	8	58.93	LPI-Ker	46	30.50
SCCT-20%F	9	58.86	LPI-20%F	47	27.71
SCCT-Ker	10	58.50	ExpD-Ker	48	27.71
FE-5%F	11	58.39	SP-Ker	49	27.07
PWSurp-20%F	12	57.18	nCor-20%F	50	26.64
SCCT-5%F	13	57.14	SP-5%F	51	25.57
RSPTsa-Ker	14	57.04	SP-20%F	52	25.36
PWSurp-10%F	15	55.79	nCorH-20%F	53	25.21
LF-Ker	16	55.29	SP-10%F	54	24.89
FE-20%F	17	54.79	A-Ker	55	23.79
cBoPH-10%F	18	54.57	LPI-10%F	56	23.64
RSPTsa-5%F	19	54.32	MRL-Ker	57	23.14
RSPTsa-20%F	20	54.29	A-5%F	58	21.07
LogCom-Ker	21	53.93	Heat-Ker	59	20.93
RSPTsa-10%F	22	53.07	LPI-5%F	60	20.43
FETsa-10%F	23	52.71	Heat-20%F	61	20.36
FETsa-20%F	24	52.57	A-10%F	62	20.07
cBoPH-20%F	25	51.93	A-20%F	63	19.86
LF-5%F	26	51.43	MRL-10%F	64	18.93
SCT-Ker	27	51.36	Q-Ker	65	18.57
FE-10%F	28	51.04	Q-5%F	66	18.00
SCT-5%F	29	50.43	ExpD-5%F	67	17.64
SCT-20%F	30	48.29	Heat-10%F	68	17.43
SCT-10%F	31	47.46	Heat-5%F	69	16.89
LogCom-20%F	32	46.96	ExpD-10%F	70	15.07
LF-20%F	33	46.75	Q-10%F	71	15.00
LF-10%F	34	44.93	Q-20%F	72	15.00
LogCom-10%F	35	44.82	ExpD-20%F	73	14.07
nCor-5%F	36	44.04	RWWR-10%F	74	12.64
nCorH-5%F	37	43.29	RWWR-20%F	75	7.71
LogCom-5%F	38	43.25	MRL-20%F	76	6.79

Table 7.26: Ranking of the 16 best-ranked methods and the three baseline methods according to Nemenyi test across all the data and the feature sets (the higher the mean rank, the better). The best-ranked method overall is the PWSurp in Ker.

comparison with all the available information in our semi-supervised classification tasks.

Future works will aim to improve our experimental setup. For instance, we could extend the number of datasets used in our experiment by including larger datasets, although not all the methods studied will scale and provide results, and also datasets from other domains than social networks or networks containing communities. Another idea to increase the dataset collection would be to generate additional datasets with the Lancichinetti-Fortunato-Radicchi (LFR) benchmark [165]. These additions would more clearly differentiate the limitations and performance of the different methods. Moreover, we could also increase the number of methods, the range of the parameter values, and the number of feature sets investigated to obtain more precise results.

Another considered future work would be to investigate other classifiers than the SVM to observe if our findings are transposable to other supervised classification models or if they are specific to the SVM. It would also be interesting to compare the performance of each classifier on all the feature sets. We already performed some preliminary comparisons between kernel SVM and graph neural networks [182, 186], like the graph convolutional neural network [152] and the diffusion convolutional neural network [13], with the help of master students in [42, 77] that lead to promising results.

Still another future work would be to explore why certain methods perform well on some graphs and what this says about those graphs. For this purpose, as already proposed in [133] for graph clustering, we could collect and generate with the LFR benchmark [165] a large number of graphs with different structures and properties. Then, we could study which method works best or worst, depending on these properties. Indeed, each similarity and dissimilarity measure captures various structural properties of the graph, and it would be worthwhile to study their impact on the results.

Graph-based fraud detection with the free energy distance

8

Analogous to the twentieth century being characterized by automobiles, airplanes, and electronics, the business and society of the twenty-first century are characterized by e-commerce of all kinds. Since its creation, global e-commerce sales have grown from a standing start to 2.43 trillion US dollars in 2017, bringing an enormous change in business firms, markets, and consumer behavior. E-commerce is expected to continue to grow rapidly over the following years, remaining the fastest growing form of commerce [167, 301].

With the emergence of e-commerce systems, the number of online credit card transactions has skyrocketed. However, not all of these transactions are legitimate – worldwide card fraud losses in 2017 reached 24.26 billion US dollars, an increase of 6.4% from 2016, and the forecast for the following years goes in the same direction [131]. This huge amount of loss has led to the development of a series of countermeasures to limit the number of frauds. Among these countermeasures, a distinction can be made between **fraud prevention system** (FPS) and **fraud detection system** (FDS). FPS describes all the actions taken to prevent fraud from occurring. In contrast, FDS comes into play, once FPS has failed, to identify perpetrated fraud as soon as possible [16, 33].

As it is described in [72], a FDS is composed of five different layers of control. These layers are illustrated in Fig. 8.1 and a summary of each layer, strongly inspired from [72], is presented in the following lines.

- **Terminal:** The terminal is the first layer that controls the PIN code, the number of attempts, the card status, the balance available, and the expenditure limit for each payment request/transaction. Specifically, these controls have to be performed in real-time (i.e. in a few milliseconds) for online transactions. If the transaction does not pass one of these security checks, the request is refused. Otherwise, it continues its way to the second layer of control.
- **Transaction-blocking rules:** The transaction-blocking rules are a set

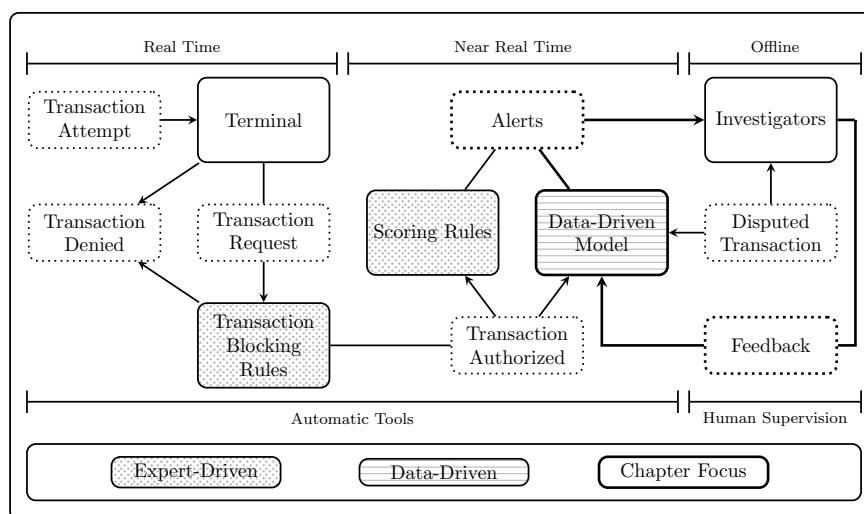


Figure 8.1: A scheme, strongly inspired from [72], illustrating the layers of control in a FDS.

of conditional statements (if/then/else), designed manually by the investigator (expert-driven), using the information available when the payment is requested. These rules should be quick to compute to guarantee real-time answers and very precise to avoid false alarms. Only the transactions passing all the blocking rules are authorized, the others are blocked. However, the fraud control does not stop there. All the data about the transactions are aggregated with new features (e.g. the average expenditure) and stacked in a feature vector. This new feature vector is the starting point of the third layer of control.

- **Scoring rules:** Scoring rules are also a set of conditional statements, manually designed by the investigator (expert-driven), but operating on the feature vector. These rules assign a risk score to each authorized transaction. A high score means that the transaction has great potential to be a fraud. As the scoring rules are designed based on the knowledge of the expert, they can only detect fraudulent strategies already well-known by these experts.
- **Data-driven model:** This fourth layer, on the contrary to the two previous ones, is purely data-driven and not expert-driven. It relies on classification models or other statistical models to predict the probability of being a fraud for each authorized transaction based on their feature vectors. A data-driven model is expected to analyze simultaneously multiple components of the feature vector to find rules, not necessarily interpretable, which go beyond the investigator experience

to detect fraudulent patterns. Only the transactions with a high probability of being a fraud or a large risk score generate alerts. This limited number of alerts is then reported to the final layer of control which is the investigators.

- **Investigators:** Investigators are human experts in analyzing credit card transactions and ensure the proper functioning of the expert-driven layers. On top of that, investigators also call cardholders to verify the veracity of the alerts and assign the label ‘genuine’ or ‘fraudulent’ to each of them. This information is then returned to the FDS in the form of feedback. Any card that has a transaction labeled as ‘fraudulent’ is immediately blocked, to avoid any further fraudulent activities. After that, investigators verify all the recent transactions of the compromised cards and provide feedback for each transaction. This means that an alert can potentially generate more than one feedback. All this process is tedious and time-consuming for investigators, who can therefore only handle a small number of alerts per day. That is why it is important that the data-driven model does not return too many false alarms, otherwise the investigators may no longer take the alarms into account.

Moreover, from Fig. 8.1, we can also observe that all the layers do not work at the same speed. The first two layers are designed to ensure a real-time answer and reduce as much as possible the number of false alarms. At this stage of the FDS, a false alarm leads to the transaction being denied, which is restrictive for the user and should be avoided at all costs. In classification terms, these layers avoid false positive (detecting fraud although not fraudulent in reality) results and thus favor a good precision at the expense of the recall.

While the next two layers are designed to provide near real-time answer and detect as many fraudulent transactions as possible, even if this means generating a larger number of false alarms. At this stage of the FDS, transactions are already authorized. Therefore, generating false alarms is less of a problem as long as their number is reasonable, so that investigators can verify all of them offline. In classification terms, these layers generate more false positive results than false negative results and thus favor a good recall at the expense of the precision.

Despite the complexity of FDS, the credit card fraud detection domain still presents several challenging issues [1, 46]. Firstly, there are millions of credit card transactions processed each day, creating a massive stream of data. That is why data mining and machine learning play an important role in FDS, as they are often applied to extract and uncover the hidden truth behind very large quantities of data [207]. Secondly, the data are unbalanced: there is (fortunately) a more prevalent number of genuine transactions than fraudulent

ones. The main risk with unbalanced classes is that the classifier tends to be overwhelmed by the majority class and to ignore the minority class [180]. Thirdly, the data are exposed to a **concept drift** as the habits and behaviors of the consumers and fraudsters change over time [72]. Finally, as we mentioned earlier, the FDS needs to process the acceptance check of an online credit card transaction within a few seconds (real-time) to decide whether to pursue the transaction or not [281].

This chapter focuses on automatically detecting fraudulent e-commerce transactions using network-related features and free energy distance [105, 156] (see Subsection 3.4.2). To deal with the large amount of data, a new way of computing the free energy distance on large, sparse, graphs, mainly based on the log-sum-exp trick interpretation of this distance proposed in [154], is investigated. Moreover, this chapter is based on a recent paper [170] which introduced several improvements to an existing collective inference algorithm called APATE [281]. More precisely, this algorithm starts from a defined number of known frauds and propagates the fraudulent influence through a graph to obtain a risk score, quantifying the fraudulent behavior for each transaction, cardholder, and merchant [281].

The main motivation of this chapter is to introduce the free energy distance into a FDS because this distance provides good results in several semi-supervised classification and clustering tasks [105, 122, 253, 254]. In short, the main contributions of this chapter are:

- ▶ an investigation of a new way to compute the free energy distance scaling on large, sparse, graphs [154];
- ▶ an application of this method to the fraud detection field through the adaptation of the existing FDS APATE [281];
- ▶ an experimental comparison between this method and others on a large real-world e-commerce credit card transaction dataset comprising more than 25 million transactions obtained from Worldline SA/NV.

The remainder of this chapter is organised as follows. Section 8.1 contains the related work. Section 8.2 introduces the proposed contributions. In Section 8.3, we present the experimental comparisons and analyze the results. Finally, Section 8.4 concludes the chapter.

8.1 Related work

Over the past few years, credit card fraud detection has generated a lot of interest and a wide range of techniques has been suggested. However, the number of publications available is only the tip of the iceberg. Indeed, credit

card issuers protect the sharing of data and most algorithms are produced in-house, concealing the details of the models [281].

Credit card fraud detection techniques can be seen as a classification problem. Therefore, it can be categorized into three broad types of learning: **supervised** (SL), **unsupervised** (USL) and **semi-supervised** (SSL). The most widespread approach is the supervised one which uses the information content in the labels, i.e. ‘fraud’ and ‘genuine’ in our case, to build a classification model. Common supervised methods are logistic regression [250], decision trees [250], Bayes minimum risk [18], support vector machines [73], meta-learning [46], case-based reasoning [292], genetic algorithms [88], hidden Markov models [29, 257], association rules [240], random forest [73] and, the most prevalent one at the moment, artificial neural networks [73, 140, 308]. Unlike supervised techniques, unsupervised learning does not use the class label to build the model, but simply extracts clusters of similar observations while maximizing the difference between these clusters. Common unsupervised methods include standard clustering methods, self-organizing map [305] and peer group analysis [291]. The interested reader is advised to consult [73, 207, 313] for more information.

The last category of classification techniques is semi-supervised learning which lies between supervised and unsupervised techniques, since it constructs predictive models using labeled samples together with a usually larger amount of unlabeled samples [103]. Some common semi-supervised methods are graph-based approaches, which consist of the creation of a graph model that reflects the relations included in the data and then transfers the labels on the graph to build a classification model [313] (see Chapter 7). Compared to the other two categories previously presented, there are few publications about semi-supervised methods applied to card fraud detection. Ramaki et al. [228] proposed a model on a semantic connection between data stored for every transaction fulfilled by a user, then represent it by ontology graph and finally store them in patterns databases. Cao et al. [45] suggested HitFraud, a collective fraud detection algorithm that captures the inter-transaction dependency based on a heterogeneous information network. Molloy et al. [196] presented a new approach to cross channel fraud detection based on feature extraction techniques applied to a graph of transactions. Finally, Lebichot et al. [170] proposed several improvements to an existing FDS, APATE, which spreads fraudulence influence through a graph by using a limited set of confirmed fraudulent transactions [281]. Our FDS is of this type.

As mentioned earlier, we base our approach on the previous work of Lebichot et al. [170] and on the methodology of APATE [281]. The rest of this section summarizes these two works.

APATE starts by building a real **tripartite** symmetric transaction cardholder merchant adjacency matrix $\mathbf{A}^{\text{tri}} = (a_{ij})$ (see Subsection 2.2.1 and

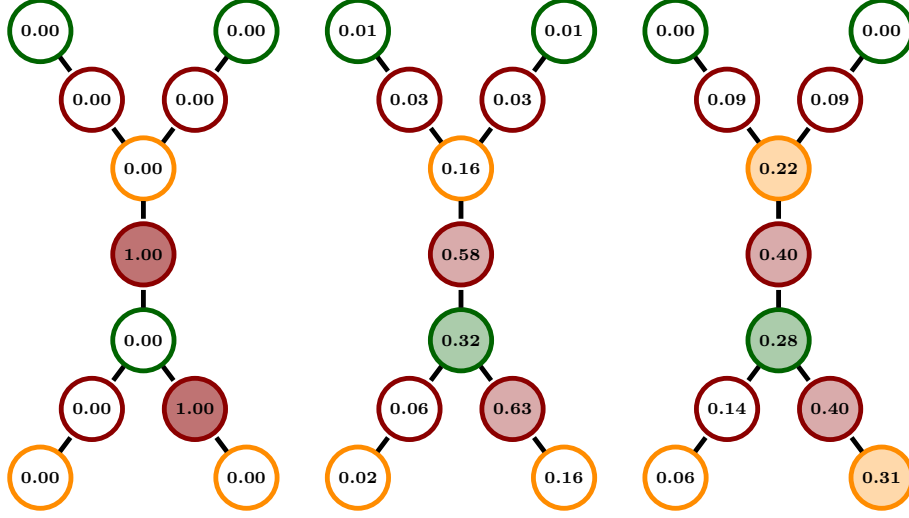


Figure 8.2: A small toy example of a tripartite graph. Left sub-figure shows a graph composed of five transactions (in red), including two fraudulent (risk score = 1), between three cardholders (in green) and three merchants (in orange). Central sub-figure represents the risk score obtained after running APATE with a $\alpha = 0.5$. A cardholder (risk score = 0.32) is identified as more fraudulent than the others. Right sub-figure represents the risk score obtained after running FraudsFree with a $\theta = 1$. A cardholder (risk score = 0.28) and two merchants (risk score = {0.22, 0.31}) are identified as more fraudulent than the others. Those risk scores are then used as features in a random forest for both methods.

Fig. 8.2) based on a list of time stamped, labeled transactions where each cardholder and merchant is known,

$$\mathbf{A}^{\text{tri}} = \begin{bmatrix} \mathbf{0}_{t \times t} & \mathbf{A}_{t \times c} & \mathbf{A}_{t \times m} \\ \mathbf{A}_{c \times t} & \mathbf{0}_{c \times c} & \mathbf{0}_{c \times m} \\ \mathbf{A}_{m \times t} & \mathbf{0}_{m \times c} & \mathbf{0}_{m \times m} \end{bmatrix} \quad (8.1)$$

where $\mathbf{A}_{c \times t}$ is a biadjacency matrix where cardholders are linked with their corresponding transactions, $\mathbf{A}_{m \times t}$ is a biadjacency matrix where merchants are linked with their corresponding transactions and $\mathbf{0}_{\dots \times \dots}$ is a correctly sized matrix full of 0s. Moreover, a column vector $\mathbf{z}_0^{\text{tri}} = [\mathbf{z}_0^{\text{Trx}}; \mathbf{z}_0^{\text{CH}}; \mathbf{z}_0^{\text{Mer}}]$, containing the risk score of each transaction (Trx), cardholder (CH) and merchant (Mer) is created and initialized with 0s, except for known fraudulent transactions (Trx) which are set to 1.

APATE also integrates a time decay factor in order to address the dynamic behaviour of fraud. Each non-zero entry of $\mathbf{A}^{\text{tri}}$ and $\mathbf{z}_0^{\text{tri}}$ is modified to express

the recency of the transaction [281]:

$$\begin{cases} [\mathbf{A}^{\text{tri}}]_{ij} = e^{-\gamma \cdot t([\mathbf{A}^{\text{tri}}]_{ij})} & \text{or 0 if no relation} \\ [\mathbf{z}_0^{\text{tri}}]_i = e^{-\gamma \cdot t([\mathbf{z}_0^{\text{tri}}]_i)} & \text{or 0 if no fraud} \end{cases} \quad (8.2)$$

where $t(\cdot)$ is the time passed since the transaction pursued, and γ is the decay constant factor set in such a way that the half-life, i.e. the time required for a quantity to reduce by half, of the exponential is: one day, one week and one month. In the end, we obtain four pairs of $\mathbf{A}^{\text{tri}}$ and $\mathbf{z}_0^{\text{tri}}$ corresponding to four different time windows : no decay, day decay, or short term (ST), week decay, or medium term (MT) and month decay, or long term (LT).

Then for each of the four time windows, in order to spread fraudulence influence through the tripartite graph, the vector $\mathbf{z}_0^{\text{tri}}$ is updated following an iterative procedure similar to the **PageRank** algorithm [214], namely the **random walk with restart** (RWWR) [273]:

$$\mathbf{z}_\tau^{\text{tri}} = \alpha \mathbf{P}^T \mathbf{z}_{\tau-1}^{\text{tri}} + (1 - \alpha) \mathbf{z}_0^{\text{tri}} \quad (8.3)$$

where τ is the iteration number, $\mathbf{P} = (p_{ij}) = (\frac{a_{ij}}{a_{i\bullet}})$ is the transition probability matrix (see Subsection 2.2.4) associated with $\mathbf{A}^{\text{tri}}$, α is the probability to continue the walk, and symmetrically $(1 - \alpha)$ is the probability to restart the walk from a fraudulent transaction. Eq. 8.3 is iterated until convergence, to reach $\mathbf{z}_{\tau^*}^{\text{tri}}$ (where τ^* stands for τ at convergence) from which three new feature vectors can be extracted, $\mathbf{z}_{\tau^*}^{\text{Trx}}$, $\mathbf{z}_{\tau^*}^{\text{CH}}$ and $\mathbf{z}_{\tau^*}^{\text{Mer}}$. These three features correspond to a risk measure for each transaction, cardholder and merchant respectively. Therefore, for each transaction, there are 12 new graph based features created. A toy example can be found on Fig. 8.2.

As this procedure cannot be computed in a few seconds, the scores for each transaction, cardholder and merchant are only re-estimated once a day or once per hour, in order to analyze transactions that will occur during the day. In cases where new merchants or cardholders appear, their scores are set to zero as nothing can be inferred from the past graph data. Regarding new transactions that did not yet happen in the past between a certain cardholder and merchant, the score of the transaction is approximated by using the scores of its direct neighbourhood, which is done through the formula [281]:

$$\text{score}(\text{Trx}_{i,k}) = \frac{1}{\sum_{j=1}^n a_{ij} + 1} \text{score}(\text{Mer}_i) + \frac{1}{\sum_{j=1}^n a_{kj} + 1} \text{score}(\text{CH}_k) \quad (8.4)$$

where $\text{score}(\text{Trx}_{i,k})$ is the score of the new transactions between cardholder k and merchant i , $\text{score}(\text{CH}_k)$ is the score of cardholder k , and $\text{score}(\text{Mer}_i)$ is the score of merchant i . Furthermore, new transactions are added to $\mathbf{A}^{\text{tri}}$ with an edge weight of 1 for each time window as it represents a current relationship.

	Variable name	Description
Transaction-related	inBEL/EURO/OTH	Issuing region : Belgium/Europa/World
	TX AMOUNT	Amount of transaction
	TX 3D SECURE	Transaction used 3D secure
	AGE	Age of cardholder
	langNED/FRE/OTH	Cardholder language : Dutch/French/Other
	isMAL/FEM	Cardholder is Male/Female
	isFoM	Cardholder gender is unknown
	BROKER	Code of card provider
	cardMCD/VIS/OTH	Card is a Mastercard/Visa/Other
Graph-based	01 Mer score	Merchant risk score (boolean, no time damping)
	ST/MT/LT Mer score	Day/week/month decay merchant risk score (3 features)
	01 CH score	Cardholder risk score (boolean, no time damping)
	ST/MT/LT CH score	Day/week/month decay cardholder risk score (3 features)
	01 Trx score	Cardholder risk score (boolean, no time damping)
	ST/MT/LT Trx score	Day/week/month decay transaction risk score (3 features)
	TX FRAUD	Target variable : Fraud/Genuine

Table 8.1: Features used by the random forest classifier. Notice that each transaction is linked with a cardholder and with a merchant at a certain date, but this information is only used to build the tripartite graph [170].

Finally, APATE combines those 12 graph-based features with the transaction related features initially present in the dataset (see Table 8.1), and uses these as inputs of a random forest classifier.

While APATE is showing good performance, according to Lebichot et al. [170], it can be improved in three ways: by dealing with hubs, introducing a time gap, and including investigators feedback.

The first way of improvement consists of dealing with hubs, which are nodes having a high degree (see Subsection 2.2.3), i.e. a large number of links with other nodes. Due to their connections to a lot of transactions, hubs tend to accumulate a high-risk score. A simple solution to counterbalance this accumulation is to divide the risk score by the node degree after convergence of Eq. 8.3, which gives

$$\mathbf{z}_{\tau*}^{\text{tri}} = \mathbf{z}_{\tau*}^{\text{tri}} \div \mathbf{d}^{\text{tri}} \quad (8.5)$$

where $\mathbf{d}^{\text{tri}}$ is the degree vector associated to $\mathbf{A}^{\text{tri}}$ (see Subsection 2.2.3) and $\div$ is the elementwise division.

As mentioned in [170], to counterbalance the hubness effect, it is also possible to divide Eq. 8.3 by any power of the degree vector or by considering different powers of degrees for the three types of nodes present in the tripartite graph (transaction, cardholder, and merchant). However, none of these other solutions investigated in [170] provide better results than simply dividing by the node degree.

Moreover, Lebichot et al. [170] relate the solution presented in Eq. 8.5 to the **Regularized Commute Time Kernel** (RCTK) [310] (see Subsection 7.1.4) in the sense that the elements of this kernel have the same interpretation as for the RWWR used in APATE. Indeed, it has been shown that for undi-

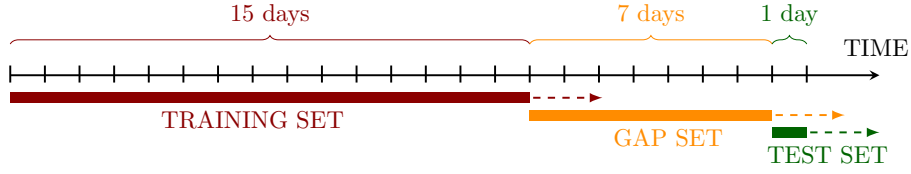


Figure 8.3: Real-world FDS scenario composed of three set of data. A training set of 15 days where the fraud tags are known. A gap set of 7 days where the fraud tags are unknown. A test set of 1 day where the fraud tag is predicted. This scenario is repeated each day until the last day.

rected graphs, the elements of the RCTK correspond to the score obtained by the RWWR divided by the degree of the nodes [103]. Therefore, Lebichot et al. recommended the use of RCTK to deal with the problem of hubs.

Their second proposal is to introduce a time gap between the training set and the test set. Lebichot et al. [170] explain that, in most real FDS, the model cannot be based on the past few days, as is proposed in APATE, for two reasons. The first reason is that, in a real setting, the fraudulent transaction tags cannot be known without human investigator feedback. However, this feedback takes usually several days, mainly because it is often the cardholders that report undetected fraud. The second reason is that the strategy of the fraudsters changes over time and so it is less reliable to build the model on old data. In arrangement with field experts, they designed a real-world scenario with a training set of 15 days and a gap set of 7 days as illustrated in Fig. 8.3 [34, 71].

This new configuration corresponds to a **semi-supervised learning scheme** (SSL) where the data used to build the graph is partially labeled. More precisely, the graph is built based on the **training set** - where the transaction fraud labels are known - and the **gap set** - where the transaction fraud labels are unknown. However, the random forest classifier is only trained on the training set. Leaving aside the gap set, the configuration is the same as APATE which corresponds to a **supervised learning scheme** (SL).

The third way of improvement consists of including **feedback** from the investigators on the predictions of the previous days. Even if it appears clear, given the second proposal, that it is impossible to know all fraud tags for the gap set, it is still conceivable that a fraction of previous alerts has been confirmed or overturned by human investigators (which is indeed the case in practice). We refer later to this option as FB (for feedback) in Section 8.3.

8.2 Free energy distance

As discussed previously, the main contributions of this chapter are three-fold, (1) to investigate a way of computing the free energy distance scaling on large, sparse, graphs, then (2) to incorporate the free energy framework into a FDS and finally (3) to evaluate its performance on a real-world large-scale fraud detection problem. In this section, we start by providing a short overview of the free energy distance and its properties, before discussing its implementation in the FDS. Note that this distance measure between nodes obtained good results in several semi-supervised classification and clustering tasks [105, 122, 253, 254].

8.2.1 Background

As already presented in Subsection 3.4.2, the **free energy distance** [156], also known as the **bag-of-paths potential distance** [105], is a distance measure between nodes of a directed, strongly connected, graph based on the bag-of-hitting-paths model. It is usually introduced by considering a statistical physics framework where it corresponds to the minimized free energy¹ of the bag-of-paths system connecting two nodes, but we consider here a more intuitive explanation.

We already introduced the random walk on the graph whose transition probability matrix is $\mathbf{P}$, see Eq. 8.3. Recall that its elements are nonnegative and each of its rows sums to one. The free energy distance is computed to some nodes of interest $\mathcal{A}$ (in our application, the fraudulent nodes), called **target nodes**. These nodes are made killing and absorbing by setting the corresponding rows in the transition probability matrix to zero. Note that if the original graph is not strongly connected, we used a common trick, namely to add a new absorbing, killing, (sink) node connected to the set of target nodes $\mathcal{A}$ with a directed link. In addition, we also assume that there is a nonnegative cost $c_{ij} \geq 0$ associated with each edge (i, j) of the graph with $\mathbf{C} = (c_{ij})$ being the cost matrix (see Subsection 2.2.2). The cost on an edge is assigned depending on the application and quantifies, in the model, the difficulty of following this edge in the random walk [103]. In our application, we fixed the cost to $c_{ij} = 1/a_{ij}$.

In this context, the **directed free energy dissimilarity** $\phi_{i\mathcal{A}}$ between any node $s = i$ (starting node) and the absorbing target nodes in $\mathcal{A}$ is simply $-\frac{1}{\theta} \log \mathbb{P}(\text{reaching}(\mathcal{A}) | s = i)$, that is, minus the logarithm of the probability of surviving during the killed random walk, i.e., of reaching an absorbing node without being killed during the walk [105]. Furthermore, as a reminder, the

¹Expected total cost of the paths plus scaled relative entropy of the probability distribution of following these paths (see [156] for details).

free energy distance between two nodes i and j is obtained by $\Delta_{ij} = \frac{\phi_{ij} + \phi_{ji}}{2}$ (see Subsection 3.4.2 for more details).

Generally, as shown in Subsection 3.4.2, the free energy distance between all pairs of nodes is computed by performing a matrix inversion [156] which is not applicable to this context. However, François et al. [105] showed that the directed free energy distance to a unique, fixed, target node t (the set of absorbing nodes reduces to node t) can also be computed thanks to an extension of the Bellman-Ford formula:

$$\phi_{it}(\tau+1) = \begin{cases} -\frac{1}{\theta} \log \left[\sum_{j \in \text{Succ}(i)} p_{ij} \exp \left[-\theta(c_{ij} + \phi_{jt}(\tau)) \right] \right] & \text{if } i \neq t \\ 0 & \text{if } i = t \end{cases} \quad (8.6)$$

where τ is the iteration number. This expression uses the softmin, or log-sum-exp, operator [200], $\text{softmax}_{\theta, \mathbf{q}}(\mathbf{x}) = -\frac{1}{\theta} \log \sum_{j=1}^n q_j \exp[-\theta x_j]$ (with $q_j \geq 0$ and $\sum_{j=1}^n q_j = 1$) where, in the present context of Eq. 8.6, $q_j = p_{ij}$ and $x_j = (c_{ij} + \phi_{jt}(\tau))$. This operator interpolates between the minimum and the weighted average of the values x_j with weights q_j . In fact, Eq. 8.6 is nothing else than the Bellman-Ford formula (based on dynamic programming; see, e.g., [111]) where the minimum operator is replaced by the soft minimum operator. Eq. 8.6 can be iterated until convergence to the directed free energy distances. After convergence, ϕ_{it} contains $-\frac{1}{\theta} \log$ of the probability of surviving during a killed random walk from i to t with transition matrix $\mathbf{W}$ (see Eq. 3.5 in Section 3.1) [105].

However, this reformulation of the directed free energy distance suffers from underflow problems when $\theta(c_{ij} + \phi_{jt}(\tau))$ is too large for all the successors of a node i . In that case, the term inside the logarithm of Eq. 8.6 risks underflowing to zero, providing an infinite value for $\phi_{it}(\tau+1)$. Note that this drawback can be easily solved thanks to some specific techniques which are explained in the next subsection.

8.2.2 Computing the directed free energy distance on large graphs

In order to scale the computation of the free energy on large, sparse, graphs, we use Eq. 8.6 and rely on two different ideas: (i) the **log-sum-exp** trick and (ii) to **bound the length** of the set of paths on which the distance is computed. This second point brings another benefit: it allows to tune the length of the walks, which has been shown to improve the performance in some situations (see, e.g., [43, 189]).

The log-sum-exp trick [154, 200] aims at pre-computing

$$x^* = \min_{j \in \{1 \dots n\}} (x_j) \quad (8.7)$$

where $x_j = (c_{ij} + \phi_{jt}(\tau))$. Then, from Eq. 8.6, x^* is subtracted to all the x_j and moved out from the logarithm, leading to the form

$$\phi_{it}(\tau + 1) = x^* - \frac{1}{\theta} \log \left[\sum_{j \in \text{Succ}(i)} p_{ij} \exp \left[-\theta(x_j - x^*) \right] \right] \text{ for } i \neq t \quad (8.8)$$

This formulation introduced in [154] ensures that the value inside the logarithm is systematically greater or equal to p_{ij} because there is always at least one successor for which $\theta(x_j - x^*) = 0$. Moreover, to accelerate the computation, the terms in the summation for which $\theta(x_j - x^*)$ is too large (exceeds a certain threshold) can be neglected. These operations are a kind of pruning and have two benefits: they reduce significantly the number of terms to be computed and avoid numerical underflow problems.

For completeness, we could also rearrange the Eq. 8.6 by moving the transition probabilities p_{ij} inside the exponential before performing the log-sum-exp trick. In that case, the Eq. 8.6 becomes

$$\phi_{it}(\tau + 1) = -\frac{1}{\theta} \log \left[\sum_{j \in \text{Succ}(i)} \exp \left[-\theta(c_{ij} + \phi_{jt}(\tau) - \log(p_{ij})/\theta) \right] \right] \text{ for } i \neq t \quad (8.9)$$

where the value x_j can be defined as $x_j = c_{ij} + \phi_{jt}(\tau) - \log(p_{ij})/\theta$. Then, the log-sum-exp trick can be applied, which gives

$$\phi_{it}(\tau + 1) = x^* - \frac{1}{\theta} \log \left[\sum_{j \in \text{Succ}(i)} \exp \left[-\theta(x_j - x^*) \right] \right] \text{ for } i \neq t \quad (8.10)$$

In comparison with the formulation of Eq. 8.8, this formulation ensures that the value inside the logarithm is systematically greater or equal to 1. Nevertheless, we did not use this formulation in our experiments as the principle is very similar.

Whereas the standard bag-of-paths framework (see Chapter 3) is based on paths of unbounded length (from 0 to ∞), our second technique considers only walks bounded by a given length ℓ [43, 189]. This is done by defining a directed lattice L unfolding the original graph G in terms of increasing walk lengths. More precisely, this lattice is made of the graph nodes repeated at walk lengths $\tau = 0, 1, \dots, \ell$, usually with $\ell \ll n$ [189] (see Fig. 8.4). Then, transitions are only allowed from nodes at walk length τ to successor nodes at length $\tau + 1$ through the transition matrix $\mathbf{P}$ of the killed random walk associated with the graph. This lattice represents a bounded random walk on the graph G where walks' lengths are in the interval $[0, \ell]$.

In this context, on lattice L and considering now a set of target nodes $\mathcal{A}$, Eq. 8.6 becomes, for the initialization of the distances at $\tau = 0$, corresponding

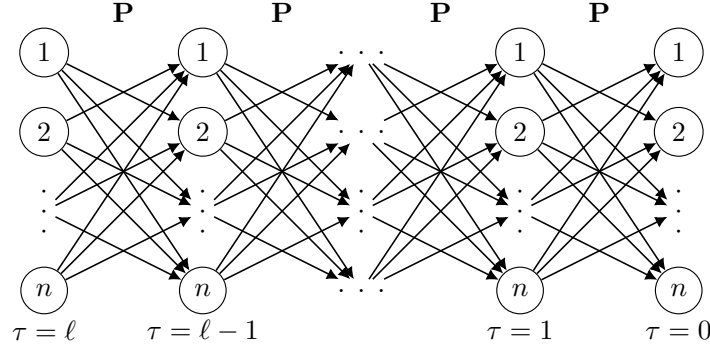


Figure 8.4: The directed lattice derived from the original graph. It only considers walks of length up to ℓ .

to zero-length walks,

$$\phi_{i\mathcal{A}}(0) = \begin{cases} \infty & \text{if } i \notin \mathcal{A} \\ 0 & \text{if } i \in \mathcal{A} \end{cases} \quad (8.11)$$

Moreover, $\phi_{i\mathcal{A}}(\tau)$ contains the directed free energy distance from node i to the absorbing nodes in $\mathcal{A}$ when considering walks up to length τ . A resulting distance of ∞ means that no walk of length up to τ exists between node i and a node in $\mathcal{A}$ – absorbing nodes cannot be reached in τ steps. Then, for walk length $\tau > 0$,

$$\phi_{i\mathcal{A}}(\tau+1) = \begin{cases} -\frac{1}{\theta} \log \left[\sum_{j \in \text{Succ}(i)} p_{ij} \exp \left[-\theta(c_{ij} + \phi_{j\mathcal{A}}(\tau)) \right] \right] & \text{if } i \notin \mathcal{A} \\ 0 & \text{if } i \in \mathcal{A} \end{cases} \quad (8.12)$$

This recurrence relation defines the directed free energy distance to target nodes $\mathcal{A}$ that is used in our fraud detection application.

8.2.3 Application to the fraud detection problem

In order to incorporate the free energy (FE) framework into FDS APATE to create the FDS called FraudsFree (FF), we introduce some further modifications.

The first modification is related to the computation of the risk score vector $\mathbf{z}_{\tau*}^{\text{tri}} = \phi_{\tau*}^{\text{tri}}$ presented in Eq. 8.3. For that, we use Eq. 8.12 considering each known fraud (Trx) as an absorbing node $a \in \mathcal{A}$. We iterate this equation until convergence to obtain the distance between all the nodes i and the set $\mathcal{A}$ which corresponds to the risk score of each transaction, cardholder, and merchant. At this point, a score near 0 represents a fraud and a high value

represents a genuine transaction. To keep the same interpretation of the risk score as in APATE², we apply the following rescaling

$$\phi_{\tau*}^{\text{tri}} \leftarrow \max(\phi_{\tau*}^{\text{tri}}) - \phi_{\tau*}^{\text{tri}} \quad (8.13)$$

where $\leftarrow$ defines the update of a quantity. A toy example can be found on Fig. 8.2.

As for APATE [281] (see Section 8.1), we set the score of a new transaction j that did not yet occur in the past between a cardholder k and a merchant i via Eq. 8.6, which provides the following expression

$$\begin{aligned} \text{score}(Tx_j) = & -\frac{1}{\theta} \log \left[p_{ji} \exp \left[-\theta(c_{ji} + \text{score}(Mer_i)) \right] \right] \\ & -\frac{1}{\theta} \log \left[p_{jk} \exp \left[-\theta(c_{jk} + \text{score}(CH_k)) \right] \right] \end{aligned} \quad (8.14)$$

where $p_{ji} = p_{jk} = 0.5$ because a transaction is linked by construction with one merchant and one cardholder and we fixed $c_{ji} = c_{jk} = 1$ as the transaction happens now (no decay), but this is still a degree of freedom of our method that we left for future work.

8.3 Experimental comparisons and discussion

In this section, we evaluate the performance of our FraudsFree model in a real-world fraud detection problem in Subsection 8.3.1. Then, we analyze the impact of the number of iterations on the results of the FraudsFree and the Regularized Commute Time Kernel in Subsection 8.3.2. Finally, the time complexity of FraudsFree is presented in Subsection 8.3.3.

8.3.1 Real-world fraud detection study case

To evaluate our approach following the methodology of [170], we perform a comparison between the different versions of our FraudsFree (FF) model and the other variations of APATE (Random Walk With Restart (RWWR) and Regularized Commute Time Kernel (RCTK)), in supervised (SL) and semi-supervised learning (SSL) with and without feedback (FB) on the same real-world e-commerce credit card transaction dataset as [170]. The dataset contains 16 socio-demographic features on 25,445,744 e-commerce transactions gathered during 139 days. The data are highly imbalanced, with only 78,119 frauds among the transactions ($< 0.31\%$). This dataset does not focus on a certain type of card fraud but contains all reported fraudulent transactions in the investigated time period [170]. Besides the 16 original features, a set of 12

²So that a score near 0 represents a genuine transaction and a high value represents a fraud. The higher the score, the higher the risk.

Acronym	Description
RWWR	Random Walk With Restart
RCTK	Regularized Commute Time Kernel
FF	FraudsFree
SL	Supervised Learning
SSL	Semi-Supervised Learning
FB	Including investigators feedback

Table 8.2: Summary of all the acronyms presented in Section 8.1, 8.2 and 8.3.

graph-based features is created for each method. A sample of this dataset is available on www.kaggle.com/mlg-ulb/creditcardfraud but the data are anonymized and the transactions are not presented day by day. Finally, all these features are fed into a class-rebalanced random forest with 400 trees following two different setups. Each tree is built based on a random selection of four features of the original dataset and four graph-based features.

To assess the performance of each method, we selected two measures. In accordance with field experts, we chose the Precision@100 (Pr@100) [270]. From that, two performance indicators were derived to the Pr@100 in terms of transactions (Trx Pr@100) and in terms of cards (Card Pr@100) [170]. According to Lebichot et al. [170], the Card Pr@100 is a more realistic metric as it corresponds to the normal workload for a human investigators team. Therefore, we chose to keep only the Card Pr@100 as a performance indicator as it is more pertinent for our problem. More precisely, we selected the most fraudulent transactions according to the model until we screened 100 cards. As a second measure, we used the average time in seconds required to create the tripartite graph and extract the 12 features between Day 41 and Day 139.

Regarding the hyperparameters, the RWWR and RCTK methods consider tuning values of $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.85, 0.9\}$ and, for the FF methods $\theta = \{0.1, 0.5, 1, 2, 3, 5, 10\}$. For each method, we tuned its parameter based on the mean Card Pr@100 between Day 30 and Day 40. The results for the best parameter is presented in Table 8.3³. We obtained these results by applying a sliding window technique in accordance with expert knowledge [170]. We set 15 days in the training set, 7 days in the gapset (see Section 8.1 and [170]), 1 day in the test set and we shifted the sliding window day by day. Furthermore, in order to exploit all the properties of the bounded FE, we fixed the number of iterations (the walk length ℓ) to five for the best FF-based method. This number comes from the recent work of Backstrom et al. [15] on the average degree of separation in a network. They observed that the average distance between two nodes of a graph is 4.74 which corresponds to 3.74 degrees of

³Numerical values differ from [170] because the dataset was further curated: some obvious fraud cases were removed.

Classifier	Hubs	Learning	Feedback	Card Pr@100	Time	Best parameter
Only Dataset Features	No	SL	No	2.16	/	/
RWWR SL = APATE	No	SL	No	18.19	94.82	0.1
RWWR SSL	No	SSL	No	17.70	150.35	0.9
RWWR SSL+FB	No	SSL	Yes	20.90	150.65	0.9
RCTK SL	Yes	SL	No	21.49	94.58	0.9
RCTK SSL	Yes	SSL	No	14.50	149.52	0.1
RCTK SSL+FB	Yes	SSL	Yes	24.03	152.69	0.7
RCTK SSL+FB 5 ITER	Yes	SSL	Yes	22.80	47.43	0.9
FF SL	No	SL	No	16.13	124.91	5
FF SSL	No	SSL	No	15.96	278.22	5
FF SSL+FB	No	SSL	Yes	24.20	309.29	0.5
FF SSL+FB 5 ITER	No	SSL	Yes	24.01	82.59	0.5

Table 8.3: Mean Card Pr@100 between Day 41 and Day 139 for each of the 12 methods (see Table 8.2 for acronyms). The average time in seconds required to create the tripartite graph and extract the 12 features for each day is also reported for each method except for the first one. The FF SSL+FB obtains the best Card Pr@100 and the RCTK SSL+FB 5 ITER is the fastest.

separation. Because this number of iterations allows propagating the fraud through the graph, we thought that it is a good trade-off between the Card Pr@100 and the computational time. For the sake of comparison, we also limited the number of iterations of the best APATE variation (RCTK SSL+FB) to the same number.

Note that we also reported the result obtained by the classifier using only the original features of the dataset in Table 8.3. However, we decided not to compare it with the other methods as it shows obvious differences. Nevertheless, this poor result does not mean that those features are information-less. Rather, several studies on the same dataset [70] and on others [17] show that used feature engineering can also lead to interesting results in fraud detection.

To analyze the results of Table 8.3, we use a nonparametric Friedman-Nemenyi statistical test and a Wilcoxon signed-ranks tests [79]. We perform all statistical tests at a level of confidence of 95%, which amounts to taking an α of 0.05. From the results of the Nemenyi test illustrated in Fig. 8.5, four methods perform equivalently in terms of Card Pr@100: the FE SSL+FB, the FF SSL+FB 5 ITER, the RCTK SSL+FB, and the RCTK SSL+FB 5 ITER. However, the Wilcoxon tests report that RCTK SSL+FB 5 ITER is significantly inferior to the three other methods in one-to-one comparisons (with respective p -values of 0.0055, 0.0252, and 0.0030). Even if we cannot ensure a statistical difference between the top three, we still observe that there are some differences in terms of their computation times. The FF SSL+FB 5 ITER method is the fastest of the top three with a reduction of 45.91% in computation time compared to the best method proposed by Lebichot et al. [170], RCTK SSL+FB. All results were obtained with Matlab (version R2020a) running on Intel Core i7-8750H CPU@4.10GHz and 32 GB of RAM.

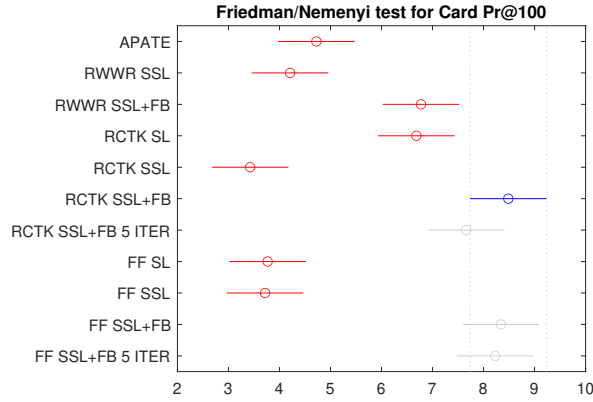


Figure 8.5: Mean ranks and 95% Nemenyi confidence intervals for the 11 methods (see Table 8.3) based on the Card Pr@100. Two methods are considered as significantly different if their confidence intervals do not overlap. The axis-x unit is the average rank of the methods. The higher the rank, the better the method. The best method highlighted in blue is the RCTK SSL+FB.

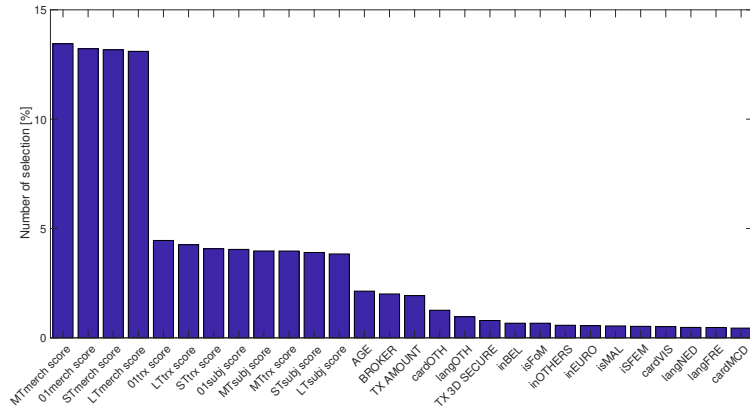


Figure 8.6: Variables selected by the random forest classifier for the RCTK SSL+FB model for all days. The four merchant risk scores are the most selected features. See Table 8.1 for acronyms.

Furthermore, from Figs. 8.6 and 8.7, it can be observed that the variables selected the most often by the random forest classifier for both the RCTK SSL+FB and the FF SSL+FB are the four merchant risk scores. Then comes, in the same proportion, the four transaction and four card user risk scores, followed by the original features of the dataset.

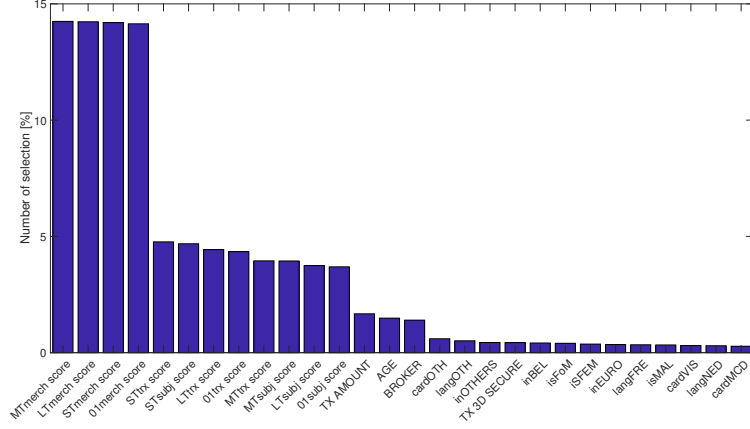


Figure 8.7: Variables selected by the random forest classifier for the FF SSL+FB model for all days. The four merchant risk scores are the most selected features. See Table 8.1 for acronyms.

8.3.2 Impact analysis of the number of iterations

Previously, we limited the number of iterations of the FF SSL+FB and the RCTK SSL+FB to five, considering this number as an external parameter of the model which can be chosen based on some properties on the network like the average degree of separation [15]. However, the number of iterations can also be considered as a parameter of the model, denoted by τ , which can be tuned in the same way as α and θ .

Following this reasoning, we investigated several values for the parameter $\tau = \{1, 2, 3, 5, 10, 20, 30, 50, 100\}$ for both the FF SSL+FB and the RCTK SSL+FB methods in addition of the previous hyperparameter. As before, we tuned these parameters based on the mean Card Pr@100 between Day 30 and Day 40 using the sliding windows technique. Fig. 8.8 shows the evolution of the mean Card Pr@100 provided by the RCTK SSL+FB and the FF SSL+FB with the number of iterations for each α and θ values.

From these two figures, it can be observed that our previous choice of five iterations has good performance. Indeed for both methods, this number of iterations corresponds to the maximum global mean Card Pr@100, and therefore to the optimal parameter, for $\alpha = 0.9$ and $\theta = 0.5$. Nevertheless, five iterations do not ensure the stability of the results as the lines continue to oscillate after this number for all values of α and θ . Taking a closer look, we observe that most of the lines are leveling out around 30 iterations for the RCTK SSL+FB and 50 iterations for the FF SSL+FB indicating a stabilization of the results and a convergence of the algorithms.

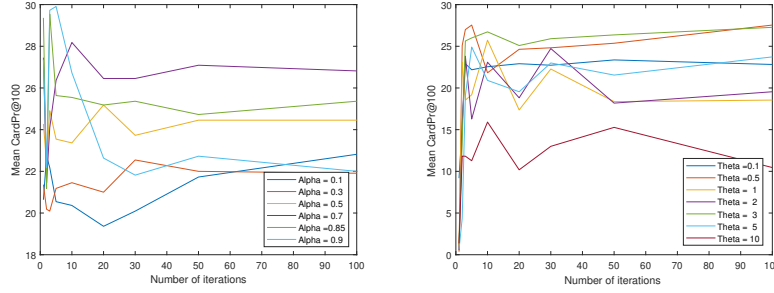


Figure 8.8: Evolution of the mean Card Pr@100 score provided by the RCTK SSL+FB (left) and the FF SSL+FB (right) with the number of iterations for each alpha and theta parameter values between Day 30 and Day 40. The maximum global mean Card Pr@100 is reached for both methods at five iterations.

As limiting the number of iterations leads to interesting results, we also try to simply use a system of majority vote to classify a transaction based on its neighborhood for various radius. In other words, we classify a transaction as fraud if the majority of its neighbors in a given radius are frauds. Nevertheless, we only achieve to obtain at maximum a mean CardPr@100 of 7.18 which is low compared to our other results.

To conclude, in view of the results, it seems that tuning the number of iterations as a parameter of the model instead of an external parameter leads to the same results in terms of mean Card Pr@100 in our configuration. This is interesting because tuning the number of iterations, rather than determining it a priori, slows all the methods, which is not desirable in a near-real-time fraud detection problem.

8.3.3 Time complexity analysis

Before diving into the details of the FF complexity time, it is important to recall that, in this chapter, we focused on a new way of computing the free energy distance scale on large sparse graphs. Therefore, the whole implementation of the FF has been limited to light and sparse operations reducing its complexity time.

Looking at Eq. 8.12, we can observe that the bottleneck is the sum over the successors of each node which is repeated until convergence. The corresponding time complexity is $2k \cdot \mathcal{O}(m)$ where m is the number of edges and k is the number of iterations needed to converge. In addition, the time complexity can also be expressed in terms of the number of transactions n_{Tx} as $k \cdot \mathcal{O}(n_{\text{Tx}})$. Indeed, the number of edges m and the number of transactions n_{Tx} are linearly related by the relation $n_{\text{Tx}} = 2m$. The time complexity of the FF is thus quite similar to that of the RCTK which is $l \cdot \mathcal{O}(n_{\text{Tx}})$ [169]

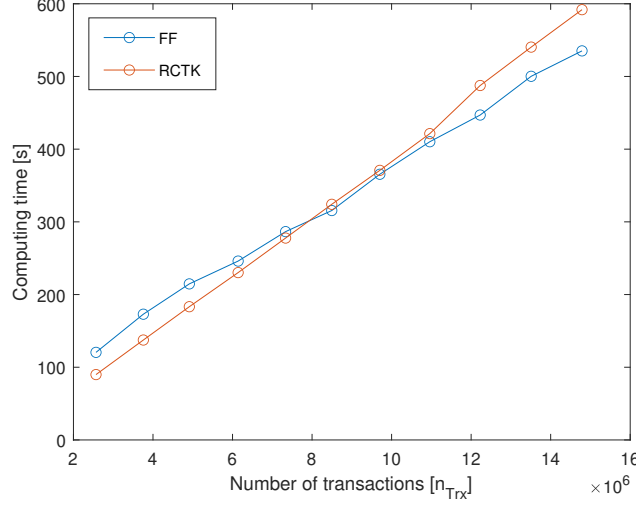


Figure 8.9: Evolution of the time required to create the tripartite graph and extract the 12 features depending on the number of transactions n_{Trx} for the FF ($\theta = 0.5$) and the RCTK ($\alpha = 0.7$). The time complexity of both methods is of the order of $\mathcal{O}(n_{\text{Trx}})$.

where l is the number of iteration required for Eq. 8.3 to converge.

As illustration, we reported the evolution of the time required to create the tripartite graph and extract the 12 features depending on the number of transactions n_{Trx} for the FF and the RCTK in Fig. 8.9. From this figure, we can see that the evolution of the computing time almost follows a regression line for both methods. Moreover, we can also observe that the FF takes less time than the RCTK from 8 million transactions.

8.4 Conclusion

In this chapter, we introduced the fraud detection system (FDS) called Frauds-Free by incorporating the free energy distance into the existing FDS APATE [281]. To develop the FDS FraudsFree, we investigated a version of the free energy distance that scales on large, sparse, graphs. This version of the free energy distance relies on an extension of the Bellman-Ford formula proposed by [105], the log-sum-exp trick [154], and bounding the length of the set of paths on which the distance is computed.

The experimental comparison on the real-world e-commerce credit card transaction dataset provided by Worldline SA/NV showed that the Frauds-Free is competitive with the best method previously developed in Lebichot et al. [170], namely the regularized commute time kernel in semi-supervised

learning including investigators feedback (RCTK SSL+FB), in terms of Pr@100 score on fraudulent card prediction with a significant speed-up in computation when the length of the set of paths is bounded to five.

Therefore, thanks to the properties of this version of the free energy distance, we managed to reduce the computational time and improve the scalability of the FDS. More precisely, we demonstrated that the time-complexity of the FraudsFree without bounding the length of the set of paths is $\mathcal{O}(n)$. This small time-complexity indicates that the free energy distance can be used in many other real-world applications involving large graphs.

One considered future work would be to study whether the combination of FraudsFree and the RCTK gives better results than those achieved by these two methods separately.

Another future work would be to compare the performance of FraudsFree with feature engineering approaches [17, 70]. Moreover, we could combine feature engineering approach with our graph-based approach and evaluate the impact on the FDS performance.

Still another avenue that could be explored is to determine if our approach is complementary, or just redundant, to the existing fraud defense lines of Worldline SA/NV.

We also plan to deal with the hub nodes by directly modifying the cost matrix, as it has shown good results in other studies [170].

As explained in Chapter 1, the bag-of-path framework, presented in Chapter 3, has been in development for the past decade. In this thesis, we have contributed to this development by proposing several theoretical and practical extensions.

This final chapter is devoted to review the content of this thesis and is organized as follows. Section 9.1 summarizes the content and the contributions of each previous chapter. Then, Section 9.2 discusses the limitations of this thesis. Finally, Section 9.3 is the conclusion of both this chapter and this thesis.

9.1 Content and contributions

The content of this thesis has been divided into nine chapters including this one. The first three chapters present the motivations for this work and the key concepts necessary to understand the following five chapters, but bring no original contribution to the literature. The following five chapters include the original contributions of this thesis and are mainly based on scientific papers produced during the thesis. The content and contributions of each of the previous eight chapters are detailed below. A summary of the main contributions of this thesis can be found in Table 9.1.

- Chapter 1 is a general introduction that presents the context of this work, namely machine learning on graphs and network data analysis. This chapter also depicts the structure of the thesis and the list of publications produced during the thesis.
- Chapter 2 presents some aspects of graph theory and network data analysis by exposing some basic concepts and matrices relevant for this thesis.
- Chapter 3 introduces the bag-of-paths framework and the three models that compose it, namely the bag-of-paths, the bag-of-hitting-paths, and the randomized shortest path models, focusing on their mathematical derivations. This chapter also presents some of the different dissimilarity measures defined in these models.

- Chapter 4 extends the randomized shortest path model considering the addition of capacity constraints into the model. The contributions of this chapter are based on the development of two new algorithms in the randomized shortest path model. The first one allows to constrain flow capacity on edges while the second one constrains net flow capacity on edges. We also show how these algorithms can be used to solve maximum flow problems.
- Chapter 5 investigates a flow-constrained randomized shortest path formulation to the relative entropy-regularized optimal transport problem on a graph with multiple inputs and outputs having fixed marginal probabilities, previously studied in [120, 123].

In this context, two new algorithms are developed. The first algorithm solves the relative entropy regularized optimal transport problem in terms of minimum expected cost flow on a graph considering edge flow constraints. The second algorithm extends the first one by incorporating flow capacity constraints on edges, based on Chapter 4, in addition to input and output flow constraints. The applicability of these two algorithms to solve transportation problems is also investigated.

Moreover, the margin-constrained bag-of-paths surprisal distance between nodes of a graph [123] is also derived in this new randomized shortest path formalism.

- Chapter 6 derives two extensions of the bag-of-paths model including a Poisson probability distribution on the path lengths and on the probability of sampling a path. In these extended models, two new distance measures between nodes of a graph are defined: the Poisson weighted surprisal distance and the Poisson surprisal distance. Furthermore, a new interpretation of the logarithmic communicability similarity measure [134] is provided.
- Chapter 7 presents a comprehensive comparison of 43 similarity and dissimilarity measures between nodes through graph-based semi-supervised classification tasks. Among these measures, the margin-constrained bag-of-paths and bag-of-hitting-paths surprisal distances developed in [123], and further studied in Chapter 5, as well as the two new distance measures between nodes of a graph introduced in Chapter 6 are evaluated for the first time on semi-supervised classification problems.

The performance of all the investigated methods are compared when using partial information about the graph structure, contained in only a few dimensions (low dimensional space 5%, 10% and 20%), and with the

full information contained in the kernel matrix. The impact of the feature extraction technique (5%, 10% and 20% and kernel) on the classification results for the best-ranked investigated methods is also analyzed. In addition, an investigation of the best performing technique among all the best-ranked methods and the feature extraction techniques is performed.

The experimental comparison shows that the Poisson weighted surprisal distance with L1-normalized degree priors introduced in Chapter 6 stands out the most from the other investigated methods, especially when using all the information contained in the kernel. We can also point out that the margin-constrained bag-of-hitting-paths surprisal distance [123], the free energy distance [105], and the Tsallis free energy distance [176] obtain good performance especially when using 5% of dominant eigenvectors and all the kernel, as well as the sigmoid corrected commute time similarity [298, 299] across all the feature sets.

Furthermore, the comparison of the impact of the four extracted feature sets shows that the results obtained by extracting 5% of dominant eigenvectors are not significantly different from those obtained by using all the information contained in the kernel for most of the best-ranked methods.

- Chapter 8 focuses on automatically detecting fraudulent e-commerce transactions using network-related features and the free energy distance [156] developed in the bag-of-hitting-paths model. To deal with this specific application, a new way of computing the free energy distance, scaling on large, sparse, graphs [154] is investigated. Then, a fraud detection system called *FraudsFree* is created by incorporating this new method of computing the free energy distance into an existing algorithm named *APATE* [281].

Moreover, an experimental comparison between *FraudsFree* and others fraud detection systems is performed on a large real-world e-commerce credit card transaction dataset composed of 25,445,744 transactions obtained from Worldline SA/NV. The experimental comparison shows that the *FraudsFree* is competitive with the best method previously developed in Lebichot et al. [170] in terms of $\text{Pr}@100$ score on fraudulent card prediction with a significant speed-up in computation when the length of the set of paths is bounded to five.

Chapter	Contribution
4	• Introduction of both raw and net flow capacity constraints in the randomized shortest path model • Application to maximum flow problems
5	• Introduction of input and output flow constraints in the randomized shortest path model • Derivation of the margin-constrained bag-of-paths surprisal distance [123] • Application to transportation problems
6	• Introduction of two ways of taking into account the length of the path in the computation of its probability in the bag-of-paths model • Definition of two new distance measures between nodes of a graph • New interpretation of the logarithmic communicability similarity [134]
7	• Comparison of 43 similarity and dissimilarity measures on graph-based semi-supervised classification problems • Comparison of the performance of all the investigated methods with partial (5%, 10% and 20%) and full information about the graph structure • Analysis of the impact of the feature extraction technique (5%, 10%, 20% and kernel) on the classification results • Investigation of the best performing technique among all the methods and the feature extraction techniques
8	• Investigation of a new way to compute the free energy distance scaling on large, sparse, graphs [154] • Development of the fraud detection system FraudsFree • Comparison of FraudsFree and other fraud detection systems on a large real-world e-commerce credit card transaction dataset

Table 9.1: Summary of the contributions of the thesis.

9.2 Limitations

All research is subject to certain limitations and this thesis is no exception to the rule. Therefore, we present in this section a non-exhaustive list of the limitations in our work for each chapter containing original contributions.

An overall limitation of this thesis is that the models developed in it only apply to graph-based data. Even if graph-based data are very common (see Chapter 1 and [100, 103]), there are many other types of data such as images, spreadsheets, videos, texts, etc. Therefore, this limitation reduces the scope of our methods to problems where the data can be represented as a graph.

Moreover, the main drawback of the methods and the algorithms presented in Chapters 4, 5, and 6, contrary to the method of Chapter 8, is their high complexities of at least $\mathcal{O}(n^3)$ making them computationally intractable for large graphs.

This limitation explains why we have not been able to test the methods developed in Chapters 4, 5, and 6 on larger graphs, which is also a limitation of these chapters. Nevertheless, as we have shown in Chapter 8, solutions exist to make the methods introduced in the bag-of-paths framework usable on graphs containing millions of nodes.

A solution to improve the scalability of the algorithms presented in the Chapter 4 and Section 5.4 would be to use more sophisticated optimization techniques, going beyond the simple gradient technique used in Section 4.2, e.g., the sequential minimal optimization technique proposed by Platt [224]. Another avenue for improvement would be to explore the introduction of capacity constraints on a directed acyclic graph, as already discussed in Subsection 4.3.2 and experimented in [280]. Regarding the other methods in Chapters 5 and 6, we did not find convincing solutions to improve their scalability at this stage.

Another limitation of Chapter 5 is that the current flow-constrained randomized shortest path model only allows us to reformulate one of the two models developed in [123], i.e. the margin-constrained bag-of-paths model and not the margin-constrained bag-of-hitting-paths model. The reformulation of the margin-constrained bag-of-hitting-paths model would allow us to define the margin-constrained bag-of-hitting-paths surprisal distance [123] in our model.

Regarding Chapter 6, another drawback is that the Poisson distributions used in the two introduced models impose that the mean path length is equal to the variance path length. One solution to overcome this constraint would be to investigate generalized Poisson distributions [62].

Still another limitation of Chapter 6 is that the first proposed model assumes independence between path length and path likelihood, which is not the case in general.

In addition, a final limitation of Chapter 6 is that we only develop equivalents of the surprisal distance [105] and not of the randomized shortest path dissimilarity [156] or even of the free energy distance [105] (see Section 3.4) within the two new formulations of the bag-of-paths model.

As the previous three chapters, Chapter 7 also has its own set of limitations. First, the number of datasets used in our experiment is relatively small compared to the number of methods studied and many of these datasets are social networks or contain communities. Furthermore, all these datasets are small: the largest one contains only 1126 nodes. Additional datasets of different sizes should be added to the dataset collection using, for example, the Lancichinetti-Fortu-nato-Radicchi (LFR) benchmark [165], even though not all the methods studied will scale and provide results. These additions would more clearly differentiate the limitations and performance of the different methods. Following the same reasoning, we could also increase the number of methods, the range of parameter values, and the number of feature sets studied to obtain more accurate results.

Second, only the support vector machine classifier is used through our experiments. Therefore, we do not know if our findings are transposable to other classifiers or if they are specific to the support vector machine. It would

be interesting to investigate other classifiers, such as graph neural networks [182, 186], and to compare the performance of each classifier on all the feature sets.

One last limitation of Chapter 7 is that we have not explored why certain methods perform better on some graphs and what this says about those graphs. For this purpose, we could, as already proposed in [133] for graph clustering, collect and generate with the LFR benchmark [165] a large number of graphs with different structures and properties. Then, we could study which method works best, or worst, depending on these properties.

Finally, the main limitation of Chapter 8 is that we do not know if our approach is complementary, or just redundant, to the existing fraud defense lines of Worldline SA/NV. Another one is that we did not exploit all the potential of our method, we could for example try to modify the cost matrix to deal with hub nodes, as it has shown good results in other fraud detection studies [170].

Still another limitation of Chapter 8 is that we did not compare the performance of our graph-based fraud detection system, *FraudsFree*, with feature engineering approaches [17, 70]. Furthermore, we did not attempt to combine *FraudsFree* with other graph-based fraud detection systems [170, 281] or with feature engineering approaches [17, 70].

9.3 General discussion

As stated in the introduction of this chapter, this thesis is part of a will to continue the development of the bag-of-paths framework. On the one hand, several theoretical improvements are suggested to broaden the applicability of the framework. More specifically, we have proposed to incorporate capacity constraints on edges (Chapter 4), marginal constraints on input and output flows (Chapter 5), and a Poisson distribution weighting and constraining path lengths (Chapter 6) into the framework. On the other hand, several practical applications are also proposed to demonstrate its applicability to different problems. Indeed, we have presented a graph-based semi-supervised classification task (Chapter 7) and a real-world fraud detection case (Chapter 8).

However, many other applications of the bag-of-paths framework are still possible, the main condition being that the problem data can be represented as a graph. For example, this framework could be used to model diffusion processes such as the spread of viruses or diseases on virus networks, or the spread of rumors or other information on social networks. Another example, following this idea, would be to use the bag-of-paths framework to do viral marketing, i.e. find people to contact to spread information about a product or service on social networks as quickly as possible.

As previously mentioned, the main advantage of the bag-of-paths framework is that it allows to extrapolate between an optimal exploitation of the graph structure (optimal behavior) and a random exploration of the graph (random behavior) via a parameter that controls the desired degree of randomness. More precisely, this framework interpolates between two traditional paradigms that are the shortest paths and the random walks frameworks.

Nevertheless, as we pointed out in the previous section, this advantage is also accompanied by a major drawback. Namely, most of the methods defined in this framework have a high computational cost that can lead to intractability in some cases. It is therefore essential to use these methods only in situations where such interpolation increases performance or is of particular interest. In other situations, the shortest path and random walk methods usually lead to better results while having a lower computational cost.

However, the complexity problem can sometimes be overcome as shown in the fraud detection application of Chapter 8. We are convinced that other solutions of this kind can be found in future research to allow the application of the bag-of-paths framework to a larger class of problems and especially to real-world problems.

Randomized shortest path model: a flow approach

A

In this appendix, we show that the randomized shortest path model introduced in [156, 238, 300] is equivalent to the flow approach developed in [23, 121]. It therefore shows that the RSP can be derived from different point of views.

In their approach, the authors of [23, 121] propose to minimize the free energy over the set of non-negative flows contained in the matrix $\mathbf{N} = (n_{ij})$, where the element n_{ij} counts the number of transition from i to j along all the hitting paths from a source node s to an absorbing target node t , subject to the unit flow conservation constraint,

$$\left\{ \begin{array}{ll} \underset{\mathbf{N}}{\text{minimize}} & \phi_{st}(\mathbf{N}) = \frac{1}{n_{\bullet\bullet}} \sum_{(i,j) \in \mathcal{E}} \left[c_{ij} \varphi(n_{ij}) + T n_{ij} \log \frac{n_{ij}}{n_{i\bullet} p_{ij}} \right] \\ \text{subject to} & n_{\bullet i} + \delta_{it} = n_{i\bullet} + \delta_{is} \quad \text{for each } i \\ & n_{ij} \geq 0 \quad \text{for each edge } (i, j) \in \mathcal{E} \end{array} \right. \quad (\text{A.1})$$

where c_{ij} and p_{ij} are respectively the elements of the cost matrix $\mathbf{C}$ and the reference transition probability matrix $\mathbf{P}$ (see Subsections 2.2.2 and 2.2.4), δ_{it} is the Kronecker delta, and $\varphi(n_{ij})$ is a smooth non-decreasing function. Recall also that the bullet $\bullet$ denotes summation on the corresponding index. Note that the constraint $n_{ij} \geq 0$ is ensured by the Kullback-Leibler divergence present in the second term of Eq. A.1.

The problem presented in Eq. A.1 can be solved by minimizing the following Lagrange function integrating unit flow conservation constraints

$$\mathcal{L}(\mathbf{N}, \boldsymbol{\lambda}) = \phi_{st}(\mathbf{N}) + \sum_{i \in \mathcal{V}} \lambda_i (n_{\bullet i} + \delta_{it} - n_{i\bullet} - \delta_{is}) \quad (\text{A.2})$$

over the flow n_{kl} . Setting its partial derivative with respect to n_{kl} to zero provides

$$\frac{\partial \mathcal{L}(\mathbf{N}, \boldsymbol{\lambda})}{\partial n_{kl}} = \frac{\partial \phi_{st}(\mathbf{N})}{\partial n_{kl}} + \frac{\partial \sum_{i \in \mathcal{V}} \lambda_i (n_{\bullet i} + \delta_{it} - n_{i\bullet} - \delta_{is})}{\partial n_{kl}} = 0 \quad (\text{A.3})$$

For simplicity, we derive the two terms of this equation separately. The deriva-

tion of the first term provides

$$\begin{aligned}
\frac{\partial \phi_{st}(\mathbf{N})}{\partial n_{kl}} &= c_{kl} \varphi'(n_{kl}) + T \log \frac{n_{kl}}{n_{k\bullet} p_{kl}} + T - T \sum_{(i,j) \in \mathcal{E}} n_{ij} \frac{\partial \log n_{i\bullet}}{\partial n_{kl}} \\
&= c_{kl} \varphi'(n_{kl}) + T \log \frac{n_{kl}}{n_{k\bullet} p_{kl}} + T - T \sum_{(i,j) \in \mathcal{E}} n_{ij} \frac{1}{n_{i\bullet}} \frac{\partial n_{i\bullet}}{\partial n_{kl}} \\
&= c_{kl} \varphi'(n_{kl}) + T \log \frac{n_{kl}}{n_{k\bullet} p_{kl}} + T - T \sum_{(i,j) \in \mathcal{E}} \frac{n_{ij}}{n_{i\bullet}} \delta_{ik} \\
&= c_{kl} \varphi'(n_{kl}) + T \log \frac{n_{kl}}{n_{k\bullet} p_{kl}} + T - T \sum_{j \in \text{Succ}(k)} \frac{n_{kj}}{n_{k\bullet}} \\
&= c_{kl} \varphi'(n_{kl}) + T \log \frac{n_{kl}}{n_{k\bullet} p_{kl}}
\end{aligned} \tag{A.4}$$

where we used

$$\frac{\partial n_{i\bullet}}{\partial n_{kl}} = \frac{\partial (\sum_{j=1}^n n_{ij})}{\partial n_{kl}} = \sum_{j=1}^n \delta_{ik} \delta_{jl} = \delta_{ik} \tag{A.5}$$

The derivation of the second term of Eq. A.3 gives

$$\begin{aligned}
\frac{\partial \sum_{i \in \mathcal{V}} \lambda_i (n_{\bullet i} + \delta_{it} - n_{i\bullet} - \delta_{is})}{\partial n_{kl}} &= \frac{\sum_{i \in \mathcal{V}} \lambda_i (\partial n_{\bullet i} + \partial \delta_{it} - \partial n_{i\bullet} - \partial \delta_{is})}{\partial n_{kl}} \\
&= \sum_{i \in \mathcal{V}} \lambda_i (\delta_{il} - \delta_{ik}) \\
&= \lambda_l - \lambda_k
\end{aligned} \tag{A.6}$$

where we used

$$\frac{\partial n_{\bullet i}}{\partial n_{kl}} = \frac{\partial (\sum_{j=1}^n n_{ji})}{\partial n_{kl}} = \sum_{j=1}^n \delta_{jk} \delta_{il} = \delta_{il} \tag{A.7}$$

Therefore Eq. A.3 can be rewritten from Eqs. A.4 and A.6 as

$$\frac{\partial \mathcal{L}(\mathbf{N}, \boldsymbol{\lambda})}{\partial n_{kl}} = c_{kl} \varphi'(n_{kl}) + T \log \frac{n_{kl}}{n_{k\bullet} p_{kl}} - \lambda_k + \lambda_l = 0 \tag{A.8}$$

Isolating the logarithm and defining $\theta = 1/T$ provides

$$\log \frac{n_{kl}}{n_{k\bullet} p_{kl}} = -\theta (c_{kl} \varphi'(n_{kl}) - \lambda_k + \lambda_l) \tag{A.9}$$

Taking the exponential and isolating $\frac{n_{kl}}{n_{k\bullet}}$, we immediately obtain

$$\frac{n_{kl}}{n_{k\bullet}} = p_{kl} \exp[-\theta (c_{kl} \varphi'(n_{kl}) - \lambda_k + \lambda_l)]$$

$$= p_{kl} \exp[-\theta c_{kl} \varphi'(n_{kl})] \exp[\theta \lambda_k] \exp[-\theta \lambda_l] \quad (\text{A.10})$$

which is the optimality condition presented in Eq. 11 of [23].

Now let us assume that we consider st -shortest paths problems [23] which means that $\varphi(n_{kl}) = n_{kl}$, and therefore $\varphi'(n_{kl}) = 1$. The Eq. A.10 becomes

$$p_{kl}^* = \frac{n_{kl}}{n_{k\bullet}} = p_{kl} \exp[-\theta c_{kl}] \exp[\theta \lambda_k] \exp[-\theta \lambda_l] = w_{kl}^{(t)} \exp[\theta \lambda_k] \exp[-\theta \lambda_l] \quad (\text{A.11})$$

where $w_{kl}^{(t)}$ is the element k, l of the auxiliary matrix¹ $\mathbf{W}^{(t)}$ (see Eq. 3.5).

Let us pose that $\lambda_l = -\frac{1}{\theta} \log y_l$ and similarly $\lambda_k = -\frac{1}{\theta} \log y_k$, according to Eq. 15 of [23], then the Eq. A.11 can be rewritten as

$$p_{kl}^* = \frac{y_l}{y_k} w_{kl}^{(t)} \quad \text{for } (k, l) \in \mathcal{E} \text{ and for } k \neq t \quad (\text{A.12})$$

Furthermore, if we inject Eq. A.10 in Eq. A.1, we find

$$\phi_{st}(\mathbf{N}) = \sum_{(i,j) \in \mathcal{E}} n_{ij} (\lambda_i - \lambda_j) \quad (\text{A.13})$$

From this last equation, we can observe that the λ_i are undefined up to a translation. Thus, we can pose that $\lambda_t = 0$ and therefore $y_t = 1$.

Now let us sum Eq. A.12 over l to obtain

$$\sum_{l \in \mathcal{V}} y_k p_{kl}^* = y_k = \sum_{l \in \mathcal{V}} w_{kl}^{(t)} y_l \quad \text{for } k \neq t \quad (\text{A.14})$$

Thus, in matrix form, we find

$$\mathbf{y} = \mathbf{W}^{(t)} \mathbf{y} + \mathbf{e}_t \quad (\text{A.15})$$

where $\mathbf{e}_t$ is the t th column of the fundamental matrix $\mathbf{I}$. Isolating $\mathbf{e}_t$ provides

$$(\mathbf{I} - \mathbf{W}^{(t)}) \mathbf{y} = \mathbf{e}_t \quad (\text{A.16})$$

Finally, we can define $\mathbf{y}$ as

$$\begin{aligned} \mathbf{y} &= (\mathbf{I} - \mathbf{W}^{(t)})^{-1} \mathbf{e}_t \\ &= \mathbf{Z}_h^{(t)} \mathbf{e}_t \end{aligned} \quad (\text{A.17})$$

which is nothing else than the backward variable $\mathbf{z}_t^{h'(t)}$ defined in Eq. 3.27 in the RSP model.

¹Recall that the target node t is made killing and absorbing by setting the corresponding row of the reference transition matrix $\mathbf{P}$ (see Subsection 2.2.4) to zero, which implies that row t of $\mathbf{W}^{(t)}$ is also zero.

Therefore, Eq. A.12 can be rewritten as

$$p_{kl}^* = \frac{z_{lt}^{h(t)}}{z_{kt}^{h(t)}} w_{kl}^{(t)} \quad \text{for } (k, l) \in \mathcal{E} \text{ and for } k \neq t \quad (\text{A.18})$$

This quantity thus corresponds to the **optimal transition probabilities** of following any edge $(i, j) \in \mathcal{E}$ induced by the set of paths $\mathcal{P}_{st}^h$ and their probability mass (Eq. 3.21) in the randomized shortest path model (see Eq. 4.4 and [238]).

This establishes a link between Bavaud and Guex's formulation [23, 121] and the randomized shortest path model [156, 238, 300].

Bibliography

- [1] A. Abdallah, M. A. Maarof, and A. Zainal. “Fraud detection system: a survey”. In: *Journal of Network and Computer Applications* 68 (2016), pp. 90–113.
- [2] G. Agnarsson and R. Greenlaw. *Graph theory: modeling, applications, and algorithms*. 1st ed. Prentice Hall, 2007.
- [3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. 1st ed. Prentice Hall, 1993.
- [4] T. Akamatsu. “Cyclic flows, Markov process and stochastic traffic assignment”. In: *Transportation Research Part B: Methodological* 30.5 (1996), pp. 369–386.
- [5] M. Alamgir and U. von Luxburg. “Phase transition in the family of p-resistances”. In: *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS’11)*. MIT Press, 2011, pp. 379–387.
- [6] A. Albert. *Regression and the Moore-Penrose pseudoinverse*. 1st ed. Academic Press, 1972.
- [7] E. Alpaydin. *Introduction to Machine Learning*. 4th ed. MIT Press, 2020.
- [8] B. Amaliah, C. Fatichah, and E. Suryani. “Total opportunity cost matrix–Minimal total: a new approach to determine initial basic feasible solution of a transportation problem”. In: *Egyptian Informatics Journal* 20.2 (2019), pp. 131–141.
- [9] H. Anton and R. C. Busby. *Contemporary linear algebra*. 1st ed. Wiley, 2003.
- [10] F. Arqué, C. A. Uribe, and C. Ocampo-Martinez. “Approximate Wasserstein attraction flows for dynamic mass transport over networks”. In: *Automatica* 143 (2022), pp. 1–17.
- [11] K. Arrow, L. Hurwicz, and H. Uzawa. *Studies in linear and non-linear programming*. 1st ed. Stanford University Press, 1958.
- [12] H. Arsham and A. B. Kahn. “A simplex-type algorithm for general transportation problems: an alternative to stepping-stone”. In: *Journal of the Operational Research Society* 40.6 (1989), pp. 581–590.

- [13] J. Atwood and D. Towsley. “Diffusion-convolutional neural networks”. In: *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS’16)*. MIT Press, 2016, pp. 2001–2009.
- [14] A. Azran. “The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks”. In: *Proceedings of the 24th International Conference on Machine Learning (ICML’07)*. 2007, pp. 49–56.
- [15] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna. “Four Degrees of Separation”. In: *Proceedings of the 4th Annual ACM Web Science Conference (WebSci’12)*. ACM. Association for Computing Machinery, 2012, pp. 33–42.
- [16] B. Baesens, V. Van Vlasselaer, and W. Verbeke. *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection*. 1st ed. John Wiley & Sons, 2015.
- [17] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten. “Feature engineering strategies for credit card fraud detection”. In: *Expert Systems with Applications* 51 (2016), pp. 134–142.
- [18] A. C. Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten. “Cost sensitive credit card fraud detection using Bayes minimum risk”. In: *Proceedings of the 12th IEEE International Conference on Machine Learning and Applications (ICMLA’13)*. IEEE. 2013, pp. 333–338.
- [19] N. Balakrishnan. “Modified Vogel’s approximation method for the unbalanced transportation problem”. In: *Applied Mathematics Letters* 3.2 (1990), pp. 9–11.
- [20] R. B. Bapat. *Graphs and matrices*. 1st ed. Springer, 2010.
- [21] A.-L. Barabási. *Network science*. 1st ed. Cambridge University Press, 2016.
- [22] F. C. A. Barata and M. S. Hussein. “The Moore-Penrose Pseudoinverse. A tutorial review of the theory”. In: *Brazilian Journal of Physics* 42 (2012), pp. 146–165.
- [23] F. Bavaud and G. Guex. “Interpolating between random walks and shortest paths: a path functional approach”. In: *Proceedings of the 4th International Conference on Social Informatics (SocInfo’12)*. Springer, 2012, pp. 68–81.
- [24] M. Belkin, P. Niyogi, and V. Sindhwani. “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples”. In: *Journal of machine learning research* 7.11 (2006), pp. 2399–2434.

- [25] M. G. H. Bell. “Alternatives to Dial’s logit assignment algorithm”. In: *Transportation Research Part B: Methodological* 29.4 (1995), pp. 287–295.
- [26] D. P. Bertsekas. *Network optimization: continuous and discrete models*. 1st ed. Athena Scientific, 1998.
- [27] D. P. Bertsekas. *Nonlinear programming*. 2nd ed. Athena Scientific, 1999.
- [28] H. R. Bhapkar, P. N. Mahalle, and P. S. Dhotre. “Virus graph and COVID-19 Pandemic: a graph theory approach”. In: *Big Data Analytics and Artificial Intelligence Against COVID-19: Innovation Vision and Approach*. Springer International Publishing, 2020, pp. 15–34.
- [29] V. Bhusari and S. Patil. “Study of hidden markov model in credit card fraudulent detection”. In: *International Journal of Computer Applications* 20.5 (2011), pp. 33–36.
- [30] C. M. Bishop. *Pattern recognition and machine learning*. 1st ed. Springer, 2006.
- [31] Blackened. *Euler’s Solution of Seven Bridges of Königsberg in Layman Terms*. 2015. URL: <https://math.stackexchange.com/q/1173328> (visited on January 26, 2021).
- [32] B. Bollobas. *Modern graph theory*. 2nd ed. Springer, 1998.
- [33] R. J. Bolton and D. J. Hand. “Statistical fraud detection: a review”. In: *Statistical science* (2002), pp. 235–249.
- [34] R. J. Bolton and D. J. Hand. “Unsupervised profiling methods for fraud detection”. In: *Credit scoring and credit control VII* (2001), pp. 235–255.
- [35] I. Borg and P. Groenen. *Modern multidimensional scaling: theory and applications*. 1st ed. Springer, 1997.
- [36] S. Boyd and L. Vandenberghe. *Convex optimization*. 1st ed. Cambridge University Press, 2004.
- [37] Y. Boykov and V. Kolmogorov. “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.9 (2004), pp. 1124–1137.
- [38] Bpost. *Bpost devient plus que jamais un acteur incontournable de la data*. 2022. URL: <https://www.bpost.be/fr/blog/bpost-media/bpost-devient-plus-que-jamais-un-acteur-incontournable-de-la-data> (visited on March 14, 2022).

- [39] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. “On modularity clustering”. In: *IEEE Transactions on Knowledge and Data Engineering* 20 (2008), pp. 172–188.
- [40] U. Brandes and D. Fleischer. “Centrality measures based on current flow”. In: *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS’05)*. 2005, pp. 533–544.
- [41] S. Brin and L. Page. “The anatomy of a large-scale hypertextual Web search engine”. In: *Computer Networks and ISDN Systems* 30.1–7 (1998), pp. 107–117.
- [42] S. Calbert. “Study of semi-supervised classification algorithms on a graph, based on convolutional neural networks and kernels on graph”. MA thesis. Université catholique de Louvain, Belgium, 2020.
- [43] J. Callut, K. Françoisse, M. Saerens, and P. Dupont. “Semi-supervised classification from discriminative random walks”. In: *Proceedings of the 19th European Conference on Machine Learning (ECML’08)*. Springer, 2008, pp. 162–177.
- [44] T. Can and H. Koçak. “Tuncay Can’s approximation method to obtain initial basic feasible solution to transport problem”. In: *Applied and Computational Mathematics* 5.2 (2016), pp. 78–82.
- [45] B. Cao, M. Mao, S. Viidu, and P. Yu. “Collective fraud detection capturing inter-transaction dependency”. In: *on Anomaly Detection in Finance of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’17)*. ACM. 2018, pp. 66–75.
- [46] P. K. Chan, W. Fan, A. L. Prodromidis, and S. J. Stolfo. “Distributed data mining in credit card fraud detection”. In: *IEEE Intelligent systems* 14.6 (1999), pp. 67–74.
- [47] A. K. Chandra, P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari. “The electrical resistance of a graph captures its commute and cover times”. In: *Computational Complexity* 6.4 (1996), pp. 312–340.
- [48] C.-C. Chang and C.-J. Lin. “LIBSVM: a library for support vector machines”. In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011), 27:1–27:27.
- [49] A. Charnes, W. W. Cooper, and A. Henderson. *An introduction to linear programming*. John Wiley & Sons, 1953.
- [50] P. Chebotarev. “A class of graph-geodetic distances generalizing the shortest-path and the resistance distances”. In: *Discrete Applied Mathematics* 159.5 (2011), pp. 295–302.

- [51] P. Chebotarev. “Studying new classes of graph metrics”. In: *Proceedings of the 1st International Conference on Geometric Science of Information (GSI'13)*. Springer. 2013, pp. 207–214.
- [52] P. Chebotarev. “The walk distances in graphs”. In: *Discrete Applied Mathematics* 160.10–11 (2012), pp. 1484–1500.
- [53] P. Chebotarev and E. Shamis. “The matrix-forest theorem and measuring relations in small social groups”. In: *Automation and Remote Control* 58.9 (1997), pp. 1505–1514.
- [54] Y. Chen, T. T. Georgiou, and M. Pavon. “Stochastic control liaisons: Richard Sinkhorn meets Gaspard Monge on a Schrodinger bridge”. In: *SIAM Review* 63.2 (2021), pp. 249–313.
- [55] Y. Chen, T. T. Georgiou, M. Pavon, and A. Tannenbaum. “Efficient robust routing for single commodity network flows”. In: *IEEE Transactions on Automatic Control* 63.7 (2017), pp. 2287–2294.
- [56] Y. Chen, T. T. Georgiou, M. Pavon, and A. Tannenbaum. “Robust transport over networks”. In: *IEEE Transactions on Automatic Control* 62.9 (2017), pp. 4675–4682.
- [57] Y. Chong, Y. Ding, Q. Yan, and S. Pan. “Graph-based semi-supervised learning: A review”. In: *Neurocomputing* 408 (2020), pp. 216–230.
- [58] E. Christoph. “Exploratory analysis of network data based on the bag-of-paths framework: clustering & embedding”. MA thesis. Université catholique de Louvain, Belgium, 2022.
- [59] F. R. Chung. *Spectral graph theory*. 2nd ed. American Mathematical Society, 1997.
- [60] E. Cinlar. *Introduction to stochastic processes*. 1st ed. Courier Corporation, 2013.
- [61] A. Clauset, M. E. J. Newman, and C. Moore. “Finding community structure in very large networks”. In: *Physical Review E* 70.6 (2004), pp. 1–6.
- [62] P. C. Consul. *Generalized Poisson distributions: properties and applications*. 1st ed. Marcel Dekker, 1989.
- [63] S. Courtain, G. Guex, I. Kivimäki, and M. Saerens. “Relative entropy-regularized optimal transport on a graph: a new algorithm and an experimental comparison”. In: *Submitted for publication, currently under minor revision* (2021).
- [64] S. Courtain, P. Leleux, I. Kivimäki, G. Guex, and M. Saerens. “Randomized shortest paths with net flows and capacity constraints”. In: *Information Sciences* (2020), pp. 341–360.

- [65] S. Courtain, B. Lebichot, I. Kivimäki, and M. Saerens. “Graph-based fraud detection with the free energy distance”. In: *Proceedings of the 8th International Conference on Complex Networks and their Applications (CNA’19)*. Springer, 2019, pp. 40–52.
- [66] S. Courtain and M. Saerens. “A simple extension of the bag-of-paths model weighting path lengths by a Poisson distribution”. In: *Proceedings of the 10th International Conference on Complex Networks and their Applications (CNA’21)*. Springer, 2022, pp. 220–233.
- [67] T. M. Cover and J. A. Thomas. *Elements of information theory*. 2nd ed. Wiley, 2006.
- [68] J.-C. Culioli. *Introduction à l’optimisation*. 2nd ed. Ellipses, 2012.
- [69] M. Cuturi. “Sinkhorn distances: lightspeed computation of optimal transport”. In: *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS’13)*. MIT Press, 2013, pp. 2292–2300.
- [70] A. Dal Pozzolo. “Adaptive machine learning for credit card fraud detection”. PhD thesis. Université Libre de Bruxelles, Belgium, 2015.
- [71] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi. “Credit card fraud detection and concept-drift adaptation with delayed supervised information”. In: *Proceedings of the 15th International Joint Conference on Neural networks (IJCNN’15)*. IEEE. 2015, pp. 1–8.
- [72] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi. “Credit card fraud detection: a realistic modeling and a novel learning strategy”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.8 (2018), pp. 3784–3797.
- [73] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi. “Learned lessons in credit card fraud detection from a practitioner perspective”. In: *Expert systems with applications* 41.10 (2014), pp. 4915–4928.
- [74] G. B. Dantzig. “Application of the simplex method to a transportation problem”. In: *Activity analysis and production and allocation* (1951).
- [75] G. B. Dantzig. *Linear programming and extensions*. 1st ed. Princeton University Press, 1963.
- [76] D. Davidson. “Paradoxes of irrationality”. In: *Philosophical Essays on Freud*. Cambridge University Press, 1982, pp. 289–305.
- [77] F.-P. De Plaen. “Study of convolutional models for semi-supervised classification on graph-structured data”. MA thesis. Université catholique de Louvain, Belgium, 2021.

- [78] J.-C. Delvenne and A.-S. Libert. “Centrality measures and thermodynamic formalism for complex networks”. In: *Physical Review E* 83.4 (2011), pp. 1–7.
- [79] J. Demšar. “Statistical comparisons of classifiers over multiple data sets”. In: *Journal of Machine learning research* 7 (2006), pp. 1–30.
- [80] R. Devooght, A. Mantrach, I. Kivimäki, H. Bersini, A. Jaimes, and M. Saerens. “Random walks based modularity: application to semi-supervised learning”. In: *Proceedings of the 23rd International World Wide Web Conference (WWW’14)*. 2014, pp. 213–224.
- [81] M. M. Deza and E. Deza. *Encyclopedia of distances*. 1st ed. Springer, 2009, pp. 1–583.
- [82] R. Dial. “A probabilistic multipath assignment model that obviates path enumeration”. In: *Transportation Research* 5 (1971), pp. 83–111.
- [83] O. Díaz-Parra, J. A. R.-V. B. Bernábe Loranca, A. Fuentes-Penna, and R. A. Barrera-Cámara. “A survey of transportation problems”. In: *Journal of Applied Mathematics* (2014), pp. 1–17.
- [84] T. Distant and T. Dist. “Method for solving unbalanced transportation problems using standard deviations”. In: *International Journal of Pure and Applied Mathematics* 119.16 (2018), pp. 4971–4989.
- [85] A. Dolan and J. Aldous. *Networks and algorithms: an introductory approach*. 1st ed. Wiley, 1993.
- [86] P. G. Doyle and J. L. Snell. *Random walks and electric networks*. 1st ed. The Mathematical Association of America, 1984.
- [87] D. Dua and C. Graff. *UCI machine learning repository*. 2017.
- [88] E. Duman and I. Elikucuk. “Solving credit card fraud detection problem by the new metaheuristics migrating birds optimization”. In: *Proceedings of the 12th International Work-Conference on Artificial Neural Networks (IWANN’13)*. Springer. 2013, pp. 62–71.
- [89] J. Edmonds and R. M. Karp. “Theoretical improvements in algorithmic efficiency for network flow problems”. In: *Journal of the ACM (JACM)* 19.2 (1972), pp. 248–264.
- [90] M. Engquist. “A successive shortest path algorithm for the assignment problem”. In: *INFOR: Information Systems and Operational Research* 20.4 (1982), pp. 370–384.
- [91] S. Erlander and N. Stewart. *The gravity model in transportation analysis. Theory and extensions*. 1st ed. VSP International Science Publishers, 1990.

- [92] E. Estrada and N. Hatano. “Communicability in complex networks”. In: *Physical Review E* 77.3 (2008), pp. 1–12.
- [93] L. Euler. “Solutio problematis ad geometriam situs pertinentis”. In: *Commentarii academiae scientiarum Petropolitanae* (1741), pp. 128–140.
- [94] P. Ferrari. “Road pricing and network equilibrium”. In: *Transportation Research Part B: Methodological* 29.5 (1995), pp. 357–372.
- [95] M. T. Fiala Timlin and W. R. Pulleyblank. “Precedence constrained routing and helicopter scheduling: heuristic design”. In: *Interfaces* 22.3 (1992), pp. 100–111.
- [96] M. Flood. “Application of transportation theory to scheduling a military tanker fleet”. In: *Journal of the Operations Research Society of America* 2.2 (1954), pp. 150–162.
- [97] R. W. Floyd. “Algorithm 97: shortest path”. In: *Communications of the ACM* 5.6 (1962), p. 344.
- [98] L. R. Ford and D. R. Fulkerson. “A primal-dual algorithm for the capacitated Hitchcock problem”. In: *Naval Research Logistics Quarterly* 4.1 (1957), pp. 47–54.
- [99] L. R. Ford and D. R. Fulkerson. *Flows in networks*. 1st ed. RAND Corporation, 1962.
- [100] L. R. Foulds. *Graph theory applications*. 1st ed. Springer, 1992.
- [101] F. Fouss, K. Francoisse, L. Yen, A. Pirotte, and M. Saelens. “An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification”. In: *Neural Networks* 31 (2012), pp. 53–72.
- [102] F. Fouss, A. Pirotte, J. M. Renders, and M. Saelens. “Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation”. In: *IEEE Transactions on Knowledge and Data Engineering* 19.3 (2007), pp. 355–369.
- [103] F. Fouss, M. Saelens, and M. Shimbo. *Algorithms and models for network data and link analysis*. 1st ed. Cambridge University Press, 2016.
- [104] F. Fouss, L. Yen, A. Pirotte, and M. Saelens. “An experimental investigation of graph kernels on a collaborative recommendation task”. In: *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM’06)*. IEEE. 2006, pp. 863–868.
- [105] K. Françoise, I. Kivimäki, A. Mantrach, F. Rossi, and M. Saelens. “A bag-of-paths framework for network data analysis”. In: *Neural Networks* 90 (2017), pp. 90–111.

- [106] S. García-Díez, F. Fouss, M. Shimbo, and M. Saerens. “A sum-over-paths extension of edit distances accounting for all sequence alignments”. In: *Pattern Recognition* 44.6 (2011), pp. 1172–1182.
- [107] S. I. Gass. “On solving the transportation problem”. In: *Journal of the Operational Research Society* 41.4 (1990), pp. 291–297.
- [108] S. I. Gass. “The dualplex method applied to special linear program-mings”. In: *Proceedings of the 1st International Federation of Informa-tion Processing Societies*. North-Holland Publishing Company. 1972, pp. 1317–1323.
- [109] A. V. Goldberg and R. E. Tarjan. “A new approach to the maximum-flow problem”. In: *Journal of the ACM (JACM)* 35.4 (1988), pp. 921–940.
- [110] G. H. Golub and C. F. Van Loan. *Matrix Computations*. 4th ed. The Johns Hopkins University Press, 2013.
- [111] M. Gondran and M. Minoux. *Graphes et algorithmes*. 4th ed. Lavoisier, 2009.
- [112] C. Gong, T. Liu, D. Tao, K. Fu, E. Tu, and J. Yang. “Deformed graph Laplacian for semisupervised learning”. In: *IEEE Transactions on Neu-ral Networks and Learning Systems* 26.10 (2015), pp. 2261–2274.
- [113] M. Gori and A. Pucci. “ItemRank: a random-walk based scoring al-gorithm for recommender engines”. In: *Proceedings of the 20th In-ternational Joint Conference on Artificial Intelligence (IJCAI’07)*. 2007, pp. 2766–2771.
- [114] J. C. Gower and M. J. Warrens. “Similarity, dissimilarity, and distance, measures of”. In: *Wiley StatsRef: Statistics Reference Online* (2014), pp. 1–11.
- [115] S. K. Goyal. “Improving VAM for unbalanced transportation prob-lems”. In: *The Journal of the Operational Research Society* 35.12 (1984), pp. 1113–1114.
- [116] L. Grady and G. Funka-Lea. “Multi-label image segmentation for medical applications based on graph-theoretic electrical potentials”. In: *Proceedings of the 8th European Conference on Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis (ECCV’04)*. Springer, 2004, pp. 230–245.
- [117] M. Grant and S. Boyd. *CVX: Matlab software for disciplined convex pro-gramming, version 2.1*. 2014. URL: <http://cvxr.com/cvx> (visited on February 14, 2022).
- [118] C. M. Grinstead and J. L. Snell. *Introduction to probability*. 1st ed. American Mathematical Society, 1997.

- [119] I. Griva, S. Nash, and A. Sofer. *Linear and nonlinear optimization*. 2nd ed. SIAM, 2008.
- [120] G. Guex. “Interpolating between random walks and optimal transportation routes: Flow with multiple sources and targets”. In: *Physica A: Statistical Mechanics and its Applications* 450 (2016), pp. 264–277.
- [121] G. Guex and F. Bavaud. “Flow-based dissimilarities: shortest path, commute time, max-flow and free energy”. In: *Data Science, Learning by Latent Structures, and Knowledge Discovery*. Springer, 2015, pp. 101–111.
- [122] G. Guex, S. Courtain, and M. Saerens. “Covariance and correlation kernels on a graph in the generalized bag-of-paths formalism”. In: *Journal of Complex Networks* 8.6 (2021), pp. 1–46.
- [123] G. Guex, I. Kivimäki, and M. Saerens. “Randomized optimal transport on a graph: framework and new distance measures”. In: *Network Science* 7.1 (2019), pp. 88–122.
- [124] G. Hadley. *Linear programming*. 1st ed. Addison-Wesley Publishing Company, 1962.
- [125] T. Hashimoto, Y. Sun, and T. Jaakkola. “From random walks to distances on unweighted graphs”. In: *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS’15)*. MIT Press, 2015, pp. 3429–3437.
- [126] D. W. Hearn and S. Lawphongpanich. “A dual ascent algorithm for traffic assignment problems”. In: *Transportation Research Part B: Methodological* 24.6 (1990), pp. 423–430.
- [127] M. P. Helme. “Reducing air traffic delay in a space-time network”. In: *Proceedings of the 4th IEEE International Conference on Systems, Man, and Cybernetics (SMC’92)*. IEEE, 1992, pp. 236–242.
- [128] M. Herbster and G. Lever. “Predicting the labelling of a graph via minimum p-seminorm interpolation”. In: *Proceedings of the 22nd Conference on Learning Theory (COLT’09)*. 2009, pp. 18–21.
- [129] F. S. Hillier and G. J. Lieberman. *Introduction to operations research*. 9th ed. McGraw-Hill Education, 2010.
- [130] F. L. Hitchcock. “The distribution of a product from several sources to numerous localities”. In: *Journal of mathematics and physics* 20.1–4 (1941), pp. 224–230.
- [131] HSN Consultants. *The nilson report Issue 1142*. 2018. URL: <https://nilsonreport.com> (visited on August 13, 2019).

- [132] T. Ito, M. Shimbo, T. Kudo, and Y. Matsumoto. “Application of kernels to link analysis”. In: *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’05)*. ACM. 2005, pp. 586–592.
- [133] V. Ivashkin and P. Chebotarev. “Dissecting graph measures performance for node clustering in LFR parameter space”. In: *Proceedings of the 10th International Conference on Complex Networks and their Applications (CNA’21)*. Springer. 2022, pp. 328–341.
- [134] V. Ivashkin and P. Chebotarev. “Do logarithmic proximity measures outperform plain ones in graph clustering?” In: *Proceedings of the 6th International Conference on Network Analysis (NET’16)*. Springer. 2016, pp. 87–105.
- [135] M. James. “The generalised inverse”. In: *The Mathematical Gazette* 62.420 (1978), pp. 109–114.
- [136] E. T. Jaynes. “Information theory and statistical mechanics”. In: *Physical Review* 106 (1957), pp. 620–630.
- [137] T. Jebara. *Machine learning: discriminative and generative*. 1st ed. Springer Science & Business Media, 2012.
- [138] Z. Juman and M. A. Hoque. “An efficient heuristic to obtain a better initial feasible solution to the transportation problem”. In: *Applied Soft Computing* 34 (2015), pp. 813–826.
- [139] D. Jungnickel. *Graphs, networks, and algorithms*. 4th ed. Springer, 2013.
- [140] J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P.-E. Portier, L. He-Guelton, and O. Caelen. “Sequence classification for credit-card fraud detection”. In: *Expert Systems with Applications* 100 (2018), pp. 234–245.
- [141] Y. Kacher and P. Singh. “A comprehensive literature review on transportation problems”. In: *International Journal of Applied and Computational Mathematics* 7.5 (2021), pp. 1–49.
- [142] L. V. Kantorovich. “On the translocation of masses”. In: *Management science* 5.1 (1958), pp. 1–4.
- [143] J. N. Kapur. *Maximum-entropy models in science and engineering*. 1st ed. Wiley, 1989.
- [144] J. N. Kapur and H. K. Kesavan. *Entropy optimization principles with applications*. 1st ed. Academic Press, 1992.

- [145] K. Karagul and Y. Sahin. “A novel approximation method to obtain initial basic feasible solution of transportation problem”. In: *Journal of King Saud University-Engineering Sciences* 32.3 (2020), pp. 211–218.
- [146] H. S. Kasana and K. D. Kumar. *Introductory operations research: theory and applications*. 1st ed. Springer, 2004.
- [147] L. Katz. “A new status index derived from sociometric analysis”. In: *Psychometrika* 18.1 (1953), pp. 39–43.
- [148] L. Kaur, M. Rakshit, and S. Singh. “An alternate approach for finding the initial basic feasible solution of transportation problem”. In: *International Journal of Scientific & Technology Research* 8.9 (2019), pp. 442–447.
- [149] J. G. Kemeny and J. L. Snell. *Finite markov chains*. 1st ed. Springer, 1976.
- [150] L. C. Kendal and J. J. Buckley. *The business of shipping*. 7th ed. Cornell Maritime Press, 2001.
- [151] J. Kepner and J. Gilbert. *Graph algorithms in the language of linear algebra*. 1st ed. SIAM, 2011.
- [152] T. N. Kipf and M. Welling. “Semi-supervised classification with graph convolutional networks”. In: *Proceedings of the 5th International Conference on Learning Representations (ICLR’17)*. 2017.
- [153] Ö. Kirca and A. Şatir. “A heuristic for obtaining and initial solution for the transportation problem”. In: *Journal of the Operational Research Society* 41.9 (1990), pp. 865–871.
- [154] I. Kivimäki. “Distances, centralities and model estimation methods based on randomized shortest paths for network data analysis”. PhD thesis. Université catholique de Louvain, Belgium, 2018.
- [155] I. Kivimäki, B. Lebichot, J. Saramäki, and M. Suerens. “Two betweenness centrality measures based on randomized shortest paths”. In: *Scientific Reports* 6 (2016), pp. 1–15.
- [156] I. Kivimäki, M. Shimbo, and M. Suerens. “Developments in the theory of randomized shortest paths with a comparison of graph node distances”. In: *Physica A: Statistical Mechanics and its Applications* 393 (2014), pp. 600–616.
- [157] D. J. Klein and M. Randic. “Resistance distance”. In: *Journal of Mathematical Chemistry* 12 (1993), pp. 81–95.
- [158] M. Klein. “A primal method for minimal cost flows with applications to the assignment and transportation problems”. In: *Management Science* 14 (1966), pp. 205–220.

- [159] R. Kolisch and C. Heimerl. “An efficient metaheuristic for integrated scheduling and staffing IT projects based on a generalized minimum cost flow network”. In: *Naval Research Logistics* 59.2 (2012), pp. 111–127.
- [160] R. I. Kondor and J. Lafferty. “Diffusion kernels on graphs and other discrete structures”. In: *Proceedings of the 19th International Conference on Machine Learning (ICML '02)*. 2002, pp. 315–322.
- [161] D. König. *Theorie der endlichten und unendlichen graphen*. 1st ed. Akademische Verlagsgesellschaft, Leipzig, 1936.
- [162] T. C. Koopmans. “Optimum utilization of the transportation system”. In: *Econometrica: Journal of the Econometric Society* (1949), pp. 136–146.
- [163] B. Korte and J. Vygen. *Combinatorial optimization. Theory and algorithms*. 6th ed. Springer, 2018.
- [164] S. Korukoğlu and S. Balli. “An improved Vogel’s approximation method for the transportation problem”. In: *Mathematical and Computational Applications* 16.2 (2011), pp. 370–381.
- [165] A. Lancichinetti, S. Fortunato, and F. Radicchi. “Benchmark graphs for testing community detection algorithms”. In: *Physical Review E* 78.4 (2008), pp. 1–5.
- [166] K. Lange. *Optimization*. 2nd ed. Springer, 2013.
- [167] K. C. Laudon and C. G. Traver. *E-commerce: business, technology, society*. 12th ed. Prentice Hall, 2016.
- [168] G. F. Lawler. *Random walk and the heat equation*. 1st ed. Vol. 55. American Mathematical Society, 2010.
- [169] B. Lebichot. “Network analysis based on bag-of-paths: classification, node criticality and randomized policies”. PhD thesis. Université catholique de Louvain, Belgium, 2018.
- [170] B. Lebichot, F. Braun, O. Caelen, and M. Saerens. “A graph-based, semi-supervised, credit card fraud detection system”. In: *Proceedings of the 5th International Conference on Complex Networks and their Applications (CNA'16)*. Springer. 2016, pp. 721–733.
- [171] B. Lebichot, G. Guex, I. Kivimäki, and M. Saerens. “A constrained randomized shortest-paths framework for optimal exploration”. In: *ArXiv preprint paper arXiv:1807.04551* (2018).

- [172] B. Lebichot, I. Kivimäki, K. Françoisse, and M. Saerens. “Semi-supervised classification through the bag-of-paths group betweenness”. In: *IEEE Transactions on Neural Networks and Learning Systems* 25.6 (2013), pp. 1173–1186.
- [173] B. Lebichot and M. Saerens. “A bag-of-paths node criticality measure”. In: *Neurocomputing* 275 (2018), pp. 224–236.
- [174] E. A. Leicht, P. Holme, and M. E. J. Newman. “Vertex similarity in networks”. In: *Physical Review E* 73.2 (2006), pp. 1–10.
- [175] P. Leleux, S. Courtain, K. Françoisse, and M. Saerens. “Design of biased random walks on a graph with application to collaborative recommendation”. In: *Physica A: Statistical Mechanics and its Applications* 590 (2022), pp. 1–20.
- [176] P. Leleux, S. Courtain, G. Guex, and M. Saerens. “Sparse randomized shortest paths routing with Tsallis divergence regularization”. In: *Data Mining and Knowledge Discovery* (2021), pp. 986–1031.
- [177] V. Lempitsky and Y. Boykov. “Global optimization for shape fitting”. In: *Proceedings of the 12th IEEE International Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’07)*. IEEE. 2007, pp. 1–8.
- [178] Y. Li, Z.-L. Zhang, and D. Boley. “From shortest-path to all-path: the routing continuum theory and its applications”. In: *IEEE Transactions on Parallel and Distributed Systems* 25.7 (2013), pp. 1745–1755.
- [179] D. Liben-Nowell and J. Kleinberg. “The link-prediction problem for social networks”. In: *Journal of the American society for information science and technology* 58.7 (2007), pp. 1019–1031.
- [180] Q. Liu and Y. Wu. “Supervised learning”. In: *Encyclopedia of the Sciences of Learning* (2012), pp. 3243–3245.
- [181] Y. Liu and K. Kirchhoff. “Graph-based semi-supervised learning for phone and segment classification”. In: *Proceedings of the 14th International Conference on Speech Communication (INTERSPEECH’13)*. 2013, pp. 1840–1843.
- [182] Z. Liu and J. Zhou. *Introduction to graph neural networks*. 1st ed. Morgan & Claypool Publishers, 2020.
- [183] R. Lougee-Heimer. “The common optimization interface for operations research: Promoting open-source software in the operations research community”. In: *IBM Journal of Research and Development* 47.1 (2003), pp. 57–66.

- [184] L. Lü, C.-H. Jin, and T. Zhou. “Similarity index based on local paths for link prediction of complex networks”. In: *Physical Review E* 80.4 (2009), pp. 1–9.
- [185] L. Lü and T. Zhou. “Link prediction in complex networks: A survey”. In: *Physica A: Statistical Mechanics and its Applications* 390.6 (2011), pp. 1150–1170.
- [186] Y. Ma and J. Tang. *Deep learning on graphs*. 1st ed. Cambridge University Press, 2021.
- [187] S. A. Macskassy and F. Provost. “Classification in networked data: a toolkit and a univariate case study”. In: *Journal of Machine Learning Research* 8 (2007), pp. 935–983.
- [188] C. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. 1st ed. Cambridge University Press, 2008.
- [189] A. Mantrach, N. Van Zeebroeck, P. Francq, M. Shimbo, H. Bersini, and M. Saerens. “Semi-supervised classification and betweenness computation on large, sparse, directed graphs”. In: *Pattern recognition* 44.6 (2011), pp. 1212–1224.
- [190] A. Mantrach, L. Yen, J. Callut, K. Françoisse, M. Shimbo, and M. Saerens. “The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.6 (2009), pp. 1112–1126.
- [191] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate analysis*. 1st ed. Probability and mathematical statistics. Academic Press, 1979.
- [192] M. Mathirajan, S. Reddy, and M. Rani Vimala. “An experimental study of newly proposed initial basic feasible solution methods for a transportation problem”. In: *OPSEARCH* (2021), pp. 1–44.
- [193] C. D. Meyer. *Matrix analysis and applied linear algebra*. 1st ed. Vol. 71. SIAM, 2000.
- [194] M. Minoux. *Programmation mathématique*. 2nd ed. Lavoisier, 2008.
- [195] T. M. Mitchell. *Machine Learning*. 1st ed. McGraw-Hill, 1997.
- [196] I. Molloy, S. Chari, U. Finkler, M. Wiggerman, C. Jonker, T. Habeck, Y. Park, F. Jordens, and R. van Schaik. “Graph analytics for real-time scoring of cross-channel transactional fraud”. In: *Proceedings of the 20th International Conference on Financial Cryptography and Data Security (FC’16)*. Springer. 2016, pp. 22–40.
- [197] G. Monge. “Mémoire sur la théorie des déblais et des remblais”. In: *Histoire de l’Académie Royale des Sciences de Paris* (1781).

- [198] D. J. H. Moore and G. E. Sharpe. "Metric transformation of an $(m+1)$ -terminal resistive network into a hyperacute angled simplex P_m in Euclidean space E_m ". In: *Proceedings of the 11th IEEE Midwest Symposium on Circuit Theory (MWSCAS'68)*. IEEE, 1968, pp. 184–192.
- [199] J. Munkres. "Algorithms for the assignment and transportation problems". In: *Journal of the society for industrial and applied mathematics* 5.1 (1957), pp. 32–38.
- [200] K. P. Murphy. *Machine learning: a probabilistic perspective*. 1st ed. MIT press, 2012.
- [201] B. Nadler, S. Lafon, I. Kevrekidis, and R. Coifman. "Diffusion maps, spectral clustering and eigenfunctions of Fokker-Planck operators". In: *Proceedings of the 18th Annual Conference on Neural Information Processing Systems (NIPS'05)*. MIT Press, 2005, pp. 955–962.
- [202] M. E. J. Newman. "A measure of betweenness centrality based on random walks". In: *Social Networks* 27.1 (2005), pp. 39–54.
- [203] M. E. J. Newman. "Mixing patterns in networks". In: *Physical Review E* 67.2 (2003), pp. 1–13.
- [204] M. E. J. Newman. "Modularity and community structure in networks". In: *Proceedings of the National Academy of Sciences of the United States of America* 103 (2006), pp. 8577–8582.
- [205] M. E. J. Newman. *Networks*. 2nd ed. Oxford University Press, 2018.
- [206] M. E. J. Newman and M. Girvan. "Finding and evaluating community structure in networks". In: *Physical Review E* 69.2 (2004), pp. 1–15.
- [207] E. W. T. Ngai, Y. Hu, Y. H. Wong, Y. Chen, and X. Sun. "The application of data mining techniques in financial fraud detection: a classification framework and an academic review of literature". In: *Decision support systems* 50.3 (2011), pp. 559–569.
- [208] C. Nguyen and H. Mamitsuka. "New resistance distances with global information on large graphs". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS'16)*. 2016, pp. 639–647.
- [209] Z.-J. Niu, D.-H. Ji, and C. L. Tan. "Word sense disambiguation using label propagation based semi-supervised learning". In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, 2005, pp. 395–402.
- [210] B. Noble and J. W. Daniel. *Applied linear algebra*. 3rd ed. Prentice Hall, 1977.

- [211] J. Nocedal and S. Wright. *Numerical optimization*. 2nd ed. Springer, 2006.
- [212] J. R. Norris. *Markov chains*. 1st ed. Cambridge University Press, 1997.
- [213] P. J. Olver and C. Shakiban. *Applied linear algebra*. 1st ed. Springer, 2006.
- [214] L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank citation ranking: bringing order to the web*. Tech. rep. Stanford InfoLab, 1999.
- [215] S. Pan, R. Hu, S.-F. Fung, G. Long, J. Jiang, and C. Zhang. “Learning graph embedding with adversarial training methods”. In: *IEEE Transactions on Cybernetics* 50.6 (2020), pp. 2475–2487.
- [216] B. Pang and L. Lee. “A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts”. In: *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL’04)*. Association for Computational Linguistics, 2004, pp. 271–278.
- [217] M. Panzacchi, B. Van Moorter, O. Strand, M. Saerens, I. Kivimäki, C. St Clair, I. Herfindal, and L. Boitani. “Predicting the continuum between corridors and barriers to animal movements using step selection functions and randomized shortest paths”. In: *Journal of Animal Ecology* 85.1 (2016), pp. 32–42.
- [218] M. Pavon and F. Ticozzi. “Discrete-time classical and quantum Markovian evolutions: Maximum entropy problems on path space”. In: *Journal of Mathematical Physics* 51.4 (2010), pp. 1–23.
- [219] L. Peliti. *Statistical mechanics in a nutshell*. 1st ed. Princeton University Press, 2011.
- [220] B. Perozzi, R. Al-Rfou, and S. Skiena. “Deepwalk: Online learning of social representations”. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’14)*. 2014, pp. 701–710.
- [221] G. Peyré and M. Cuturi. “Computational Optimal Transport”. In: *Foundations and Trends in Machine Learning* 11.5–6 (2019), pp. 355–607.
- [222] J. I. Ping and K. F. Chu. “A dual-matrix approach to the transportation problem”. In: *Asia-Pacific Journal of Operational Research* 19.1 (2002), pp. 35–45.
- [223] M. Pióro and D. Medhi. *Routing, flow, and capacity design in communication and computer networks*. 1st ed. Elsevier, 2004.

- [224] J. Platt. “Fast training of support vector machines using sequential minimal optimization”. In: *Advances in Kernel Methods - Support Vector Learning* (1998), pp. 41–65.
- [225] S.-D. Poisson. *Recherches sur la probabilité des jugements en matière criminelle et en matière civile: précédées des règles générales du calcul des probabilités*. 1st ed. Bachelier, 1837.
- [226] P. Pons and M. Latapy. “Computing communities in large networks using random walks”. In: *Proceedings of the 20th International Symposium on Computer and Information Sciences (ISCIS’05)*. Vol. 3733. Springer. 2005, pp. 284–293.
- [227] W. L. Price. *Graphs and networks: an introduction*. 1st ed. London Butterworths, 1971.
- [228] A. A. Ramaki, R. Asgari, and R. E. Atani. “Credit card fraud detection based on ontology graph”. In: *International Journal of Security, Privacy and Trust Management (IJSPTM)* 1.5 (2012), pp. 1–12.
- [229] C. R. Rao and S. K. Mitra. “Generalized inverse of a matrix and its applications”. In: *Proceedings of the 6th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics*. The Regents of the University of California. 1972, pp. 601–620.
- [230] S. S. Rao. *Engineering optimization: theory and practice*. 5th ed. John Wiley & Sons, 2019.
- [231] A. Reggiani, L. Schintler, D. Czamanski, and R. Patuelli. *Handbook on entropy, complexity and spatial dynamics*. 1st ed. Edward Elgar Publishing, 2021.
- [232] L. E. Reichl. *A modern course in statistical physics*. 2nd ed. Wiley, 1998.
- [233] N. V. Reinfeld and W. R. Vogel. *Mathematical programming*. 1st ed. Prentice-Hall, 1958.
- [234] R. Rockafellar. “The multiplier method of Hestenes and Powell applied to convex programming”. In: *Journal of Optimization Theory and Applications* 12.6 (1973), pp. 555–562.
- [235] K. H. Rosen and K. Krithivasan. *Discrete mathematics and its applications*. 7th ed. McGraw-Hill, 2013.
- [236] S. M. Ross. *Introduction to probability models*. 11th ed. Academic Press, 2014.
- [237] S. Ryu, A. Chen, X. Xu, and K. Choi. “A dual approach for solving the combined distribution and assignment problem with link capacity constraints”. In: *Networks and Spatial Economics* 14.2 (2014), pp. 245–270.

- [238] M. Saerens, Y. Achbany, F. Fouss, and L. Yen. “Randomized shortest-path problems: two related models”. In: *Neural Computation* 21.8 (2009), pp. 2363–2404.
- [239] M. Saerens, F. Fouss, L. Yen, and P. Dumont. “The principal components analysis of a graph, and its relationships to spectral clustering”. In: *Proceedings of the 15th European Conference on Machine Learning (ECLM’04)*. Vol. 3201. Springer. 2004, pp. 371–383.
- [240] D. Sánchez, M. A. Vila, L. Cerda, and J.-M. Serrano. “Association rules applied to credit card fraud detection”. In: *Expert systems with applications* 36.2 (2009), pp. 3630–3640.
- [241] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. 1st ed. MIT press, 2001.
- [242] J. Schott. *Matrix analysis for statistics*. 2nd ed. Wiley, 2005.
- [243] S. Searle. *Matrix algebra useful for statistics*. 1st ed. Wiley, 1982.
- [244] SeedScientific. *How much data is created every day?* 2022. URL: <https://seedscientific.com/how-much-data-is-created-every-day> (visited on March 14, 2022).
- [245] D. Serre. *Matrices, theory and applications*. 1st ed. Springer, 2002.
- [246] J. K. Sharma. *Operations research: theory and applications*. 6th ed. Trinity Press, an imprint of Laxmi Publications Pvt. Limited, 2016.
- [247] G. E. Sharpe. “On the $(m + 1)$ -terminal resistive network problem”. In: *Proceedings of the Institution of Electrical Engineers*. Vol. 116. 4. 1969, pp. 503–509.
- [248] G. E. Sharpe. “Solution of the $(m + 1)$ -terminal resistive network problem by means of metric geometry”. In: *Proceedings of the 1st Asilomar Conference on Circuits and Systems*. 1967, pp. 319–328.
- [249] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. 1st ed. Cambridge university press, 2004.
- [250] A. Shen, R. Tong, and Y. Deng. “Application of classification models on credit card fraud detection”. In: *Proceedings of the 4th International Conference on Service Systems and Service Management (ICSSSM’07)*. IEEE. 2007, pp. 1–4.
- [251] D. Shimshak, J. Alan Kaslik, and T. Barclay. “A modification of Vogel’s approximation method through the use of heuristics”. In: *INFOR: Information Systems and Operational Research* 19.3 (1981), pp. 259–263.

- [252] A. J. Smola and R. Kondor. “Kernels and regularization on graphs”. In: *Proceedings of the 16th Annual Conference on Learning Theory and 7th Kernel Workshop (COLT-Kernel’03)*. Vol. 2777. Springer, 2003, pp. 144–158.
- [253] F. Sommer, F. Fouss, and M. Saerens. “Comparison of graph node distances on clustering tasks”. In: *Proceedings of the 25th International Conference on Artificial Neural Networks (ICANN’16)*. Vol. 9886. Springer, 2016, pp. 192–201.
- [254] F. Sommer, F. Fouss, and M. Saerens. “Modularity-driven kernel k-means for community detection”. In: *Proceedings of the 26th International Conference on Artificial Neural Networks (ICANN’17)*. Vol. 10614. Springer, 2017, pp. 423–433.
- [255] Z. Song, X. Yang, Z. Xu, and I. King. “Graph-based semi-supervised learning: A comprehensive review”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2022), pp. 1–21.
- [256] D. Spielman and S.-H. Teng. “Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems”. In: *Proceedings of the annual ACM Symposium on the Theory of Computing (STOC’04)*. Vol. 4. ACM, 2004, pp. 81–90.
- [257] A. Srivastava, A. Kundu, S. Sural, and A. Majumdar. “Credit card fraud detection using hidden Markov model”. In: *IEEE Transactions on dependable and secure computing* 5.1 (2008), pp. 37–48.
- [258] N. Storozhyshina, F. Pargar, and F. J. Vasko. “A comprehensive empirical analysis of 16 heuristics for the transportation problem”. In: *OR insight* 24.1 (2011), pp. 63–76.
- [259] A. Subramanya and P. P. Talukdar. “Graph-based semi-supervised learning”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8.4 (2014), pp. 1–125.
- [260] V. J. Sudhakar, N. Arunsankar, and T. Karpagam. “A new approach for finding an optimal solution for transportation problems”. In: *European journal of scientific Research* 68.2 (2012), pp. 254–257.
- [261] M. Szummer and T. Jaakkola. “Partially labeled classification with markov random walks”. In: *Proceedings of the 14th International Conference on Neural Information Processing Systems (NIPS’01)*. MIT Press, 2001, pp. 945–952.
- [262] T. H. Szymanski. “Max-flow min-cost routing in a future-Internet with improved QoS guarantees”. In: *IEEE transactions on communications* 61.4 (2013), pp. 1485–1497.

- [263] S. Tabassum, F. S. F. Pereira, S. Fernandes, and J. Gama. "Social network analysis: an overview". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.5 (2018), pp. 1–21.
- [264] H. A. Taha. *Operations research: an introduction*. 8th ed. Prentice Hall, 2007.
- [265] A. Tahbaz-Salehi and A. Jadbabaie. "A one-parameter family of distributed consensus algorithms with boundary: From shortest paths to mean hitting times". In: *Proceedings of the 45th IEEE Conference on Decision and Control (CDC'06)*. IEEE. 2006, pp. 4664–4669.
- [266] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. "Line: Large-scale information network embedding". In: *Proceedings of the 24th International Conference on World Wide Web (WWW'15)*. 2015, pp. 1067–1077.
- [267] L. Tang and H. Liu. "Relational learning via latent social dimensions". In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*. ACM. 2009, pp. 817–826.
- [268] L. Tang and H. Liu. "Scalable learning of collective behavior based on sparse social dimensions". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM'09)*. ACM. 2009, pp. 1107–1116.
- [269] L. Tang and H. Liu. "Toward predicting collective behavior via social dimension extraction". In: *IEEE Intelligent Systems* 25.4 (2010), pp. 19–25.
- [270] S. Theodoridis and K. Koutroumbas. *Pattern recognition*. 4th ed. Academic Press, 2008.
- [271] E. Todorov. "General duality between optimal control and estimation". In: *Proceedings of 47th IEEE Conference on Decision and Control (CDC'08)*. IEEE. 2008, pp. 4286–4292.
- [272] E. Todorov. "Linearly-solvable Markov decision problems". In: *Proceedings of the 19th Annual Conference on Neural Information Processing Systems (NIPS'06)*. MIT Press, 2006, pp. 1369–1375.
- [273] H. Tong, C. Faloutsos, and J.-Y. Pan. "Fast random walk with restart and its applications". In: *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM'06)*. IEEE. 2006, pp. 613–622.
- [274] P. Toth and D. Vigo. *The vehicle routing problem*. 1st ed. SIAM, 2002.
- [275] C. Tsallis. "Generalized entropy-based criterion for consistent testing". In: *Physical Review E* 58.2 (1998), pp. 1442–1445.

- [276] C. Tsallis. *Introduction to nonextensive statistical mechanics*. 1st ed. Springer, 2009.
- [277] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu. “Deep recursive network embedding with regular equivalence”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’18)*. ACM, 2018, pp. 2357–2366.
- [278] M. Udell, K. Mohan, D. Zeng, J. Hong, S. Diamond, and S. Boyd. “Convex optimization in Julia”. In: *Proceedings of the 1st IEEE Workshop for High Performance Technical Computing in Dynamic Languages (HPCWPL’14)*. IEEE, 2014, pp. 18–28.
- [279] S. Vajda. *Mathematical programming*. 1st ed. Addison-Wesley, 1961.
- [280] B. Van Moorter, I. Kivimäki, A. Noack, R. Devooght, M. Panzacchi, K. R. Hall, P. Leleux, and M. Saerens. “Accelerating advances in landscape connectivity modeling with the ConScape library”. In: *Methods in Ecology and Evolution* (2022), pp. 1–34.
- [281] V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akogu, M. Snoeck, and B. Baesens. “APATE: a novel approach for automated credit card transaction fraud detection using network-based extensions”. In: *Decision Support Systems* 75 (2015), pp. 38–48.
- [282] C. Villani. *Optimal transport: old and new*. 1st ed. Springer, 2008.
- [283] C. Villani. *Topics in optimal transportation*. 1st ed. American Mathematical Society, 2003.
- [284] U. von Luxburg, A. Radl, and M. Hein. “Getting lost in space: large sample analysis of the commute distance”. In: *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS’10)*. MIT Press, 2010, pp. 2622–2630.
- [285] D. Wackerly, W. Mendenhall, and R. Scheaffer. *Mathematical statistics with applications*. 7th ed. Cengage Learning, 2008.
- [286] B. Wang, Y. Sun, T. Q. Duong, L. D. Nguyen, and L. Hanzo. “Risk-Aware identification of highly suspected COVID-19 cases in social IoT: a joint graph theory and reinforcement learning approach”. In: *IEEE Access* 8 (2020), pp. 115655–115661.
- [287] D. Wang, P. Cui, and W. Zhu. “Structural deep network embedding”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’16)*. ACM, 2016, pp. 1225–1234.
- [288] J. Wang, F. Wang, C. Zhang, H. C. Shen, and L. Quan. “Linear neighborhood propagation and its applications”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.9 (2008), pp. 1600–1615.

- [289] S. Warshall. "A theorem on boolean matrices". In: *Journal of the ACM (JACM)* 9.1 (1962), pp. 11–12.
- [290] D. B. West. *Introduction to graph theory*. 2nd ed. Prentice Hall, 2000.
- [291] D. J. Weston, D. J. Hand, N. M. Adams, C. Whitrow, and P. Juszczak. "Plastic card fraud detection using peer group analysis". In: *Advances in Data Analysis and Classification* 2.1 (2008), pp. 45–62.
- [292] R. Wheeler and S. Aitken. "Multiple algorithms for fraud detection". In: *Applications and Innovations in Intelligent Systems VII*. Springer, 2000, pp. 219–231.
- [293] F. Wilcoxon. "Individual comparisons by ranking methods". In: *Biometrics Bulletin* 1.6 (1945), pp. 80–83.
- [294] A. C. Williams. "A treatment of transportation problems by decomposition". In: *Journal of the Society for Industrial and Applied Mathematics* 10.1 (1962), pp. 35–48.
- [295] D. P. Williamson. *Network flow algorithms*. 1st ed. Cambridge University Press, 2019.
- [296] A. G. Wilson. "A statistical theory of spatial distribution models". In: *Transportation Research* 1.3 (1967), pp. 253–269.
- [297] A. G. Wilson. *Entropy in urban and regional modelling*. 1st ed. Routledge, 1970.
- [298] L. Yen, F. Fouss, C. Decaestecker, P. Francq, and M. Saerens. "Graph nodes clustering based on the commute-time kernel". In: *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'07)*. Springer, 2007, pp. 1037–1045.
- [299] L. Yen, F. Fouss, C. Decaestecker, P. Francq, and M. Saerens. "Graph nodes clustering with the sigmoid commute-time kernel: a comparative study". In: *Data & Knowledge Engineering* 68.3 (2009), pp. 338–361.
- [300] L. Yen, M. Saerens, A. Mantrach, and M. Shimbo. "A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*. ACM, 2008, pp. 785–793.
- [301] J. Young. *Global e-commerce sales grow 18% in 2018*. 2018. URL: <https://www.digitalcommerce360.com/article/global-e-commerce-sales> (visited on August 13, 2019).

- [302] J. Yuan, E. Bae, and X.-C. Tai. "A study on continuous max-flow and min-cut approaches". In: *Proceedings of the 15th IEEE International Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'10)*. IEEE. 2010, pp. 2217–2224.
- [303] U. Yüceer and A. Özakç. "A truck loading problem". In: *Computers & Industrial Engineering* 58.4 (2010), pp. 766–773.
- [304] M. J. Zaki and W. Meira. *Data mining and analysis: Fundamental concepts and algorithms*. 2nd ed. Cambridge University Press, 2020.
- [305] V. Zaslavsky and A. Strizhak. "Credit card fraud detection using self-organizing maps". In: *Information and Security* 18 (2006), pp. 48–63.
- [306] D. Zhang and R. Mao. "A new kernel for classification of networked entities". In: *Proceedings of 6th International Workshop on Mining and Learning with Graphs (MLG'08)*. 2008, pp. 1–4.
- [307] D. Zhang and R. Mao. "Classifying networked entities with modularity kernels". In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*. ACM. 2008, pp. 113–122.
- [308] Z. Zhang, X. Zhou, X. Zhang, L. Wang, and P. Wang. "A model based on convolutional neural network for online transaction fraud detection". In: *Security and Communication Networks* (2018), pp. 1–10.
- [309] Y. Zhao, R. Ball, J. Mosesian, J.-F. de Palma, and B. Lehman. "Graph-based semi-supervised learning for fault detection and classification in solar photovoltaic arrays". In: *IEEE Transactions on Power Electronics* 30.5 (2015), pp. 2848–2858.
- [310] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. "Learning with local and global consistency". In: *Proceedings of the 16th Annual Conference on Neural Information Processing Systems (NIPS'03)*. MIT Press, 2004, pp. 321–328.
- [311] D. Zhou, J. Huang, and B. Schölkopf. "Learning from labeled and unlabeled data on a directed graph". In: *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*. 2005, pp. 1036–1043.
- [312] T. Zhou, L. Lü, and Y.-C. Zhang. "Predicting missing links via local information". In: *The European Physical Journal B* 71.4 (2009), pp. 623–630.
- [313] X. Zhou, S. Cheng, M. Zhu, C. Guo, S. Zhou, P. Xu, Z. Xue, and W. Zhang. "A state of the art survey of data mining-based fraud detection and credit scoring". In: *Proceedings of the 2nd International Conference on Material Engineering and Advanced Manufacturing Technology (MEAMT'18)*. 2018, pp. 1–15.

- [314] X. Zhu and A. B. Goldberg. “Introduction to semi-supervised learning”. In: *Synthesis lectures on artificial intelligence and machine learning* 3.1 (2009), pp. 1–130.